On Proofs and Translation

Orr Paradise

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2025-92
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2025/EECS-2025-92.html

May 16, 2025

On Proofs and Translation

by

Orr Paradise

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Shafi Goldwasser, Co-chair
Assistant Professor Avishay Tal, Co-chair
Assistant Professor Nikita Zhivotovskiy

Spring 2025

On Proofs and Translation

Abstract

On Proofs and Translation

by

Orr Paradise

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Shafi Goldwasser, Co-chair

Assistant Professor Avishay Tal, Co-chair

This dissertation examines proof systems and translation methods, addressing both theoretical foundations and practical considerations in each domain.

The first part, on Proofs, introduces rectangular probabilistically checkable proofs (rectangular PCPs), wherein proofs are thought of as square matrices, and the verifier's randomness can be split into two independent parts, one determining the row of each query and the other determining the column. We construct rectangular PCPs and use them to show that proofs for hard languages are rigid—extending and strengthening recent rigid matrix constructions.

We then propose Self-Proving models: learned models that prove the correctness of their output to a verification algorithm via an Interactive Proof. We devise a generic method for learning Self-Proving models, and prove its convergence under certain assumptions. We empirically examine our methods by training a Self-Proving transformer to compute the GCD of two integers, and prove correctness of its output. We also introduce Pseudointelligence, a complexity-theoretic framework of model evaluation cast as an interactive proof between a model and a learned evaluator.

The second part, on Translation, explores unsupervised machine translation (UMT) without shared linguistic structure. We develop a theoretical framework for analyzing this setting, and prove sample complexity bounds in stylized yet informative settings. The results show that translation quality improves with language complexity, informing feasibility of animal communication translation. Finally, we present WhAM, a transformer-based model for generating synthetic sperm whale codas. WhAM is trained on real acoustic data and generates audio that approaches the statistical and perceptual properties of whale communication as evaluated by domain experts. Its learned representations also perform well on classification tasks, contributing to our understanding of non-human communication systems.

Contents

List of Figures

List of Tables

Acknowledgments

The hardest part about writing this dissertation is knowing that, upon its submission, I will no longer be the student of my advisors, Shafi Goldwasser and Avishay Tal. During my doctoral studies much has changed, not always for the better, and often at very short notice. Yet, you were always there when I wanted to chat (even when I had nothing "interesting" to say, for months on end). Thank you for the complete—at times terrifying—freedom to carve my own path, and for sharing in the excitement whenever a spark was found. When I ventured beyond theoretical computer science, I was given no push-back nor pressure to return; instead, I was given space to discover which direction to pursue (or create). And for your open arms when these explorations eventually converged back with theory—a full circle that neither of us could have planned at the outset.

Avishay and Shafi are, by now, more than academic family. I was looking forward to your home-cooked dinners as much as I was to our research meetings. Indeed, you not only opened doors for me, but had me forgetting that there was ever a door to be opened; the joy of being fully immersed in research for these years, without "practical" concerns, is something I hope to provide my students one day.

I am grateful to at least two other scientific mentors during this chapter: Adam Tauman Kalai, whose enthusiasm encouraged me to venture beyond traditional complexity theory, and David F. Gruber, who expanded my vision of what science could be. Frankly, I never imagined working on decoding whale communication as a part of my doctoral research, and sometimes it still feels like a dream. Thank you both for creating spaces where wonder and scientific rigor coexist, and for treating intellectual curiosity as the most important metric. And to Amey Bhangale and Eylon Yogev for providing older-sibling-like support as I was transitioning into these doctoral studies—things were much easier thanks to you.

The students in the Berkeley EECS department transformed what could have been an isolated academic pursuit into a shared adventure. I'm particularly grateful to the undergraduate who worked with me: Circle Chen, Pranav Muralikrishnan, Annamira O'Toole, and Chirag Sharma. Thank you for letting me learn research mentorship "from the other side." It was a pleasure watching you grow.

The administrative staff at UC Berkeley EECS—Carissa Caloud, Susanne Kauer, and Jean Nguyen—along with Amy Ambrose, Frida Orre, and Carson Young at the Simons Institute ensured that my challenges were mostly non-bureaucratic.

Thank you to Alane Suhr, Gasper Begus, and Nikita Zhivotovskiy for helpful comments and guidance via my qualifying and dissertation committees. To Alessandro Chiesa, for support during my first semester. And to Massimo Mazzotti for teaching me Science, Technology and Society (STS), an impactful class, which I took during my second year.

Serving alongside colleagues at the EECS Peer Advising program and the Equal Access to Application Assistance program was an important part of this journey. Alok Tripathy, thank you for coordinating the student-side of the faculty hiring committee with me—all aspects of this role were enjoyable, because we did them together.

Various institutions welcomed me as a visitor, and for this I thank my gracious hosts: Amey Bhangale, Michael Bronstein, Jonah Brown-Cohen, Ned Caisley, Josephine Eberhardt, Oneg Eckerling, Tom Gur, Robin Labsch, Michele Orrù, Ruzica Piskac, Rüdiger Urbanke, and the Zuse Institute Berlin. Un grand merci aux Devillards—multiple chapters of this dissertation were written while enjoying your hospitality.

I've been fortunate to work with extraordinary coauthors: Noga Amit, Timos Antonopoulos, James Bartusek, Yonatan Belinkov, Rotem Ben Zion, Thiago Bergamaschi, Amey Bhangale, Dave Bignell, Boaz Carmeli, Micah Carroll, Circle Chen, Zaria Chen Shui, Daniel N. Coore, Roee Diamant, Anca Dragan, Ferhat Erata, Daniel T. Fokum, Hugo Flores García, Raluca Georgescu, Shane Gero, Shafi Goldwasser, David F. Gruber, Sam Gunn, Prahladh Harsha, Matthew Hausknecht, Katja Hofmann, Doseok Jang, Adam Tauman Kalai, Seri Khoury, Ido Levy, Jessy Lin, Gunjan Mansingh, Ron Meir, Stephanie Milani, Saachi Mutreja, Shikhar Murty, Pranav Muralikrishnan, ThanhVu Nguyen, Bryan Pardo, Ruzica Piskac, Guy N. Rothblum, Pratyusha Sharma, Lucas Spangher, Costas J. Spanos, Mingfei Sun, Avishay Tal, and Kerene Wright. A special thank you to Micah Carroll and Lucas Spangher for saying, at a particularly intimidating moment, that I should just "run some experiments." That encouragement was instrumental.

I often use the terms *challenging and rewarding* to describe this doctorate, and JamCoders epitomized this combination, year after year, one month at a time. Thank you to the organizing team, lecturers, chaperones, and Miss Barnett. To our Jamaican hosts who welcomed us so patiently and made Kingston feel like home each summer. To the donors who made it possible, and especially to Chronixx for his support and inspiration.

The JamCoders experience wouldn't have been the same without my fellow teaching assistants, who became flatmates and comrades in what was always far more than we signed up for: Tarun Amarnath, Bryan Baker, Micah Carroll, Zaria Chen Shui, James Cheng, Li Dayan, Ian DeHaan, Reginald Frank, Michael Girma, Emaan Hariri, Jabari Hastings, Xavier Henry, Tyler Hou, Ecy King, Anita Liu, Bereket Molla, Annamira O'Toole, Anakai Richards, Jonathan Shafer, Nadia Sharp, Elijah Tai, Jonathan Tay, Liam Tan, Natnael Teshome, Alok Tripathy, Pawat Unjitwattana, Kerene Wright, and Kimberli Zhong.

JamCoders' most important component is, of course, the JamCoders themselves—all 150 high-schoolers across three iterations. I've had the privilege of learning from (and teaching) each of you throughout this program. You are the brightest folks I've met at graduate school, and I consider myself lucky that our paths crossed so early in your academic voyage.

The Lighthouse and Hillegass–Parker cooperatives were much more than just affordable housing: There I've learned electronics, political science, statistics, history, philosophy, linguistics, finance, and how to unclog a sink. Propelling me forward while distracting me from my own research—I am grateful for both. Thank you, Adam Zaid Austin Bouyamourn and Henri Danavrett Wadsworth, for picking me up so effortlessly when I was down.

To my musical collaborators here in Berkeley—there were many, and you were all integral to my experience. I especially want to acknowledge those with whom I've performed: the Melon Collective; Michael Jordan and the Bulls (Anastasios Angelopolous, Amit Kohli, Mariel Werner, Lydia Zakynthinou); James Bartusek, Lena Bhadia, Gianna Caudillo, Hank

Figueroa Lindelli, Samantha Friedland, Luc Le Pottier, Ian Mitchell, Andrew Ntim, and Ian Waudby-Smith. A special thank you to Ben Goldberg and his class, for teaching me to *make a mess* and *mean it*—lessons I've found useful in research as well. And to Tijana Zrnic.

Thank you to my friends from back home for the many notions of "home," whose timezone-spanning presence meant I always had someone to talk to no matter the time of day. Especially Shahar Bracha, Kelsey Greenway, and Alon Gurman whom I called often.

To my family: I've learned that it's sometimes better to replace "I'm sorry" with "Thank you." So thank you for letting me be far away for so long. I miss you dearly.

Once I realized that a doctorate is a journey of self-understanding as much as a scientific one, the endeavor seemed much more worthwhile. I am deeply grateful to my therapist for accompanying me on that aspect of the journey. And to readers of this paragraph who are in graduate school: I strongly encourage you to find a therapist, too.

Introduction

This dissertation traces a six-year research trajectory that traverses seemingly distant domains: from the foundations of proof systems in theoretical computer science to the application of machine learning for deciphering non-human communication. What began as a study of probabilistically checkable proofs (PCPs) and their structural properties gradually expanded into a broader investigation of verification—how we come to trust claims, systems, and signals—across both formal and empirical settings.

This progression was not preordained. My early work on specialized PCPs revealed deep connections between semantic hardness and syntactic complexity. These findings, rooted in worst-case complexity theory, raised new questions about verification in domains where formal guarantees are rare—particularly in the emerging landscape of large language models. As these models became increasingly capable yet remained difficult to interpret or test rigorously, the need for principled approaches to machine-verifiable reasoning led to my work on Self-Proving models and the broader framework of pseudointelligence.

Around the same time, I became involved with the Cetacean Translation Initiative (CETI), which seeks to understand sperm whale communication through computational means. At first, this seemed far afield from my previous work. But as I engaged more deeply, unexpected parallels emerged: both in theory and in practice, I was grappling with the challenge of interpreting signals in the absence of ground truth. The verification problems I had studied in formal contexts—How can we validate a claim when we cannot directly observe its justification?—resurfaced in this new domain in a different guise.

Overview

This dissertation is organized into two parts. Part 1 focuses on proof systems with provable guarantees, developing theoretical foundations for verification in computational systems. Part 2 explores unsupervised translation theories, applying them to both human and non-human communication challenges. Together, these parts span from abstract theoretical foundations to practical applications, offering novel frameworks for establishing trust and evaluating capabilities in complex systems. Next, I provide a an overview of each chapter's contributions.

Part 1: Proofs

In **Chapter 1**, we[2] introduce a variant of Probabilistically Checkable Proofs (PCPs) called *rectangular* PCPs. In this formulation, proofs are conceptualized as square matrices, and the random coins used by the verifier can be partitioned into two disjoint sets—one determining the *row* of each query and the other determining the *column*. We construct PCPs that are *efficient*, *short*, *smooth* and (almost-)*rectangular*. As a key application, we demonstrate that proofs for hard languages in $\mathsf{NTIME}(2^n)$, when viewed as matrices, are rigid infinitely often. This strengthens and simplifies a recent result by Alman and Chen (2019) constructing explicit rigid matrices in FNP. Specifically, we prove that there exists a constant $\delta \in (0, 1)$ such that there is an FNP-machine that, for infinitely many N, on input 1^N outputs $N \times N$ matrices with entries in $\mathbb{F}_2$ that are δN^2-far (in Hamming distance) from matrices of rank at most $2^{\log N/\Omega(\log \log N)}$.

Our construction of rectangular PCPs begins with an analysis of how randomness yields queries in the Reed–Muller-based outer PCP of Ben-Sasson et al. (2005). We then demonstrate how to preserve rectangularity under PCP composition and a smoothness-inducing transformation, requiring refined and stronger notions of rectangularity that we prove for the outer PCP and its transforms.

While Chapter 1 explores proof systems within their traditional theoretical foundations, **Chapter 2** extends these notions to address emerging challenges in machine learning and AI. Moving from complexity theory to more practical verification concerns, we adapt the formal guarantees of proof systems to the pressing domain of trustworthy generative modeling.

This chapter addresses a fundamental question in machine learning: How can we trust the correctness of a learned model on a particular input of interest? Conventional model accuracy is typically measured *on average* over a distribution of inputs, providing no guarantee for any specific input. We propose a theoretically-founded solution to this problem: to train *Self-Proving models* that prove the correctness of their output to a verification algorithm V via an Interactive Proof.

Self-Proving models satisfy that, with high probability over an input sampled from a given distribution, the model generates a correct output *and* successfully proves its correctness to V. The *soundness* property of V guarantees that, for *every* input, no model can convince V of the correctness of an incorrect output. Thus, a Self-Proving model proves correctness of most of its outputs, while *all* incorrect outputs (of any model) are detected by V. We devise generic methods for learning Self-Proving models and prove their convergence under certain assumptions. Our theoretical framework is complemented by experiments on an arithmetic capability: computing the greatest common divisor (GCD) of two integers. Using our learning method, we train a Self-Proving transformer that computes the GCD *and* proves the correctness of its answer.

[2]Throughout this dissertation, I use the first person plural pronoun following the standard convention in computer science and mathematics. Furthermore, all chapters presented here are based on collaborative work with various coauthors (see Appendix A.5).

Chapter 3 extends proof systems to address the broader challenge of evaluating intelligence in computational systems. With large language models surpassing human performance on an increasing number of benchmarks, we argue for a principled approach to targeted evaluation of model capabilities. Inspired by pseudorandomness, we propose *pseudointelligence*, which captures the maxim that "(perceived) intelligence lies in the eye of the beholder." This framework acknowledges that claims of intelligence are meaningful only when their evaluator is taken into account.

Concretely, we propose a complexity-theoretic framework of model evaluation cast as an interactive proof between a model and a learned evaluator. We demonstrate that this framework can be used to reason about two case studies in language model evaluation and analyze existing evaluation methods.

Part 2: Translation

Chapter 4 transitions to the domain of translation, specifically unsupervised translation where parallel corpora are unavailable. Neural networks have demonstrated the capability to translate between languages—in some cases even between two languages with little or no access to parallel translations, known as Unsupervised Machine Translation (UMT). Given this progress, we explore whether machine learning tools can ultimately enable understanding animal communication, particularly that of highly intelligent animals.

We propose a theoretical framework for analyzing UMT when no parallel translations are available and when it cannot be assumed that the source and target corpora address related subject domains or possess similar linguistic structure. We exemplify this theory with two stylized models of language, for which our framework provides bounds on necessary sample complexity; these bounds are formally proven and experimentally verified on synthetic data. Our results show that error rates are inversely related to language complexity and amount of common ground, suggesting that unsupervised translation of animal communication may be feasible if the communication system is sufficiently complex.

Bridging theory and application, **Chapter 5** shifts our focus to the practical challenge of modeling acoustic elements in non-human communication. There, we engage with the reality of working with sperm whale vocalizations. For sperm whales, who communicate through short sequences of clicks known as codas, acoustic modeling is a fundamental component of any translation effort. We present WhAM (Whale Acoustics Model), the first transformer-based model capable of generating synthetic sperm whale codas from any audio prompt. WhAM is built by finetuning VampNet, a masked acoustic token model pretrained on musical audio, using 10,000 coda recordings collected over the past two decades.

Through iterative masked token prediction, WhAM generates high-fidelity synthetic codas that preserve key acoustic features of the source recordings. We evaluate WhAM's synthetic codas using Fr'echet Audio Distance and through perceptual studies with expert marine biologists. On downstream classification tasks including rhythm, social unit, and vowel classification, WhAM's learned representations achieve strong performance, despite being trained for generation rather than classification.

Proofs as translation

Over the course of this dissertation, my research has moved across disciplinary boundaries—from complexity theory to applied machine learning to translation and evaluation in biological systems. These projects appeared in venues spanning theoretical computer science, machine learning, and natural language processing. Navigating these transitions often felt disorienting. The standards of evidence, the language of rigor, and even the definitions of success shifted dramatically between communities. And yet, throughout this journey, the concept of a *proof*—broadly construed—reappeared in many forms. Sometimes it was formal and mathematical, sometimes embodied in experimental design, and sometimes purely based on "vibes" (to quote a collaborator). At times, I felt that the act of constructing or interpreting proofs became a way of *translating* between research cultures: of asserting compatibility across different standards, and of enabling conversation without consensus.

To make sense of this recurring role, I found it helpful to borrow the notion of a *boundary object* from science and technology studies (STS) (Star and Griesemer, 1989; Leigh Star, 2010). I do not claim to apply this concept with disciplinary rigor—this is not an argument within STS, but a framing that helped me understand how one idea could persist, transform, and hold things together across disparate contexts. Perhaps proofs, as they function in this dissertation, operate at the boundary between disciplines: flexible enough enough establish claims in theoretical computer science, adapt to the needs of applied AI systems, and guide the evaluation of cross-species communication all at once. Towards that end, I borrow from boundary object theory not to claim that proofs *are* boundary objects in some strict ontological sense, but to reflect on how they helped me move across and between domains.

Very briefly, boundary objects (Star and Griesemer, 1989) are entities that inhabit multiple communities of practice and are used differently by each, yet maintain enough identity to serve as sites of coordination. They are "plastic enough to adapt to local needs... yet robust enough to maintain a common identity across sites" (Star and Griesemer, 1989, p. 393). Proofs, in this dissertation, may function in this way. They appear in at least five forms, mapped to the five chapters.

In Chapter 1, Probabilistically Checkable Proof systems (PCPs) are "constructed" towards improving the rigid matrices of Alman and Chen (2022). Here "constructed" is used in the theoretical sense: these proof systems are *not* meant to (in fact, cannot) be implemented in software.[3]

In Chapter 2, we devise a framework for establishing trust in AI systems. Namely, we introduce Self-Proving models that generate proofs of correctness alongside their outputs. Formally, we require such models to convince a verifier in an Interactive Proof system (IP). IPs and PCPs share similar roots (Bellare et al., 1994), but unlike Chapter 1, Self-Proving models are intended to be implemented in practice. Indeed, we released a library implementing a Self-Proving GPT and used it for our experiments on provably generating the Greatest

[3]Specifically, they are used for showing that matrices of certain rigidity can be constructed in nondeterministic polynomial time (FNP)—a step towards obtaining circuit lower bounds for FNP. See Section 1.2.

Common Divisor in Section 2.3.

Similarly, Chapter 3 relies on IPs to formally capture the interaction between an allegedly intelligent model and a (learned) evaluator of a given capability. This approach reimagines the classic Imitation Game of Turing (1950) through the lens of interactive proof systems. By formalizing Turing's Test within a complexity-theoretic framework, we place evaluators at the center of claims on "intelligence" claims, arguing that such claims are meaningful only with respect to specific evaluators (what we term *pseudointelligence*). Here too, proof systems are the glue that connect formal, provable guarantees with existing real-world evaluation systems (see Section 3.2), translating between philosophical notions of machine intelligence and the technical rigor of complexity theory.

In Chapter 4, the focus shifts. There is no explicit proof system embedded in the algorithmic pipeline, nor a verifier exchanging messages with a model. Instead, proof resurfaces in a more foundational sense: we prove theorems about the (im)possibility of unsupervised translation between languages. These results are not about verifying individual outputs, but rather about certifying structural conditions under which meaningful translation is possible at all. In this context, proofs serve as a tool for reasoning about semantic fidelity in the absence of direct supervision. The boundary object quality of proofs is especially visible here: their form has shifted from a protocol to a conceptual mechanism for reasoning across representational systems. Still, their core function—establishing fidelity or correctness under uncertainty—remains.

Finally, in Chapter 5, the role of proofs becomes fully embedded in practice. There are no theorems nor proof system protocols; rather, we face a novel challenge: how does one evaluate a model trained to acoustically translate sperm whale vocalizations? One of this chapter's main contributions is precisely to introduce an evaluation suite for this task—a concrete methodology for determining whether generated outputs plausibly align with real whale codas. This includes an experiment in which domain experts attempt to distinguish between real and synthetic vocalizations. The fact that their success rate is only moderate offers a form of proof of indistinguishability: it confirms that the model's outputs are, to some degree, indistinguishable from authentic ones. In the terms of Chapter 3, we could say that the model is *pseudointelligent* with respect to the capability of whale communication, and our experts act as the evaluators. Although informal, this setup recapitulates many of the structural elements that appear in more formal proof systems: an interaction, a challenge, and a verdict. The vocabulary has changed, but the underlying concern—how we establish that something holds true, behaves faithfully, or mimics a target concept—persists.

Despite their differences, these instantiations all share a core concern: establishing correctness, fidelity, or validity. That common concern is what allows proofs to function—to some degree—as a boundary object. They are "ill-structured" enough to be adapted to the epistemic needs of each field, but structured enough to retain a recognizable identity across them. Of course, this coherence is not automatic—it is produced through a kind of intellectual boundary work. In each chapter, proofs are adapted, formalized, or instantiated according to the epistemic standards of the relevant community.

This makes them not just a recurring theme, but a kind of conceptual interface—

something that allows ideas to travel from one domain to another, to be reinterpreted, applied, and sometimes transformed in the process. Specifically, this process spans communities such as theoretical computer scientists concerned with worst-case guarantees, machine learning researchers focused on empirical reliability, and biologists working on animal communication. Each of these groups brings distinct assumptions, methodologies, and validation norms, yet proofs—understood broadly as demonstrations of correctness—offer enough structure to enable conceptual exchange across these boundaries. In this sense, they operate as what Carlile (2002) calls a "translation device": a shared structure that enables communication across domains with different languages and standards.

From this perspective, proofs become more than formal tools; they become a kind of *translator* across domains. They connect formal claims to empirical evaluation, and computational structure to communication—not by collapsing their differences, but by offering a shared point of reference. In that sense, the role proofs play in this dissertation echoes its structure: it begins in proofs, ends in translation, and uses proofs to move between them. This casts the title *On Proofs and Translation* in a new light. The *and* is not just a conjunction linking two separate parts; it is a hinge, a site of composition. Through the lens of boundary objects, we might even say that proofs *are* a kind of translation—specifically, translation across epistemic communities. In each domain, they facilitate the movement of meaning across boundaries: from prover to verifier, from model to user, from one species to another.

I want to be careful not to overstate this theme. While the boundary object framing has been generative for me, Leigh Star (2010) cautions against reducing the concept to mere interpretive flexibility without considering its full architecture of informatic structures and the dynamic between ill-structured and well-structured uses. To be clear, *Proofs as Translation* is not offered as a universal solvent, but as an adhesive—a way of reading the dissertation as more than a collection of technical chapters. It frames a burgeoning dialogue between computational complexity, statistical learning theory, and ethology. And it helps explain why the chapters that follow feel to me like parts of the same story.

Notation

Before proceeding to the technical chapters, we establish the notation and conventions used throughout this dissertation. For an integer $n \in \mathbb{N}$ we let $[n] \coloneqq \{1, \ldots, n\}$.

$\mathbb{E}[X]$ denotes the expected value of a random variable X; we write $x \sim X$ to denote a single sample, and $(x_1, \ldots, x_n) \sim X^n$ for n many independent samples—although we may sometimes simply write $x_1, \ldots, x_n \sim X$. $\Pr[\omega]$ denotes the probability of an event ω.

For a set S, we write S^n to denote the n-fold Cartesian product $S \times \cdots \times S$, and $S^* \coloneqq \bigcup_{n \in \mathbb{N}} S^n$. When given a finite set Σ as an *alphabet*, then Σ^n denotes *strings* of length n and Σ^* denotes all finite strings. For a symbol $\sigma \in \Sigma$, we use σ^n to denote its n-fold repetition. In particular, when working over the Boolean alphabet $\Sigma = \{0, 1\}$, then 1^n should be read

as "the string of n many ones—not as "one to the power of n". For any string $x \in \Sigma^*$, we use $|x|$ to denote the length of x.

A set of strings $S \subseteq \Sigma^*$ (also known as *language* in the literature) induces a natural decision problem, namely, given a string x decide whether $x \in S$. A relation $R \subseteq \Sigma^* \times \Sigma^*$ induces a search problem: given a string x, find y such that $(x, y) \in R$. In this thesis, relations will capture notions "correctness': thinking of x as the *input*, y is a *correct output* for x if $(x, y) \in R$.[4] Without loss of generality, we will assume that every input x has at least one correct output by introducing a special string $\perp$ that indicates no solution was (previously) found.

We refer to Turing machines throughout this thesis. These can be either deterministic or probabilistic. A probabilistic (also known as randomized) Turing machine has access to internal coin flips. An *oracle Turing machine* can query an external string (equivalently, function) *oracle* during its computation. We write $V^{\pi}(x)$ to denote the Turing machine V given (full access to) input x and oracle access to π.

A Turing machine V may *interact* with an external "all-powerful"[5] P by issuing queries and receiving answers (both are strings). We write $\langle V, P \rangle (x)$ to denote the interaction of V with P when both are given the input x. Note that, unlike oracle access, in an interaction P may change its answers based on previous queries from V.

The term *algorithm* is used interchangeably with (deterministic or randomized) Turing machine. By default, *efficient* means running in time polynomial in the length of the input. When there are multiple inputs (e.g., a verifier V takes both x and an alleged proof π), the relevant measure of efficiency is polynomial in $|x|$, unless stated otherwise. This convention is standard in complexity theory and will be made explicit when it affects the analysis.

[4]One can also think of x as a *problem* and y as a *(valid) solution*, but this is somewhat confusing in the context of this discussion of decision/search problems.

[5]Formally, P need not even be a Turing machine; but in this thesis, it always will be.

Part I

Proofs

Probabilistic Proof Systems, Briefly

Besides the notation defined in the Introduction, each chapter in this dissertation is self-contained and can be read independently. That said, Proofs—as they are understood in modern complexity theory as Probabilistic Proof Systems—are the central technical component of this part of the dissertation. Therefore, let us take a moment to introduce them for the benefit of the unfamiliar reader. Our goal is to have the main two variants of such Proof Systems defined in a single section for ease of perusal, comparison and digestion; the reader is referred to Goldreich (2008a) for a full introduction, including foundational results.

In this dissertation, we adopt the common association between proofs and verification: when one talks about *proof*, it is always done so in the context of a *proof system*, which is itself prescribed by a *verifier*. In the words of Shimon Even: "A proof is whatever convinces me!" (Goldreich, 2008a); that is to say, the notion of a proof is always with respect to a predefined (at times implicit) verifier.

Proofs and verification have been central to computational complexity theory since its inception. Indeed, the notorious "P vs. NP" question asks whether any decision problem that can be efficiently verified can also be efficiently solved.[6]

In this brief exposition, we restrict attention to decision problems. Chapter 2, which focuses on search problems, introduces a refined version of interactive proofs adapted to that setting (see Definition 2.2). Since the focus is conceptual, we allow ourselves to be intentionally loose with formalism here (and here alone). Specifics such as the alphabet, soundness error, number of rounds, or other complexity measures are left undefined; each chapter will provide precise definitions tailored to its particular use case. The goal of this section is not to fix a universal framework, but to place the main two types of probabilistic proof systems side by side, allowing the reader to better appreciate their similarities and differences. Although Chapter 1 uses proof systems in a fundamentally different way from how Chapters 2 and 3 use them—but both are instances of the same overarching paradigm.

We begin with the classic notion of a proof system: one in which an efficient, deterministic verifier reads a given proof in its entirety and comes to a decision: "accept" if the proof convinced the verifier, and "reject" if it did not.

Definition (NP-proof system, informally)**.** *An efficient (deterministic) Turing machine V is an* NP-proof system *for (membership in) a set S if it satisfies the following two properties.*

[6] Or more verbosely, any decision problem for which there exists a proof system in which the verifier is efficient, admits an efficient decision algorithm.

- *Completeness: For any $x \in S$, there exists a* proof $\pi^* \in \{0,1\}^*$ *such $V(x,\pi)$ accepts.*[7]
- *Soundness: For any $x \notin S$ and any (alleged proof) string $\pi \in \{0,1\}^*$, $V(x,\pi)$ rejects.*

Next, in an Interactive Proof system, The verifier is randomized and engages in a multi-round dialogue with a prover. The verifier issues queries and receives answers (both are strings), and uses randomness to guide its queries. Because it is probabilistic, the verifier is permitted to err, but only with bounded probability. That is, for inputs not in the language, we require that no prover—regardless of strategy—can convince the verifier to accept except with low probability (here arbitrarily taken to be 50%). Informally, the designer of the proof system is expected to establish an upper bound on the likelihood of error in the form of a *soundness guarantee.*

Definition (Interactive Proof system (IP), informally)**.** *An efficient probabilistic oracle Turing machine V is an* Interactive Proof system (IP) *for (membership in) a set S if it satisfies the following two properties.*

- *Completeness: There exists an* honest prover P^* *such that for any $x \in S$,*

$$\Pr[\langle V, P^* \rangle (x) \text{ accepts }] = 1.$$

- *Soundness: For any $x \notin S$ and (alleged) prover P,*

$$\Pr[\langle V, P \rangle (x) \text{ accepts }] \leq 1/2.$$

Lastly, in a probabilistically checkable proof system, the verifier is still probabilistic, but but has access *restricted access* to a proof; the verifier must come to its decision while reading only few symbols from the proof.

Definition (Probabilistically Checkable Proof system (PCP), informally)**.** *An efficient probabilistic oracle Turing machine V is a* Probabilistically Checkable Proof system (PCP) *for (membership in) a set S if it satisfies the following two properties.*

- *Completeness: For any $x \in S$, there exists a proof oracle π^* such that*

$$\Pr[V^{\pi^*}(x) \text{ accepts }] = 1.$$

- *Soundness: For any $x \notin S$ and (alleged) proof oracle π,*

$$\Pr[V^{\pi}(x) \text{ accepts }] \leq 1/2.$$

The number of queries the verifier is allowed make to the proof oracle should be small; ideally a constant $q \in \mathbb{N}$ independent of the input length $|x|$. Additionally, the length of the proof oracle (viewed as a string) should be small as well; ideally, a nearly-linear function $\ell(n)$ where $n = |x|$ is the length of the input.

[7] Note that $|\pi|$ must be polynomial in $|x|$ by our definition of efficiency; see Introduction.

As the definitions make clear, both IPs and PCPs are forms of probabilistic proof systems: they feature efficient verifiers with access to randomness and allow a small probability of error. In an interactive proof (IP), the verifier engages in a dynamic exchange with a prover, who can adapt its responses to the verifier's queries. In a probabilistically checkable proof (PCP), by contrast, the proof is fixed in advance, and the verifier samples a small number of locations to inspect. Though historically connected, these two frameworks have given rise to different communities and research trajectories, with distinct technical concerns. Rather than surveying those literatures here, I offer this section as a conceptual and definitional reference—one that makes the structural similarities visible before each system is developed more formally in the chapters ahead.

Chapter 1

Some Hard Claims Have Complex Proofs

An $N \times N$ matrix with entries from a field $\mathcal{F}$ is said to be (Δ, ρ)-*rigid* if its Hamming distance from the set of $N \times N$ matrices of rank at most ρ is greater than Δ. In other words, an (Δ, ρ)-rigid matrix is a matrix that cannot be expressed as a sum of two matrices, $L + S$, where $\text{rank}(L) \leq \rho$ and S has at most Δ non-zero entries. For concreteness, this chapter focuses on rigidity with respect to the field $\mathcal{F}_2$.

Constructing rigid matrices has been a long-standing open problem in computational complexity theory since their introduction by Valiant (1977) more than four decades ago. Valiant showed that for any $(N^{1+\varepsilon}, N/\log\log(N))$-rigid matrix, evaluating the corresponding linear transformation requires circuits of either super-linear size or super-logarithmic depth. Thus, an explicit construction of a such matrices gives explicit problems that cannot be solved in linear-size logarithmic-depth circuits.

Razborov (1989) (see also Wunderlich 2012) considered the other end of the spectrum of parameters, in which the distance Δ is quite high but the rank ρ is much smaller; namely, $\Delta = \delta \cdot N^2$ for constant $\delta > 0$ and $\rho = 2^{(\log\log N)^{\omega(1)}}$. Razborov showed that strongly-explicit matrices[1] with these rigidity parameters imply a lower bound for the communication-complexity analog of the Polynomial Hierarchy, PH^{cc}.

In other words, while Valiant's regime focuses on very high rank ρ, Razborov's focuses on very high distance Δ. Achieving lower bounds for either PH^{cc} (via Razborov's reduction) or linear-size log-depth circuits (via Valiant's reduction) are two central long-standing open questions in complexity theory.

Despite a lot of effort, state of the art results on matrix rigidity fall short of solving both Valiant and Razborov's challenges. The current best poly-time constructions yields $(\frac{N^2}{\rho}\log(\frac{N}{\rho}), \rho)$-rigid matrices, for any parameter ρ (see Friedman 1993; Shokrollahi, Spielman, and Stemann 1997). Goldreich and Tal (2018) gave a randomized poly-time algorithm that uses $O(N)$ random bits and produces an $N \times N$ matrix that is $(\frac{N^3}{\rho^2 \log N}, \rho)$-rigid for any parameter $\rho \geq \sqrt{N}$, with high probability.[2] Recent breakthrough results (Alman and Williams,

[1] An $N \times N$ matrix A is strongly-explicit if, given $i, j \in N$, one can compute $A_{i,j}$ in $\mathsf{polylog}(N)$ time.

[2] Despite its "semi-explicitness", if this construction obtains Valiant's rigidity parameters, then lower

2017; Dvir and Edelman, 2019; Dvir and Liu, 2020) showed that several long-standing candidate construction of explicit matrices, like the Hadamard or the FFT matrices, are less rigid than previously believed, and in particular do not meet Valiant's challenge.

Most previous attempts were of combinatorial or algebraic nature (see surveys by Lokam 2009 and Ramya 2020). In contrast to these, a recent remarkable work of Alman and Chen (2022) proposes a novel approach that uses ideas from complexity theory to construct rigid matrices in FNP:[3]

Theorem 1.1 (Alman and Chen 2022)**.** *There exists a constant $\delta \in (0,1)$ such that for all $\varepsilon \in (0,1)$, there is an FNP-machine that, for infinitely many N, on input 1^N outputs an $N \times N$ matrix that is $(\delta \cdot N^2, 2^{(\log N)^{1/4-\varepsilon}})$-rigid.*[4]

Their result still does not attain the rank bounds required for Valiant's lower bounds, yet it vastly improves the state of the art of explicit rigid matrix constructions. On Razborov's end, the construction indeed meets the required rigidity parameters (in fact, greatly exceeds them), but does not fulfill the requirement of super-explicitness. That said, they use a tensoring argument to obtain $N \times N$ matrices still within Razborov's rigidity parameters, in which each entry is computable in non-deterministic time $2^{(\log\log N)^{\omega(1)}}$. While this is not super-explicit, it is an exponential improvement over previous results.

The surprising construction of Alman and Chen is a tour-de-force that ties together seemingly unrelated areas of complexity theory. A key area is the theory of Probabilistically Checkable Proofs (PCPs). PCPs provide a format of rewriting classical NP-proofs that can be efficiently verified based only on a small amount of random queries into the rewritten proof. The PCP Theorem (Arora and Safra, 1998; Arora et al., 1998) asserts that any NP-proof can be rewritten into a polynomially-longer PCP that can be verified using a constant number of queries. Alman and Chen make use of *efficient* and *short* PCPs for $\mathsf{NTIME}(2^n)$, as well as *smooth* PCPs. Momentarily, we too will make use of these properties, so let us give an informal description of these:

Efficient PCP: A PCP for $\mathsf{NTIME}(T(n))$ is said to be *efficient* if the running time of the PCP verifier is sub-linear (or even logarithmic) in the length of the original NP-proof (i.e., $T(n)$).

Short PCP: A PCP for $\mathsf{NTIME}(T(n))$ is said to be *short* if the length of the PCP is nearly linear (i.e., $T(N)^{1+o(1)}$) in the length of the original NP-proof (i.e., $T(n)$).

---

bounds would be implied, since one can take the randomness to be part of the input, yielding a (related) explicit problem that has no linear-size logarithmic-depth circuits.

[3]The complexity class FNP is the function-problem extension of the decision-problem class NP. Formally, a relation $R(x,y)$ is in FNP if there exists a non-deterministic polynomial-time Turing machine M such that for any input x, $M(x)$ outputs y such $R(x,y)=1$ or rejects if no such y exists.

[4]Their result is stated for $\mathsf{FP}^{\mathsf{NP}}$ but can be strengthened to FNP. More precisely, on infinitely many inputs 1^N that are accepted by the FNP-machine, any accepting path outputs a rigid matrix. Matrices obtained on different accepting paths may differ, but all of them are rigid. If one insists on outputting the same matrix on all accepting paths, then this can be done in $\mathsf{FP}^{\mathsf{NP}}$.

Smooth PCP: A PCP is said to be *smooth* if for each input, every proof location is equally likely to be queried by the PCP verifier.

The PCP Theorem has had a remarkable impact on our understanding of the hardness of approximation of several combinatorial problems (see, e.g., a survey Trevisan 2013). Parallel to this line of work, Babai et al. (1991) initiated a long sequence of works (Ben-Sasson et al., 2003; Ben-Sasson et al., 2006; Ben-Sasson and Sudan, 2008; Ben-Sasson et al., 2005; Dinur, 2007; Moshkovitz and Raz, 2008; Mie, 2009; Ben-Sasson and Viola, 2014; Paradise, 2021a) that prove that there exist efficient, short and smooth PCPs for $\mathsf{NTIME}(2^n)$. Alman and Chen's construction makes use of these PCPs, as well as the non-deterministic time-hierarchy theorem (Zák, 1983) and a fast (i.e., faster than $N^2/\mathsf{polylog}(N)$ time for $N \times N$ matrices) algorithm for counting the number of ones in a low-rank matrix (Chan and Williams, 2021).

Main result: Improved rigid matrices in FNP

Our work arises from asking if there exist PCPs with additional "nice" properties that can strengthen the above construction due to Alman and Chen (2022). We answer this question in the affirmative, by (1) introducing a new variant of PCPs that we refer to as *rectangular PCPs*, (2) constructing efficient, short and smooth rectangular PCPs and (3) using these rectangular PCPs to strengthen and simplify the rigid-matrix construction in Theorem 1.1. We begin by stating the improved rigid matrix construction.

Theorem 1.2. *There is a constant $\delta \in (0,1)$ such that there is an FNP-machine that for infinitely many N, on input 1^N outputs an $N \times N$ matrix that is $(\delta \cdot N^2, 2^{\log N/\Omega(\log\log N)})$-rigid.*

We remark that Alman and Chen obtained a conditional result which proved a similar conclusion using the easy witness lemma of Impagliazzo, Kabanets, and Wigderson (2002): either $\mathsf{NQP} \not\subset \mathcal{P}/\mathsf{poly}$ or for all $\varepsilon \in (0,1)$ there exists an FNP-algorithm that for infinitely many N, on input 1^N outputs an $N \times N$ matrix that is $(\delta \cdot N^2, 2^{(\log N)^{1-\varepsilon}})$-rigid. Our main result (Theorem 1.2) is thus a common strengthening of both Theorem 1.1 as well as this conditional result.

Rectangular PCPs

Our result is obtained by a new notion of PCPs, called *rectangular PCPs*.[5] Briefly put, rectangular PCPs are PCPs where the proofs are thought of as square matrices, and the random coins used by the verifier can be partitioned into two disjoint sets, one determining the *row* of each query and the other determining the *column*. To get a better feel for this new property, we examine the constraint satisfaction problem (CSP) underlying a rectangular PCP,[6] and defer the full definition to Section 1.1.

[5] We thank Ramprasad (RP) Saptharishi for suggesting the term "rectangular PCPs".

[6] See, for example, Chapter 18 of Arora and Barak 2009 for a description of the CSP underlying a PCP.

Consider the classical NP-hard constraint satisfaction problem MAXCUT whose instance is a directed graph $G = (V, E)$ with n vertices and m edges and the goal is to find a subset $S \subseteq V$ that maximizes the number of edges cut (in either direction) between S and $V \setminus S$. The instance G is *rectangular* if the following condition is met.

- There exist two directed graphs G_1 and G_2 with $\ell = \sqrt{n}$ vertices and $r = \sqrt{m}$ edges each such that G is the product graph $G_1 \times G_2$, i.e., the edges of G satisfy the following *rectangular* property:

$$((u_1, u_2), (v_1, v_2)) \in E(G) \iff (u_1, v_1) \in E(G_1) \text{ and } (u_2, v_2) \in E(G_2).$$

An instance G is said to be *τ-almost rectangular* for $\tau \in [0, 1)$ if G is the edge-disjoint union of m^τ product graphs $G_1^{(j)} \times G_2^{(j)}, j \in [m^\tau]$ where each of the product graphs $G_1^{(j)} \times G_2^{(j)}$ is defined on the same vertex set V and satisfies $|E(G_1^{(j)})| = |E(G_2^{(j)}| = m^{(1-\tau)/2}$. To distinguish rectangular graphs from almost-rectangular graphs, we will sometimes refer to them as perfectly rectangular.

This definition of rectangularity can be extended to arbitrary q-CSPs as follows. Let Φ be a q-CSP instance on a set V of $n = \ell^2$ variables. Let $\mathcal{C}$ be the set of $m = r^2$ constraints of Φ. As both the number of variables ($n = \ell^2$) and the number of constraints ($m = r^2$) are perfect squares, we will w.l.o.g. index them with double indices, $(i_i, i_2) \in [\ell] \times [\ell]$ and $(j_1, j_2) \in [r] \times [r]$. Let the (j_1, j_2)-th constraint in $\mathcal{C}$ involve the q variables $x_{c_1(j_1,j_2)}, \ldots, x_{c_q(j_1,j_2)}$. The instance Φ is said to be rectangular if for any $k \in [q]$, the address function $c_k : [r] \times [r] \to [\ell] \times [\ell]$ that specifies the k-th variable in the (j_1, j_2)-th clause can be decomposed into a product function $a_k \times b_k$ where $a_k, b_k \colon [r] \to [\ell]$. Almost rectangularity is defined similarly.

Back in the "proof systems" view, a PCP is said to be *(τ-almost) rectangular* if its underlying CSP is (τ-almost) rectangular. Thus, rectangularity can be viewed as natural structural property referring to the clause-variable relationship in the CSPs produced by the PCP. Our main technical result is that there exists an efficient, short, smooth and almost-rectangular PCP. A simplified version of our result is as follows (see Theorem 1.49 for the exact statement.)

Theorem 1.3. *Let L be a language in* NTIME(2^n). *For every constants $s \in (0, 1/2)$ and $\tau \in (0, 1)$, there exists a constant-query, smooth and τ-almost rectangular PCP for L over the Boolean alphabet with perfect completeness, soundness error s, proof length at most $2^n \cdot \mathsf{poly}(n)$ and verifier running time at most $2^{O(\tau n)}$.*

From rectangular PCPs to rigid matrices

We now sketch how rectangular PCPs can be used to construct rigid matrices. This will also serve as a motivation for the definition of rectangular PCPs. Our construction follows that of Alman and Chen (which fits within the *lower bounds from algorithms* framework of

Williams (2013) – see Appendix A.1), with the main difference being the use of rectangular PCPs. We show that this simplifies their construction and improves the rigidity parameters it attains. The construction is inspired by the maxim:

Hard claims have complex proofs.

Informally speaking, the construction is an instantiation of this maxim where "complexity" refers to rigidity, "proofs" are PCPs, and "hard claims" are instances of the hard language guaranteed by the non-deterministic time hierarchy theorem (see next).

The main ingredients in our construction are as follows:

1. The non-deterministic time-hierarchy theorem (Zák, 1983): There exists a unary language $L \in \mathsf{NTIME}(2^n) \setminus \mathsf{NTIME}(2^n/n)$.

2. A non-trivial (i.e., sub-quadratic time) algorithm to compute the number of ones in a low-rank $\{0,1\}$-valued matrix when given as input its low-rank decomposition $N = P \cdot Q$, where P and Q are matrices of dimensions $N \times \rho$ and $\rho \times N$, respectively. Such results with running time $N^{2-\varepsilon(\rho)}$ were developed by Chan and Williams (2021) with $\varepsilon(\rho) = \Omega(1/\log \rho)$.

3. The existence of efficient, short, smooth and rectangular PCPs for $\mathsf{NTIME}(2^n)$, as guaranteed by Theorem 1.3.

Let $L \in \mathsf{NTIME}(2^n) \setminus \mathsf{NTIME}(2^n/n)$ be as guaranteed by the non-deterministic time-hierarchy theorem. By Theorem 1.3, there exist efficient, short and smooth rectangular PCPs for L. Our goal is to show that either (a natural transformation of) the PCP yields a rigid matrix, or $L \in \mathsf{NTIME}(2^n/n)$ – a contradiction.

For simplicity of presentation, we will assume that the rectangular PCPs obtained in Theorem 1.3 are perfectly rectangular and furthermore that the underlying CSP of the PCP is MAXCUT with completeness c and soundness s for some constants $0 < s < c < 1$. In other words, the PCP reduction reduces instances $x \in L$ to digraphs G which have a fractional cut of size at least c, and instances $x \notin L$ to digraphs G which do not have any cut of fractional size larger than s.

Let us understand what it means for the PCP to be short, smooth, efficient and rectangular: *"Rectangular"* refers to the fact that the digraph G is a product graph $G_1 \times G_2$; *"Short"* implies that the size of G (i.e., the number of vertices and edges) is at most $r^2 = 2^n \cdot \mathsf{poly}(n)$; *"Smooth"* implies that the digraph G is regular (the degree of a vertex is the sum of its in-degree and out-degree); and *"Efficient"* implies that for each of the two graphs G_1 and G_2, given an edge the vertices incident on the edge can be obtained in time $2^{\gamma n}$ (for a small constant $\gamma > 0$ of our choice).

For any instance x of the language L, any cut of the corresponding graph G is of the form $(S, V(G) \setminus S)$. Since $V(G) = V(G_1) \times V(G_2)$, we can identify the cut S with a $V(G_1) \times V(G_2)$-matrix with $\{0,1\}$ entries. Let $L_1, R_1 \in \{0,1\}^{E(G_1) \times V(G_1)}$ be the incidence matrices indicating the left and right endpoints of the edges in G_1 (i.e., if $e = (u, v) \in E(G_1)$

then $L_1(e,u) = R_1(e,v) = 1$). Similarly, define matrices $L_2, R_2 \in \{0,1\}^{E(G_2)\times V(G_2)}$. These matrices will not be computed explicitly; efficiency of the PCP implies that for any given row-index, the non-zero column of that row can be computed in time $2^{\gamma n}$. We will refer to this fact as the *somewhat-efficient* computation of these matrices.

Observe that the matrix $L_1 \cdot S \cdot L_2^T$ is an indicator matrix indicating if the left endpoint of the edge is in the set S or not. Similarly $R_1 \cdot S \cdot R_2^T$ refers to the indicator of the right endpoint of the edge. Hence, the matrix $M(S) := L_1 \cdot S \cdot L_2^T + R_1 \cdot S \cdot R_2^T$ is the indicator matrix of whether the edge is cut by the set S or not (with addition over $\mathcal{F}_2$). Thus, the size of the cut induced by the set S is exactly the number of ones in the matrix $M(S)$. Let us denote this quantity by $\mathrm{val}(S) := \#_1(M(S))$.

We will prove that for infinitely many $x \in L$, every cut $S^* \in \{0,1\}^{V(G_1)\times V(G_2)}$ of fractional size at least c is a $(\delta \cdot N^2, \rho)$-rigid matrix, for $\delta = (c-s)/3$ and $\rho = 2^{n/\Omega(\log n)}$. Assume towards contradiction that this was not the case. Then, for every long enough $x \in L$, there exists a cut S^* th at is non-rigid. Since S^* is non-rigid, it is δ-close to some Boolean matrix $S = P \cdot Q$ such that P and Q are Boolean matrices of dimensions $V(G_1) \times \rho$ and $\rho \times V(G_2)$, respectively. We now make two observations.

- Since the cut S^* is of size at least c and is δ-close to S, it follows from the regularity of G that the cut induced by the set S is of size at least $c - 2\delta$. In other words, $\mathrm{val}(S) \geq c - 2\delta$.
- We can compute $\mathrm{val}(S) = \#_1(M(S))$ in time $O\left(r \cdot (\rho + 2^{\gamma n}) + r^{2-\varepsilon(2\rho)}\right)$ as follows. Recall that $M(S) = L_1 \cdot S \cdot L_2^T + R_1 \cdot S \cdot R_2^T$ and $S = P \cdot Q$. Hence,

$$M(S) = L_1 \cdot P \cdot Q \cdot L_2^T + R_1 \cdot P \cdot Q \cdot R_2^T = \underbrace{\begin{pmatrix} L_1 & R_1 \end{pmatrix} \cdot \begin{pmatrix} P & 0 \\ 0 & P \end{pmatrix}}_{=:\widetilde{P}} \cdot \underbrace{\begin{pmatrix} Q & 0 \\ 0 & Q \end{pmatrix} \cdot \begin{pmatrix} L_2^T \\ R_2^T \end{pmatrix}}_{=:\widetilde{Q}}$$

  So $M(S)$ is a matrix of rank at most 2ρ with a low-rank decomposition given by $M(S) = \widetilde{P} \cdot \widetilde{Q}$. Given matrices P and Q and the somewhat-efficient computation of the matrices L_1, L_2, R_1, R_2, the matrices $\widetilde{P}$ and $\widetilde{Q}$ may be computed in time $r \cdot (\rho + 2^{\gamma n})$. Finally, we invoke the algorithm of Chan and Williams to compute $\mathrm{val}(S) = \#_1(\widetilde{P} \cdot \widetilde{Q})$ in time $O\left(r^{2-\varepsilon(2\rho)}\right)$.

This suggests the following non-deterministic algorithm for checking membership in the unary language L:

- On input 1^n:
  1. Non-deterministically guess matrices $P \in \{0,1\}^{\ell \times \rho}$ and $Q \in \{0,1\}^{\rho \times \ell}$.
  2. Use the efficient PCP verifier to somewhat-efficiently compute the matrices L_1, L_2, R_1, and R_2.

3. Compute the matrices $\widetilde{P}$ and $\widetilde{Q}$.
4. Compute the number of ones ν of the matrix $\widetilde{P} \cdot \widetilde{Q}$.
5. Accept if and only if $\nu > s \cdot r^2$.

Indeed, this algorithm decides L: If $1^n \in L$, then there exists a guess $S = P \cdot Q$ that would get $\mathrm{val}(S) \geq (c - 2\delta) \cdot r^2 > s \cdot r^2$. On the other hand, if $1^n \notin L$, then by the soundness of the PCP, any cut S would have $\mathrm{val}(S) \leq s \cdot r^2$.

However, for a suitable choice of ρ, this algorithm runs in time

$$O\left(\ell \cdot \rho + r \cdot (\rho + 2^{\gamma n}) + r^{2-\varepsilon(2\rho)}\right) = O(2^n/n),$$

thereby contradicting the time-hierarchy theorem. Hence, it is false that for every long enough $x \in L$, there exists a cut $S^* \in \{0,1\}^{V(G_1)\times V(G_2)}$ of fractional size at least c which is a non-rigid matrix. This immediately yields an FNP-algorithm that infinitely often outputs rigid matrices.

In Section 1.2 we complete this sketch into a full proof that deals with two significant caveats: the PCP is only almost-rectangular, and the predicate of the PCP is not necessarily MAXCUT. The first is not a significant obstacle and the generalization is rather immediate. The second requires more care, but examining the proof reveals that we only used the fact that each clause has the same predicate or, in PCP jargon, that the predicate is *oblivious to the randomness*. To this end, we define a property of PCPs termed randomness-oblivious predicates (ROP) and show that the rectangular PCPs constructed in Theorem 1.3 can also be made ROP (see Sections 1.5 and 1.7 for exact details).

Comparison with the Alman–Chen construction: Alman and Chen obtained a similar result conditioned on the assumption $\mathsf{NQP} \subseteq \mathcal{P}/\mathsf{poly}$, using the easy witness lemma. To obtain an unconditional result, they used a bootstrapping argument which results in rigidity for rank at most $2^{(\log N)^{1/4-\varepsilon}}$. The above proof, on the other hand, is not conditioned on any assumption, does not require the easy witness lemma, and implies rigid matrices for rank $2^{\log N/\Omega(\log\log N)}$. In fact, there is almost no loss due to the PCPs in the above argument. For instance, if the number of ones in $N \times N$ matrices of rank $N^{0.999}$ could be computed in sub-quadratic time, then our construction would yield matrices rigid for rank $N^{0.99}$.

Constructing rectangular PCPs

We now show how one constructs rectangular PCPs. Recall that the rectangular property of PCPs states that the underlying constraint satisfaction problem (CSP) has a product structure.

Warm-up: Rectangularity of some known constructions

As a warm-up, let us examine the rectangularity of some common PCP building blocks. The purpose of this warm-up is to become comfortable with the notion of rectangularity. Towards

this end, we chose some simple examples from the PCP literature, rather than examples that are actually used in our construction.

First, it is immediate that PCPs obtained from *parallel repetition* are rectangular. Unfortunately, the size of these PCPs are far from being nearly-linear. Next, recall *BLR linearity tester* of Blum, Luby, and Rubinfeld (1993) that checks if a given function $f : \mathcal{F}_2^m \to \mathcal{F}_2$ is linear.

Algorithm (BLR Tester)**.** On oracle access to $f \colon \mathcal{F}_2^m \to \mathcal{F}_2$,

1. Sample $x, y \in_R \mathcal{F}_2^m$.
2. Query f at locations x, y, and $x + y$.
3. Accept if and only if $f(x) + f(y) + f(x + y) = 0$.

The (x, y)-th constraint in the above test queries the three locations $x, y, x + y \in \mathcal{F}_2^{3m}$. For even m, we can write $x = (x_1, x_2)$ and $y = (y_1, y_2)$ where $x_1, x_2, y_1, y_2 \in \mathcal{F}_2^{m/2}$. Thus, the $((x_1, x_2), (y_1, y_2))$-test queries the three locations (x_1, x_2), (y_1, y_2) and (x_1+y_1, x_2+y_2). Hence, the BLR test is perfectly rectangular. For similar reasons, the low-degree test (actually used in our construction) is also perfectly rectangular.

The actual construct

The warm-up gives us hope that PCPs constructed using from low-degree test are rectangular or can be made so with some modification. Our construction, essentially, realizes this hope. In particular we take a closer look at the short and efficient PCP construction of Ben-Sasson et al. (2006) and Ben-Sasson et al. (2005) and modify it suitably to obtain a rectangular PCP. This is a rather delicate operation and involves several subtleties along the way. We highlight the salient steps in the construction below.

For starters, recall another key ingredient in the construction of PCPs: the composition paradigm of Arora and Safra (1998). We will use the modular composition paradigm of Ben-Sasson et al. (2006) and Dinur and Reingold (2006), wherein a *robust* PCP is composed with a PCP *of proximity*. Our construction of rectangular PCPs will proceed along the following lines.

1. We first show that the Reed–Muller based PCP construction due to Ben-Sasson et al. (2006) and Ben-Sasson et al. (2005) can be modified to yield a short almost-rectangular robust PCP. This involves a careful, step-by-step examination of this PCP. As indicated above, the low-degree component of this PCP is perfectly rectangular. However, this PCP also involves a sum-check component, which is inherently *not* rectangular, but is fortunately *almost* rectangular.
2. We then show that composition of an almost-rectangular robust PCP with a (not necessarily rectangular) PCP of proximity yields an almost-rectangular PCP.

Composing the outer robust PCP obtained in Item 1 with the short and efficient PCP of proximity of Mie (2009) yields a short, efficient and rectangular PCP with constant query complexity. However, this PCP is not necessarily smooth. By now, there are several standard techniques to "smoothify" a PCP in literature, but these techniques do not necessarily retain the rectangular property. To obtain a rectangular and smooth PCP, we actually work with a stronger notion of rectangularity, that we refer to as "rectangular neighborhood-listing (RNL)" and show that a short and efficient PCP with RNL can be "smoothified" to yield the desired short, efficient, smooth and rectangular PCP.

Organization

The rest of the chapter is organized as follows.

Preliminaries (Section 1.1). We begin by giving a definitional treatment of PCPs and their variants. In particular, we formally define the rectangular PCP, which is the central object of our focus. We also define the two aforementioned related properties: rectangular neighborhood-listing (RNL) and randomness-oblivious predicates (ROP).

From rectangular PCPs to Rigid Matrices (Section 1.2). We show how the existence of efficient, short and smooth rectangular PCPs with ROP for $\mathsf{NTIME}(2^n)$ yields rigid matrices (thus proving Theorem 1.2, modulo the actual rectangular PCP construction).

A Construction of Rectangular PCPs (Sections 1.3 to 1.7). In the remaining sections of the chapter, we construct efficient, short and smooth rectangular PCPs for $\mathsf{NTIME}(T(n))$. The main steps in the construction are as follows:

- Section 1.3: We show how any PCP with RNL and ROP can be converted to a smooth and rectangular PCP with ROP. Hence, from this point onwards, we seek PCPs with RNL, rather than rectangular PCPs.
- Section 1.4: We show that the robust PCP verifier of Ben-Sasson et al. (2006) and Ben-Sasson et al. (2005) has RNL.
- Section 1.5: We show how to add ROP to any robust PCP with RNL with $q \geq r$ (i.e., whose query complexity is larger than its randomness complexity).
- Section 1.6: We show that any PCP of proximity, when composed with a robust PCP that has RNL and ROP, yields a PCP with RNL and ROP. Note that the PCP of proximity need not be rectangular, and, consequently, the composite PCP is "less rectangular" than the robust PCP.
- Section 1.7: Finally, we combine the results proved in Sections 1.3 to 1.6 to obtain our main construct: an efficient, short and smooth rectangular PCP (thus proving Theorem 1.3).

1.1 Defining rectangular PCPs

The main focus of this section is to introduce the notion of *rectangular PCPs*, the central object of interest in this chapter. To this end, we begin by recalling the standard definition of PCPs and related objects (PCP verifier, robust soundness, smooth PCPs) before proceeding to define rectangular PCPs.

Notation. Let Σ be any finite alphabet. For $u, v \in \Sigma^n$, the relative Hamming distance between u and v, denoted by $\delta(u, v)$, is the fraction of locations on which u and v differ (i.e., $\delta(u, v) \coloneqq |\{i : u_i \neq v_i\}|/n$). We say that u is δ-*close* to v (resp., δ-*far* from v) if $\delta(u, v) \leq \delta$ (resp., $\delta(u, v) > \delta$). The relative distance of a string u to a set V of strings is defined as $\delta(u, V) \coloneqq \min_{v \in V}\{\delta(u, v)\}$.

Standard PCPs

We begin by recalling the formalism of a PCP verifier. As is standard in this literature, we restrict our attention to *non-adaptive* verifiers.

Definition 1.4 (PCP verifiers)**.**

- *Let $r, q, m, d, t, \sigma \colon \mathbb{N} \to \mathbb{N}$. A (r, q, m, d, t)-restricted verifier over alphabet $\Sigma \coloneqq \{0,1\}^\sigma$ is a probabilistic algorithm[7] V that, on an input x of length n, tosses $r \coloneqq r(n)$ random coins R and generates a sequence of $q \coloneqq q(n)$ query locations $I \coloneqq (i^{(1)}, \ldots, i^{(q)})$, where each $i^{(k)} \in [m(n)]$, and a (decision) predicate $D \colon \Sigma^q \to \{0,1\}$ in time at most $t(n)$. The decision predicate D is specified by a circuit of size at most $d(n)$.*

  Think of V as representing a probabilistic oracle machine that queries the proof oracle $\pi \in \Sigma^m$, for the positions in I, receives the q symbols $\pi|_I \coloneqq (\pi_{i^{(1)}}, \ldots, \pi_{i^{(q)}})$, and accepts iff $D(\pi|_I) = 1$.

- *We write $(I, D) \overset{R}{\sim} V(x)$ to denote the queries and predicate generated by V on input x and random coin tosses. To explicitly mention the random coins R, we write $(I, D) \sim V(x; R)$.*

- *We call r the randomness complexity, q the query complexity, m the proof length, d the decision complexity and t the running time of V. The length σ of symbols in Σ is called the* answer complexity *of V, and will usually be omitted.*[8]

It will be convenient at times to have the following graphical description of the verifier. Given a (r, q, m, d, t)-restricted verifier and input x, consider the bipartite graph $G(V, x) \coloneqq$

[7] In this chapter, algorithm refers to a multi-tape Turing machine.

[8] All PCPs in this chapter will be Boolean (i.e., $\sigma = 1$), except for an intermediate PCP in Section 1.4. Even there, it will be more convenient to consider the alphabet size $|\Sigma| = 2^\sigma$ rather than σ.

$(L = \{0,1\}^r, R = [m], E)$ where $(R, i) \in E$ if the verifier V on input x and random coins R queries location i in the proof. Clearly, the graph $G(V, x)$ is q-left regular.

We can now define the standard notion of PCPs with perfect completeness.

Definition 1.5 (PCP)**.** *For a function $s : \mathbb{N} \to [0,1]$, a verifier V is a probabilistically checkable proof system (PCP) for a language L with soundness error s if the following two conditions hold for every string x:*

Completeness: *If $x \in L$ then there exists π such that $V(x)$ accepts oracle π with probability 1. Formally,*

$$\exists \pi \qquad \Pr_{(I,D) \overset{R}{\sim} V(x)} [D(\pi|_I) = 1] = 1.$$

Soundness: *If $x \notin L$ then for every oracle π, the verifier $V(x)$ accepts π with probability strictly less than s. Formally,*

$$\forall \pi \qquad \Pr_{(I,D) \overset{R}{\sim} V(x)} [D(\pi|_I) = 1] < s(|x|).$$

While constructing PCPs, we will sometimes be interested in PCPs with a stronger notion of soundness, referred to as *robust soundness*.

Definition 1.6 (robust soundness)**.** *For functions $s, \rho : \mathbb{N} \to [0,1]$, a PCP verifier V for a language L has robust-soundness error s with robustness parameter ρ if the following holds for every $x \notin L$: For every oracle π, with probability strictly less than s, the symbols read by the verifier V are ρ-close to being accepted. Formally,*

$$\forall \pi \quad \Pr_{(I,D) \overset{R}{\sim} V(x)} [\exists a \text{ s.t. } D(a) = 1 \text{ and } \delta(a, \pi|_I) \leq \rho] < s(|x|).$$

By now, we know of several such efficient PCP constructions, one of which we state below.

Theorem 1.7 (efficient PCPs for $\mathsf{NTIME}(T)$, Theorem 2.6 of Ben-Sasson et al. 2005)**.** *Suppose that L is a language in $\mathsf{NTIME}(T(n))$ for some non-decreasing function $T \colon \mathbb{N} \to \mathbb{N}$. Then for every $\varepsilon \in (0,1)$, L has a PCP verifier over $\{0,1\}$ with soundness error ε, query complexity $O(1/\varepsilon)$ and randomness complexity $\log T(n) + \log^{O(\varepsilon)} T(n)$.*

While constructing variants of the above PCP, we will particularly be interested in smooth PCPs.

Definition 1.8 (smooth PCP)**.** *Given a (r, q, m, d, t)-restricted verifier V, an input x and $i \in [m]$, let $Q_x(i)$ denote the probability with which the verifier V outputs i on a random query $k \in [q]$. Formally,*

$$Q_x(i) \coloneqq \Pr_{R, k \in [q]} [i^{(k)} = i | (I, D) \sim V(x; R)].$$

The PCP verifier V is said to be smooth if for all $i, j \in [m]$, $Q_x(i) = Q_x(j)$.

Thus, smooth PCPs refer to PCPs whose verifiers query all locations of the proof oracle equally likely (or equivalently in the above graphical description, verifiers whose corresponding bipartite graphs are also right-regular).[9]

Remark 1.9 (tolerance of smooth PCPs)**.** *A smooth PCP is* tolerant *of errors in a correct proof, in the sense that a proof that is close to a correct one is accepted with good probability. Concretely, suppose V makes q queries to its proof and is smooth. Then if π is a correct proof for V (i.e., accepted w.p. 1) and π^* is δ-close to π in relative Hamming distance, then π^* is accepted with probability at least $1 - q \cdot \delta$.*

The PCPs constructed in Theorem 1.7 are not necessarily smooth, however they can be made smooth without too much of an overhead. In this chapter we will be interested in *smoothening* the PCP maintaining yet another property, *rectangularity*, which we introduce in the following section.

Rectangular PCPs

We now define *rectangular PCPs*, the central object of interest in this chapter. As the name suggests, rectangular PCPs are PCPs in which the proof oracle $\pi : [m] \to \Sigma$, an m-length string, is interpreted as a matrix $\pi : [\ell] \times [\ell] \to \Sigma$ for some ℓ such that $m = \ell^2$ (yes, we assume that the proof lengths are always squares of integers). Furthermore, the verifier is also "rectangular" in the sense that the randomness $R \in \{0,1\}^r$ is also partitioned into 2 parts $R = (R_{\text{row}}, R_{\text{col}})$ such that the row index of the queries is obtained from the "row randomness" R_{row} while the column index of the queries is obtained from the "column randomness" R_{col}.

The above informal description assumes "perfect" rectangularity while the definition below allows for the relaxed notion of "almost-rectangularity", in which randomness is partitioned into three parts: row and column (as above), as well as a small *shared* part that is used for obtaining both the rows and the columns of the queries.

Definition 1.10 (Rectangular PCP)**.** *For $\tau \in [0,1)$, a (r, q, ℓ^2, d, t)-restricted verifier V is said to be τ-*rectangular *if there exist probabilistic algorithms V_{row}, V_{col} such that the following holds.*

The random coin tosses $R \in \{0,1\}^r$ can be partitioned into 3 parts

$$R = (R_{\text{row}}, R_{\text{col}}, R_{\text{shared}}) \in \{0,1\}^{(1-\tau)r/2} \times \{0,1\}^{(1-\tau)r/2} \times \{0,1\}^{\tau r},$$

[9] *Minor historical inconsistencies in the definition of smoothness:* Several previous works (Katz and Trevisan, 2000; Paradise, 2021a) defined a smooth oracle machine as one in which each location of the oracle has equal probability of being queried by the machine *in any of its queries* (rather than in a random query, as in Definition 1.8 as well as other prior works (Goldreich and Sudan, 2006; Ben-Sasson et al., 2006)). Indeed, both definitions are equivalent assuming the machine never queries the same location twice for any given random coin sequence R. Our definition is more convenient as it coincides with right-regularity of the corresponding bipartite graph even without this assumption.

such that the verifier V on input x of length n and random coins R produces a sequence of q proof locations $I = ((i^{(1)}_{\text{row}}, i^{(1)}_{\text{col}}), \ldots, (i^{(q)}_{\text{row}}, i^{(q)}_{\text{col}}))$ as follows:

- $I_{\text{row}} \coloneqq (i^{(1)}_{\text{row}}, \ldots, i^{(q)}_{\text{row}}) = V_{\text{row}}(x; R_{\text{row}}, R_{\text{shared}})$,
- $I_{\text{col}} \coloneqq (i^{(1)}_{\text{col}}, \ldots, i^{(q)}_{\text{col}}) = V_{\text{col}}(x; R_{\text{col}}, R_{\text{shared}})$,
- *Generating I_{row}, I_{col}, and the decision predicate*[10] *take a total of at most $t(n)$ time.*

In other words, the row (respectively column) indices of the queries are only a function of the row (respectively column) and shared parts of the randomness. If $\tau = 0$, we will say the verifier V is perfectly rectangular, and otherwise V is almost rectangular. When it is obvious from context, we will say that V is simply rectangular, omitting the "τ-" qualifier.

Rectangular Neighbor-Listing (RNL)

A careful reading of the construction of PCPs mentioned in Theorem 1.7 will reveal that they are in fact rectangular. However, for our application, we will need rectangular PCPs that are also smooth. Later, we will "smoothen" a PCP while maintaining its rectangularity (see Section 1.3), for which we need a stronger property that we refer to as *rectangular neighbor-listing (RNL)*. To define this property, we first define *configurations* and *neighboring configurations*.

Definition 1.11 (configurations and neighboring configurations)**.** *Given a (r, q, m, d, t)-restricted verifier V and an input x, a configuration refers to a tuple $(R, k) \in \{0,1\}^r \times [q]$ composed of the randomness of the verifier and query index. The verifier V describes how to obtain the query location $i^{(k)} \in [m]$ from the configuration (R, k) (and the input x).*

We say that two configurations (R, k) and (R', k') of a PCP verifier V on input x are neighbors *if they both yield the same query location $i \in [m]$. (In particular, every configuration is a neighbor of itself.)*

In the graphical representation of a verifier, a configuration refers to an edge of the bipartite graph and two configurations are said to be neighbors if they are incident on the same right vertex. A configuration $(R, k) = (R_{\text{row}}, R_{\text{col}}, R_{\text{shared}}, k)$ of a rectangular PCP can be broken down into a row configuration $(R_{\text{row}}, R_{\text{shared}}, k)$ and a column configuration $(R_{\text{col}}, R_{\text{shared}}, k)$. Rectangularity states that the query location $(i^{(k)}_{\text{row}}, i^{(k)}_{\text{col}}) \in [\ell] \times [\ell]$ satisfy that $i^{(k)}_{\text{row}}$ is a function of the row configuration while $i^{(k)}_{\text{col}}$ is a function of the column configuration.

Definition 1.12 (rectangular neighbor-listing (RNL))**.** *For $\tau \in [0, 1)$ and $t_{\text{RNL}} : \mathbb{N} \to \mathbb{N}$, an (r, q, m, d, t)-restricted verifier V is said to have the τ-rectangular neighbor listing property (τ-RNL) with time $t_{\text{RNL}}(n)$ if the following holds.*

[10] It is natural to wonder how the decision predicate depends on the randomness. This is considered in Section 1.1.

- *The random coin tosses $R \in \{0,1\}^r$ can be partitioned into 4 parts*

  $$R = (R_{\text{row}}, R_{\text{col}}, R_{\text{shared.row}}, R_{\text{shared.col}}) \in \{0,1\}^{(1-\tau)r/2} \times \{0,1\}^{(1-\tau)r/2} \times \{0,1\}^{\tau r/2} \times \{0,1\}^{\tau r/2},$$

  where we refer to the 4 parts $R_{\text{row}}, R_{\text{col}}, R_{\text{shared.row}}, R_{\text{shared.col}}$ as the row part*,* column part*,* row-shared part *and* column-shared part *respectively. We will refer to the combined shared randomness $R_{\text{shared}} \coloneqq (R_{\text{shared.row}}, R_{\text{shared.col}})$ as the* shared part*.*

- *There exist two algorithms, a* row agent *(denoted A_{row}) and a* column agent *(denoted A_{col}) that list, in time $t_{\text{RNL}}(n)$, all neighbors of a given configuration (R, k) in the following "rectangular and synchronized" fashion:*

  - *On input a row configuration $(R_{\text{row}}, R_{\text{shared}}, k)$, the row agent A_{row} outputs a list L_{row} of tuples $(R'_{\text{row}}, R'_{\text{shared.row}}, k')$.*
  - *On input a column configuration $(R_{\text{col}}, R_{\text{shared}}, k)$, the column agent A_{col} outputs a list L_{col} of tuples $(R'_{\text{col}}, R'_{\text{shared.col}}, k')$.*

  satisfying the following properties

  1. *The two lists L_{row} and L_{col} are of equal length and entrywise-matching k' values, such that the "zipped" list*

     $$L \coloneqq \left\{ (R'_{\text{row}}, R'_{\text{col}}, R'_{\text{shared.row}}, R'_{\text{shared.col}}, k') \;\middle|\; \begin{array}{c} i \in [|L_{\text{row}}|] \\ (R'_{\text{row}}, R'_{\text{shared.row}}, k') \coloneqq L_{\text{row}}[i] \\ (R'_{\text{col}}, R'_{\text{shared.col}}, k') \coloneqq L_{\text{col}}[i] \end{array} \right\} \tag{1.1}$$

     is the list of all full configurations that are neighbors of (R, k).
  2. *Not only are the contents of L the same for each two neighboring locations, but the* order *of configurations in L is the same too. That is, for any two neighboring configurations (R, k) and $\left(\widetilde{R}, \widetilde{k}\right)$, the resulting configuration lists L and $\widetilde{L}$ are equal as ordered lists (element-by-element).*
  3. *Both agents output the index of (R, k) in the list L (despite not "knowing" (R, k) entirely).*

Informally speaking, rectangularity asserts that the query location can be obtained in a "rectangular" fashion from the randomness, while RNL asserts that the entire list of neighboring configurations of the query location can be obtained in a "rectangular" fashion.

A PCP with RNL can be made *smooth* and *rectangular*, as shown in Section 1.3.

Remark 1.13. *Barak and Goldreich (2008) defined PCPs with a* reverse-sampling *procedure that outputs a uniformly random neighbor of any given configuration. The important difference between RNL and reverse-sampling is that the former offers a procedure that outputs neighboring configurations in a rectangular fashion.*

Randomness-oblivious predicates (ROP)

For our application of rectangular PCPs (Section 1.2), we would like the decision circuit to depend only on the shared part of the randomness. However, we do not know how to obtain such a PCP (that is also *smooth* and *short*), so we allow the decision circuit to take a limited number of *parity checks* of the entire randomness. Like the decision circuit, the choice of parity checks depends only on the shared part of the randomness.

Definition 1.14 (efficient PCP verifiers with τ-ROP)**.** *For $\tau \in [0,1)$, a (r,q,m,d,t)-restricted verifier V is said to have the τ-randomness-oblivious predicates (τ-ROP) if the following holds.*

The random coin tosses $R \in \{0,1\}^r$ can be partitioned into two parts

$$R = (R_{\text{obliv}}, R_{\text{aware}}) \in \{0,1\}^{(1-\tau)r} \times \{0,1\}^{\tau r},$$

such that the verifier V on input x of length n and random coins R runs in time $t(n)$, and

1. *Based only on R_{aware}:*
   a) *Constructs a (decision) predicate $D \leftarrow V(x; R_{\text{aware}})$ of size at most $d(n)$.*
   b) *Constructs a sequence of randomness parity checks $(C_1, \ldots, C_p) \sim V(x; R_{\text{aware}})$, each of which is an affine function from $\{0,1\}^{(1-\tau)r}$ to $\{0,1\}$.*[11]
2. *Based on all of the randomness $R = (R_{\text{obliv}}, R_{\text{aware}})$, produces a sequence of q proof locations $I = (i^{(1)}, \ldots, i^{(q)})$, where each $i^{(k)} \in [m(n)]$.*

Think of V as representing a probabilistic oracle machine that queries proof oracle π and gets answer symbols $\pi|_I$, computes parity checks $P := (C_1(R_{\text{obliv}}), \ldots, C_p(R_{\text{obliv}}))$ and accepts iff $D(\pi|_I, P) = 1$.

We write $(I, P, D) \overset{R}{\sim} V(x)$ to denote the queries, predicate, and parities generated by V on input x. To explicitly mention the random coins R, we write $(I, P, D) \sim V(x; R)$.

We call p the parity-check complexity of V.

We view ROP as a secondary property to RNL and rectangularity, and for simplicity we sometimes omit it from informal discussions (e.g., the title Section 1.2). Indeed, in Section 1.5 we show a simple way of adding ROP to any PCP while essentially increasing only its decision complexity.

[11] These are affine functions of the oblivious part only, but they encompass parities on *all* of the randomness by including the parity of the aware part in the constant term.

A description of a rectangular verifier with ROP

All new PCP notions that are key to our work deal with a modified view of the run of a PCP verifier based on a partitioning of its randomness. Thus, let us take a moment to provide a streamlined description of a rectangular PCP verifier that has ROP, where the *shared* and *aware* parts of the randomness are the same.[12] We hope this description helps the reader picture the new properties of our main PCP verifier, which we eventually construct in Theorem 1.49, and use in our construction of rigid matrices (Section 1.2). Specifically, we wish to clarify the dependence of the queries, the decision predicate and the parity checks on the different parts of the randomness.

Note 1.15 (Rectangular verifier with ROP)**.** *Let $\tau \in (0,1)$, and let V be a τ-rectangular (r, q, p, ℓ^2, d, t)-verifier with τ-ROP. Assume further that the* shared *and* aware *parts of the randomness of V are the same,*[12] *such that its randomness R is partitioned as follows:*

$$\begin{aligned} R_{\text{obliv}} &= (R_{\text{row}}, R_{\text{col}}) \\ R_{\text{aware}} &= R_{\text{shared}} \\ R &= (R_{\text{row}}, R_{\text{col}}, R_{\text{shared}}) = (R_{\text{obliv}}, R_{\text{aware}}). \end{aligned}$$

Since the shared *and* aware *parts of the randomness are the same, we will refer only to the* shared *part of the randomness.*

The run of V given input x and proof oracle π can be described as follows:

1. *Sample* shared *randomness $R_{\text{shared}} \in \{0,1\}^{\tau \cdot r}$. Based on it,*
   a) *Construct a decision predicate $D := D(x; R_{\text{shared}})$ of size d.*
   b) *Construct randomness parity checks $(C_1, \ldots, C_p) := (C_1(x; R_{\text{shared}}), \ldots, C_p(x; R_{\text{shared}}))$.*
2. *Sample* row *randomness $R_{\text{row}} \in \{0,1\}^{(1-\tau)r/2}$. Construct proof row locations*
$$i^{(1)}_{\text{row}} := i^{(1)}_{\text{row}}(x; R_{\text{row}}, R_{\text{shared}}), \ldots, i^{(q)}_{\text{row}} := i^{(q)}_{\text{row}}(x; R_{\text{row}}, R_{\text{shared}}).$$
3. *Sample* column *randomness $R_{\text{col}} \in \{0,1\}^{(1-\tau)r/2}$. Construct proof column locations*
$$i^{(1)}_{\text{col}} := i^{(1)}_{\text{col}}(x; R_{\text{col}}, R_{\text{shared}}), \ldots, i^{(q)}_{\text{col}} := i^{(q)}_{\text{col}}(x; R_{\text{col}}, R_{\text{shared}}).$$
4. *Compute randomness parity checks $P := (C_1(R_{\text{row}}, R_{\text{col}}), \ldots, C_p(R_{\text{row}}, R_{\text{col}}))$.*
5. *Query the proof oracle to obtain $\pi|_I := \left(\pi_{i^{(1)}_{\text{row}}, i^{(1)}_{\text{col}}}, \ldots, \pi_{i^{(q)}_{\text{row}}, i^{(q)}_{\text{col}}}\right)$.*
6. *Output the result of the computation $D(\pi|_I, P)$.*

[12] This is indeed the case throughout this chapter.

PCPs of Proximity

Recall that we think of a PCP verifier as accepting input x and proof π if the answers received from π, denoted $\pi|_I$, satisfy the decision circuit D generated by V. In a nutshell, PCP composition is done by replacing the naive verification of the claim *"$\pi|_I$ satisfies D"* with a verification by an inner verifier.

The goal of PCP composition is to reduce the query complexity of an (outer) verifier by composing it with an inner verifier of smaller (even constant) query complexity. Hence, the inner verifier has restricted access not only to its proof, but also to part of its input (namely, $\pi|_I$). Since such a constrained verifier cannot distinguish between answers that satisfy D to those that are close to satisfying D, its soundness condition is relaxed to rejection of answers that are *far* from all satisfying assignments to D. Indeed, if the outer PCP had suitable robustness, this relaxation still yields a sound PCP. We formalize this discussion next.

Definition 1.16 (Pair language and CVP)**.** *A pair language L is a subset of $\{0,1\}^* \times \{0,1\}^*$. The Circuit Valuation Problem (CVP) is the pair language consisting of circuits and their accepting inputs. Formally,*

$$\mathsf{CVP} \coloneqq \{(C, y) \mid C(y) = 1\}.$$

A PCP of Proximity for a pair language L is given a pair (x, y) and a proof π, where access to x is *explicit* (i.e., x can be read entirely) while only oracle access is given to y and π (so queries to y are accounted for in the verifier's query complexity). The soundness condition is weakened to rejection with high probability only of (x, y) such that y is *far* from $L(x) \coloneqq \{y' | (x, y') \in L\}$. Formally:

Definition 1.17 (PCP of Proximity (PCPP))**.** *Let $L \subseteq \{0,1\}^* \times \{0,1\}^*$ be a pair language. For $s, \delta \colon \mathbb{N} \to \mathbb{N}$, a restricted verifier V over Σ is a PCP of proximity verifier for L with proximity parameter δ and soundness error s if the following two conditions hold for any $x, y \in \{0,1\}^*$:*

Completeness *If $(x, y) \in L$, then there exists a proof π such that $V(x)$ accepts the oracle $y\pi$ (y is called the* input oracle *and π is called the* proof oracle*) with probability 1. Formally,*

$$\exists \pi \qquad \Pr_{(I,D) \overset{R}{\sim} V(x)} [D(y\pi|_I) = 1] = 1.$$

Soundness *If y is δ-far from the set $L(x) = \{y' \mid (x, y') \in L\}$, then for every proof oracle π, $V(x)$ accepts the oracle $y\pi$ with probability strictly less than s. Formally,*

$$\forall \pi \qquad \Pr_{(I,D) \overset{R}{\sim} V(x)} [D(y\pi|_I) = 1] < s(|x|).$$

For convenience, we assume that the input locations $I = (i_1, \ldots, i_q)$ are each of the form $i_k = (b_k, j_k)$, where b_k is a bit signifying the oracle of the k-th query, and j_k is the location in that oracle. Formally, for each $k \in [q]$, if $b_k = 0$ (resp. $b_k = 1$) then $j_k \in [|y|]$ (resp. $j_k \in [|\pi|]$) is the location in y (resp. in π) of the k-th query of V.

1.1.1 Three additional preliminaries: Error-correcting codes, sampler graphs and λ-biased sets

Error-correcting codes, that are defined next, are used to reduce the alphabet size of the PCP.

Definition 1.18 (Error-correcting codes)**.** *A (binary) error-correcting code C is given by an encoding map $Enc : \{0,1\}^k \to \{0,1\}^n$. The rate R, which measures how much information can be packed into a codeword, is defined by $R = \frac{k}{n}$. The minimum distance δ, which measures the error-correcting capability of the code, is defined to be the smallest relative Hamming distance between $Enc(x)$ and $Enc(y)$ for distinct $x, y \in \{0,1\}^k$.*

The error-correction property of codes comes from the observation that for any given word $w \in \{0,1\}^n$, there is at most one $x \in \{0,1\}^k$ such that $Enc(x)$ is within distance $\delta/2$ of w and finding this x given w is the problem of decoding.

A *linear code* is an error-correcting code where the encoding map $Enc : \{0,1\}^k \to \{0,1\}^n$ is given by $Enc(x) = Gx$ for some $k \times n$ Boolean matrix G. A systematic error-correcting code is a code in which the input message bits are embedded in the encoded output (say the first k locations of the output). We use the following lemma which gives a constant rate and constant distance linear code such that the decoding and encoding time is linear in the RAM model.

Theorem 1.19 (Spielman 1996)**.** *There is a constant rate, constant distance linear error correcting code with a linear-time encoder, and a linear-time decoder recovering a message from a codeword with up to a fixed constant fraction of errors. Furthermore, the code is systematic.*

In our PCP construction, we will use explicit construction of *sampler graphs* that are defined next.

Definition 1.20 (Sampler graph)**.** *Fix $\alpha \in [0,1]$. A graph $G = (V, E)$ is an α-sampler if for every $S \subseteq V$,*

$$\Pr_{v \in V}\left[\left|\frac{|S|}{|V|} - \frac{|\Gamma(v) \cap S|}{|\Gamma(v)|}\right| > \alpha\right] < \alpha.$$

We will use the following efficient construction of sampler graphs.

Fact 1.21 (Section 5.1 of Goldreich 2011)**.** *There exists an algorithm that given an integer n and $\alpha \in (0,1)$, constructs a $(4/\alpha^3)$-regular graph on n vertices which is an α-sampler in time $\mathsf{poly}(n)$.*

Next, we need λ-biased sets in our construction of PCPs. We first recall basic notation about characters over a field $\mathcal{F}$. For a field $\mathcal{F}$ of characteristic p, a character of $\mathcal{F}^m$ is a homomorphism $\chi : \mathcal{F}^m \to \omega_p$, where ω_p is the (multiplicative) group of complex p^{th} roots of unity. In other words, $\chi(x+y) = \chi(x) \cdot \chi(y)$ for every $x, y \in \mathcal{F}^m$. The trivial character maps $\mathcal{F}^m$ to 1.

Definition 1.22. *(λ-biased sets) A set $S \subseteq \mathcal{F}^m$ is called λ-biased if for every character χ of $\mathcal{F}^m$, we have $|\mathbb{E}_{x\in S}[\chi(x)]| \leq \lambda$.*

We have the following explicit construction of λ-biased sets from Alon et al. (1992).

Lemma 1.23 (Alon et al. 1992)**.** *For every $\mathcal{F}$ of characteristic 2, $m \in \mathbb{Z}^+$, and $\lambda > 0$, there is an explicit construction of a λ-biased set $S \subseteq \mathcal{F}^m$ of size at most $O\left(\frac{\log(|\mathcal{F}|^m)}{\lambda}\right)^2$.*

The next lemma shows that we can have λ-biased set of similar size even when we restrict the set S to have elements from $\mathcal{F}^m$ with the first coordinate non-zero.

Lemma 1.24. *For every $\mathcal{F}$ of characteristic 2, $m \in \mathbb{Z}^+$, and $\lambda > \frac{3}{|\mathcal{F}|}$, there is an explicit construction of a λ-biased set $S_\lambda \subseteq \mathcal{F}^m$ with $y_1 \neq 0$ for all $y \in S_\lambda$ and $|S| = O\left(\frac{\log(|\mathcal{F}|^m)}{\lambda}\right)^2$.*

Proof. Take a $\lambda/10$-biased set S from Lemma 1.23. Consider the subset $B \subseteq \mathbb{F}$ where $B = \{a \mid \exists x \in S, x_1 = a\}$. If $0 \notin S$, then we are done. Otherwise, we consider two cases.

1. Case 1, $|B| < |\mathbb{F}|$: In this case, let $b \in \mathbb{F} \setminus B$ be the missing element from B. Consider the set $S' = \{x + (-b, 0, 0, \ldots, 0) \mid x \in S\}$, where the addition is a coordinate-wise addition over $\mathbb{F}$. Note, that for every $y \in S'$, $y_1 \neq 0$. Furthermore, for any character χ, we have

$$\left|\mathop{\mathbb{E}}_{y\in S'}[\chi(y)]\right| = \left|\mathop{\mathbb{E}}_{x\in S}[\chi(x)\cdot\chi((-b,0,0,\ldots,0))]\right| \leq \left|\mathop{\mathbb{E}}_{x\in S}[\chi(x)]\right| \leq \lambda/10,$$

   where we used the fact that $\chi(.)$ has absolute value at most 1.

2. Case 2, $|B| = |\mathbb{F}|$: In this case, suppose $b \in \mathbb{F}$ be a field element that occurs the least amount of time as the first coordinate in S (breaking ties arbitrarily). Consider the set $S' = \{x \mid x \in S, x_1 \neq b\}$. Note that $|S'| \geq (1 - 1/|\mathbb{F}|)|S|$. Furthermore, for any character χ, we have

$$\begin{aligned}\lambda/10 \geq \left|\mathop{\mathbb{E}}_{x\in S}[\chi(x)]\right| &= \left|\left(\frac{|S'|}{|S|}\right)\mathop{\mathbb{E}}_{x\in S}[\chi(x) \mid x \in S'] + \left(1 - \frac{|S'|}{|S|}\right)\mathop{\mathbb{E}}_{x\in S}[\chi(x) \mid x \notin S']\right| \\ &\geq \left|\left(\frac{|S'|}{|S|}\right)\mathop{\mathbb{E}}_{x\in S}[\chi(x) \mid x \in S']\right| - \left(1 - \frac{|S'|}{|S|}\right)\end{aligned}$$

   This implies that,

$$|\mathop{\mathbb{E}}_{y\in S'}[\chi(y)]| = |\mathop{\mathbb{E}}_{x\in S}[\chi(x) \mid x \in S']| \leq \frac{\frac{\lambda}{10} + \frac{1}{|\mathbb{F}|}}{(1 - \frac{1}{|\mathbb{F}|})}.$$

   Thus, when $\lambda > 3/|\mathbb{F}|$, S' is a λ-biased set where $B' = \{a \mid \exists y \in S', y_1 = a\}$ is such that $|B'| < |\mathbb{F}|$. Therefore, we can apply Case 1 to S' and get S'', a λ-biased set, with $y_1 \neq 0$ for all the $y \in S''$.

□

1.2 Main application: rigid matrices from rectangular PCPs

Alman and Chen (2022) show how to construct rigid matrices using efficient, short and smooth PCPs. In this section, we show how efficient, short, smooth and *rectangular* PCPs can be used to obtain a simpler and stronger construction of rigid matrices.

Lemma 1.25. *Let $\tau \in (0,1)$ and $\rho : \mathbb{N} \to \mathbb{N}$. Let $L \in \mathsf{NTIME}(2^n) \setminus \mathsf{NTIME}(O(2^n/n))$ be a unary language. Suppose L has a PCP with soundness error s and a (r, q, ℓ^2, d, t)-restricted verifier V over alphabet $\{0,1\}$ with $\ell(n) = \mathsf{poly}(2^n)$ strictly monotone increasing and computable in time $\mathsf{poly}(n)$. Assume further that V is τ-rectangular and has τ-ROP with parity-check complexity p, and that the* shared *and the* aware *parts of the randomness are the same. Lastly, assume that the following inequalities hold:*

1. $\frac{1+\tau}{2} \cdot r + \log(t+\rho) \leq n - \log n$.
2. $q + p + r - \Omega\left(\frac{(1-\tau)r}{\log((q+p)\rho)}\right) \leq n - \log n$.

Then, there is an FNP-machine such that, for infinitely many $N \in \mathbb{N}$, on input 1^N, outputs an $N \times N$ matrix with $\mathcal{F}_2$ entries which is $\left(\frac{1-s}{q} \cdot N^2, \rho(\ell^{-1}(N))\right)$-rigid.

To prove Lemma 1.25, we make use of the following fast algorithm that counts the number of 1's in a low rank matrix (given its low rank decomposition).

Theorem 1.26 (Chan and Williams 2021; Alman and Chen 2022)**.** *Given two matrices $A \in \mathcal{F}_2^{n \times \rho}$ and $B \in \mathcal{F}_2^{\rho \times n}$ where $\rho = n^{o(1)}$, there is a (deterministic) algorithm that computes the number of 1's in the matrix $A \cdot B$ in time $T(n,\rho) \coloneqq n^{2-\Omega\left(\frac{1}{\log \rho}\right)}$.*

In addition to the above fast counting algorithm, we need a simple claim on representing affine functions as a low-rank matrix.

Claim 1.27. *There is a procedure with the following properties:*

- *Input: An integer m, Boolean vectors $u, v \in \mathcal{F}_2^m$ and a bit $b \in \mathcal{F}_2$.*
- *Output: Two matrices, $A \in (\mathcal{F}_2)^{2^m \times 3}$ and $B \in (\mathcal{F}_2)^{3 \times 2^m}$, such that $(A \cdot B)_{x,y} = \langle x, u\rangle + \langle y, v\rangle + b$.*
- *Runtime $\widetilde{O}(2^m)$.*

Proof. We compute A column-by-column: The first column is an enumeration of $\langle x, u\rangle$, for all $x \in \mathcal{F}_2^m$; the second column is the all-ones vector, denoted by $\vec{1}$; the third column is $b\vec{1}$.

Next, we compute B row-by-row: The first row is $\vec{1}$; the second row is an enumeration of $\langle y, v\rangle$, for all $y \in \mathcal{F}_2^m$; the third row is $\vec{1}$.

It is easy to verify that $(A \cdot B)_{x,y} \equiv \langle x, u\rangle + \langle y, v\rangle + b$, and that the runtime is $O(2^m \cdot m) = \widetilde{O}(2^m)$. □

We now prove Lemma 1.25.

Proof of Lemma 1.25. Let V be the postulated PCP verifier for L. The FNP-machine computing rigid matrices (infinitely often) runs as follows. On input 1^N,

1. If $N = \ell(n)$ for some $n \in \mathbb{N}$: [13]
   a) Guess an $N \times N$ matrix denoted by π.
   b) Emulate V when given explicit input 1^n and proof π on all possible 2^r random coins. If V accepted on all randomness, accept and output π; else, reject.
2. Else ($N \neq \ell(n)$ for any n), reject.

This machine runs in time $n^{O(1)} + O(2^r \cdot t)$. As $O(2^r \cdot t) = \mathsf{poly}(2^n)$, which follows from Item 1 from Lemma 1.25, the machine runs in time $\mathsf{poly}(2^n) = \mathsf{poly}(\ell(n)) = \mathsf{poly}(N)$, and whenever $N = \ell(n)$ for some n such that $1^n \in L$, one of its non-deterministic guesses lead to acceptance by completeness of the PCP. Note that there could be multiple non-deterministic guesses that lead to acceptance. We show that for infinitely many $N = \ell(n)$ such that $1^n \in L$, any guessed π that leads to acceptance is $\left(\frac{1-s}{q} \cdot N^2, \rho\right)$-rigid for $\rho \coloneqq \rho(n) = \rho(\ell^{-1}(N))$.

Assume towards contradiction that this is not the case. Then, there exists an n_0 such that for any $n \geq n_0$, $1^n \in L$ if and only if there exists a proof π (for the verifier V) which is $\frac{1-s}{q}$-close to a rank ρ matrix. We describe a non-deterministic algorithm that decides L in time $O(2^n/n)$ – a contradiction. Given input 1^n,

1. Guess matrices A and B of dimensions $\ell \times \rho$ and $\rho \times \ell$ respectively. (The right guess is when $A \cdot B$ is δ-close to π; by smoothness, the acceptance probability of $A \cdot B$ will then be close to that of π, and the task is reduced to estimating the acceptance probability of $A \cdot B$.)
2. Compute the acceptance probability of $A \cdot B$ by V as follows. For each sequence of coins in the shared part of the randomness $R_{\text{shared}} \in \{0,1\}^{\tau \cdot r}$:
   a) Compute the predicate $D \coloneqq D(R_{\text{shared}})$ and randomness parity checks $C_j \coloneqq C_j(R_{\text{shared}})$, $j \in [p]$.
   b) **Prepare queries into proof:** For each $k \in [q]$,
      i. **Prepare left matrices:** Compute the $2^{(1-\tau)r/2} \times \rho$ matrix $A^{(k)}$ whose R_{row}-th row is just the row indexed by $i^{(k)}_{\text{row}}(R_{\text{row}}, R_{\text{shared}})$ in A, for any $R_{\text{row}} \in \{0,1\}^{(1-\tau)r/2}$.
      ii. **Prepare right matrices:** Compute the $\rho \times 2^{(1-\tau)r/2}$ matrix $B^{(k)}$ whose R_{col}-th column is just the column indexed by $i^{(k)}_{\text{col}}(R_{\text{col}}, R_{\text{shared}})$ in B, for any $R_{\text{col}} \in \{0,1\}^{(1-\tau)r/2}$.

[13] This can be done in time $\mathsf{poly}(n) = \mathsf{poly}(\log N)$ by guessing the an integer n and verifying that $\ell(n) = N$.

Observe that $\left(A^{(k)} \cdot B^{(k)}\right)_{R_{\text{row}},R_{\text{col}}}$ is exactly the k-th bit read by the verifier on randomness $(R_{\text{row}}, R_{\text{col}}, R_{\text{shared}})$.

c) **Prepare randomness parity checks:** For each $j \in [p]$,

i. Compute the $2^{(1-\tau)r/2} \times 3$ matrix $A^{(q+j)}$ and the $3 \times 2^{(1-\tau)r/2}$ matrix $B^{(q+j)}$, for which $\left(A^{(q+j)} \cdot B^{(q+j)}\right)_{R_{\text{row}},R_{\text{col}}} = C_j(R_{\text{row}}, R_{\text{col}})$. Such matrices exist and can be computed in time $O(r \cdot 2^{(1-\tau)r/2})$, as described in Claim 1.27.

d) **Fast counting:** Fourier analysis tells us that there are (unique) coefficients $\left\{\widehat{D}(K)\right\}_{K\subseteq[q+p]}$ such that for any $y \in \{0,1\}^{q+p}$,

$$D(y_1, \ldots, y_{q+p}) = \sum_{K\subseteq[q+p]} \widehat{D}(K)(-1)^{\bigoplus_{i\in K} y_i}.$$

In particular, by linearity of expectation, to compute the expected value (which is also the acceptance probability) of $D(y)$ over a random y sampled from some distribution, it suffices to compute the expected value of all parity predicates on over y, namely $\mathbb{E}_y\left[\bigoplus_{i\in K} y_i\right]$ for all $K \subseteq [q+p]$.

This observation is useful because the final task is to compute the acceptance probability of the predicate D on inputs $\left(A^{(1)} \cdot B^{(1)}\right)_{R_{\text{row}},R_{\text{col}}}, \ldots, \left(A^{(q+p)} \cdot B^{(q+p)}\right)_{R_{\text{row}},R_{\text{col}}}$ for uniformly random R_{row} and R_{col}. Thus, it suffices to compute the acceptance probability of all parity predicates, i.e., the number of 1's in $\bigoplus_{k\in K} A^{(k)} \cdot B^{(k)}$ for each $K \subseteq [q+p]$.

For each K, note that $\bigoplus_{k\in K} A^{(k)} \cdot B^{(k)}$ is the product of a $2^{(1-\tau)r/2} \times (|K|\rho)$ and a $(|K|\rho) \times 2^{(1-\tau)r/2}$ matrix over $\mathcal{F}_2$ (namely, concatenate the rows of the $|K|$ matrices $\{A^{(k)}\}_{k\in K}$ and concatenate the columns of the $|K|$ matrices $\{B^{(k)}\}_{k\in K}$). Thus, for each K, computing the acceptance probability of $\bigoplus_{k\in K} A^{(k)} \cdot B^{(k)}$ can be done with the fast counting algorithm for low-rank matrices of Theorem 1.26. Its runtime is

$$T(2^{(1-\tau)r/2}, (q+p)\cdot\rho) = \left(2^{(1-\tau)r/2}\right)^{\left(2-\Omega\left(\frac{1}{\log((q+p)\cdot\rho)}\right)\right)} = 2^{(1-\tau)r-\Omega\left(\frac{(1-\tau)r}{\log((q+p)\cdot\rho)}\right)}$$

3. We have thus computed the acceptance probability of $A \cdot B$ by the verifier V. If this probability is at least the soundness error s, decide that the input 1^n is in L. Otherwise, decide that the input is not in L.

We claim that the algorithm correctly decides L, for input length $n \geq n_0$. Indeed, if $1^n \in L$ then, when guessing A and B such that $A \cdot B$ is $(\frac{1-s}{q})$-close to the correct proof π for 1^n (such A and B exist by our assumption towards contradiction), smoothness of the verifier V implies that its acceptance probability is at least $1 - q \cdot (\frac{1-s}{q}) = s$. On the other hand, if $1^n \notin L$, then soundness of V implies any guessed A and B leads to rejection with probability strictly less than s. Since n_0 is constant we can hard-wire the values of L on 1^n for $n < n_0$ so that the algorithm correctly decides L on all inputs.

As for its runtime, observe that Item 2a takes time $O(t)$, Item 2b takes time $O(2^{(1-\tau)r/2} \cdot (t+\rho))$ and Item 2c takes time $O(r \cdot 2^{(1-\tau)r/2})$. Since $t \geq r$, these are dominated by $O(2^{(1-\tau)r/2} \cdot (t+\rho))$. Therefore the runtime of the algorithm is

$$\begin{aligned} &O\Big(2^{\tau \cdot r} \cdot \Big(2^{\frac{(1-\tau)r}{2}} \cdot (t+\rho) \;+\; 2^{q+p} \cdot 2^{(1-\tau)r - \Omega\left(\frac{(1-\tau)r}{\log((q+p)\cdot\rho)}\right)}\Big)\Big) \\ &= O\Big(2^{\frac{(1+\tau)r}{2}} \cdot (t+\rho) \;+\; 2^{q+p+r-\Omega\left(\frac{(1-\tau)r}{\log((q+p)\cdot\rho)}\right)}\Big). \end{aligned}$$

By the assumption on the parameters of the PCP, this is at most $O(2^n/n)$ – a contradiction. □

Now that we formalized a connection between rectangular PCPs and rigid matrices, let us introduce the PCP that we construct in the remainder of this chapter, and show how it implies the rigid matrix construction asserted in Theorem 1.2.

Theorem 1.28 (Theorem 1.49, instantiated)**.** *For any $L \in \mathsf{NTIME}(2^n)$, and constants $s \in (0, 1/2)$ and $\tau \in (0,1)$, L has a PCP verifier over alphabet $\{0,1\}$ with the following parameters:*

- *Randomness complexity $r(n) = n + O(\log n)$*
- *Proof length $\ell(n) = 2^n \cdot \mathsf{poly}(n)$.*
- *Soundness error s.*
- *Decision, query and parity-check complexities all $O(1)$.*
- *Verifier runtime $t(n) = 2^{O(\tau n)}$.*
- *The verifier is τ-rectangular and has τ-ROP. Furthermore, the shared and the aware parts of the randomness are the same.*

Theorem 1.28 is obtained by instantiating Theorem 1.49 with parameters $T(n) \coloneqq 2^n$ and $m \coloneqq \Omega(1/\tau)$, and noting that the proof length is $\ell(n) \leq 2^{r(n)} \cdot q(n) = 2^n \cdot \mathsf{poly}(n)$. Next, we restate the rigid matrix construction asserted in Theorem 1.2 and reckon that it is obtained by combining Theorem 1.28 and Lemma 1.25.

Corollary 1.29 (Theorem 1.2, restated)**.** *There is a constant $\delta \in (0,1)$ such that there is an FNP-machine that for infinitely many N, on input 1^N outputs an $N \times N$ matrix that is $(\delta \cdot N^2, 2^{\log N / \Omega(\log\log N)})$-rigid.*

Proof. From Zák (1983), there exists a unary language $L \in \mathsf{NTIME}(2^n) \setminus \mathsf{NTIME}(O(2^n/n))$. Let L be any such language. Fix $\rho(n) \coloneqq 2^{n/(K \log n)}$ for a large enough constant K to be determined later. We verify that the parameters of the PCP of Theorem 1.28, for a sufficiently small $\tau \in (0,1)$, satisfy all the conditions from Lemma 1.25 for this ρ. Let q and

p denote the query complexity and parity-check complexity of the PCP respectively. Set $\delta = (1-s)/q$. Now, for small enough τ,

$$\left(\tfrac{1+\tau}{2}\right) \cdot r + \log(\rho + t) \leq (1/2 + O(\tau))n + O(n/\log n) < n - \log n,$$

as required in Item 1. Also, the parameters satisfy Item 2, because

$$q + p + r - \Omega\left(\frac{(1-\tau)r}{\log((q+p)\rho)}\right) \leq n + O(\log n) - \Omega(K \log n) < n - \log n,$$

where the last inequality holds for a suitable choice of the constant K. As the proof length is $\ell(n) = 2^n \cdot \mathsf{poly}(\log n)$, we have $\ell^{-1}(N) = \Theta(\log N)$. Therefore, $\rho(\ell^{-1}(N)) = 2^{\log N/\Omega(\log\log N)}$ and the corollary follows from Lemma 1.25. □

Remark 1.30. *The only bottleneck preventing Lemma 1.25 from giving rigid matrices for polynomial ranks, $\rho = N^{\Omega(1)}$, via Theorem 1.28 is the runtime of the counting algorithm used in Item 2d. That is, such results would be obtained if there was an algorithm for counting the number of nonzero entries in a rank $N^{\Omega(1)}$ matrix (of dimensions $N \times N$) that ran in time slightly better than $O(N^2)$. Our reduction would go through even if the algorithm's answer was only* approximately *correct (while losing the respective approximation factor in the distance of the resulting matrices from low rank ones). In fact, this seems to be the only bottleneck even up to rank $\rho = N^{1-O(\tau)}$.*

1.3 From rectangular neighbor-listing (RNL) to smooth and rectangular PCPs

In this section, we show how any PCP verifier with Recatngular Neighbor Listing (RNL) can be made into a smooth and rectangular PCP. This conversion preserves the ROP.

Theorem 1.31. *Suppose L has a PCP with verifier V_{old} as described in Table 1.1, and τ-RNL and τ-ROP such that the* shared *and* aware *parts of the randomness are the same. Let t_{RNL} be the running time of the row and column neighbor-listing agents of V_{old}. Then for any $\mu \in (0,1)$, L has a PCP with verifier V_{new} as described in Table 1.1 which is* smooth*, τ-rectangular and τ-ROP such that the* shared *and* aware *parts of the randomness are the same.*

The smooth proof system of Theorem 1.31 utilizes an explicit construction of sampler graphs from Fact 1.21.

Complexity	V_{old}	V_{new}
Alphabet size	2	2
Soundness error	s	$s + \mu$
Randomness	r	r
Query	q	$\mathsf{poly}(q/\mu)$
Parity-check	p	p
Proof length	m	$2^r \cdot q$
Decision	d	$d + \mathsf{poly}(q/\mu)$
Runtime	t	$t + q \cdot \mathsf{poly}(t_{\text{RNL}})$

Table 1.1: The complexities of the original V_{old} and the smooth verifier V_{new}.

1.3.1 The smooth and rectangular PCP

The smooth and rectangular PCP verifier is obtained by applying the degree reduction transformation of Dinur and Harsha, 2013, Theorem 5.1. We restate this transformation with syntactic changes that will be helpful for showing rectangularity.[14]

Let V_{old} be the verifier postulated in Theorem 1.31 and denote by V_{new} the new, smooth and rectangular, verifier. We start by describing the proofs expected by V_{new}.

Proofs in the new PCP system. New proofs are of length $2^r \cdot q$, which we think of as indexed over $\{0,1\}^r \times [q]$. Each location in the new proof corresponds to a full configuration of the original verifier. Correct proofs for V_{new} are as follows: For an input $x \in L$ and a correct proof π for V_{old} (i.e., one that is accepted w.p. 1), the (R,k)-th location of the correct proof for V_{new} will have the answer of π to the k-th query issued by V_{old} upon sampling random coin sequence R. Notice that in a correct proof for the new verifier, any two locations in the new proof corresponding to neighboring configurations (see Definition 1.11) should take the same value.

The new verifier. The basic idea is for the new verifier to emulate the original one: when the original samples coin sequence R, the new one queries locations $(R,1),\ldots,(R,q)$ in the new proof. However, if the original verifier queried the same location $i \in [\ell]$ for two different (neighboring) configurations (R,k) and (R',k'), a new ("cheating") proof could be *inconsistent* in its answers, using this inconsistency to cause the new verifier to accept when the original would not.

Thus, consistency between neighboring configurations must be checked. To guarantee smoothness and preserve the randomness of the new verifier, consistency is checked only between certain neighboring configurations, and not all. Namely, neighboring configurations

[14] Our presentation is from the "proof systems" perspective of PCP verifiers, rather than the "label cover" perspective given in Dinur and Harsha, 2013, Theorem 5.1.

are connected by a $O(\mu/q)$-sampler graph, with edges on the sampler corresponding to consistency tests. Two configurations that are adjacent on the sampler are said to be sampler-neighbors, which is a stronger condition than being neighbors as per Definition 1.11.

In fact, both consistency and the original PCP verification are done in one fell swoop: when emulating the original verifier, the new verifier replaces a query to (R, k) with queries to its sampler-neighborhood, and checks its consistency. A sampler graph guarantees that inconsistency between neighboring configurations is reflected by this test w.h.p., so severely inconsistent proofs are rejected by the new verifier. On the other hand, regularity of the sampler implies smoothness, and its degree incurs only a small blowup to the number of queries.

The point of this theorem is in showing rectangularity of the new verifier. Specifically, we ought to show how construction of the sampler and sampler-neighborhoods can be done *rectangularly*. That is, for any location $(R_{\text{row}}, R_{\text{col}}, R_{\text{shared}}, k)$, it is not enough to find all other sampler-neighboring $(R'_{\text{row}}, R'_{\text{col}}, R'_{\text{shared}}, k')$; it should be the case that the *row-part* of the sampler-neighbors can be found based on $(R_{\text{row}}, R_{\text{shared}}, k)$. Similarly, the *column-part* of the sampler-neighbors should be found only from $(R_{\text{col}}, R_{\text{shared}}, k)$.

We clarify what we mean by "row-part" and "column-part" of a query. The new proof (of length $2^r \cdot q$) can be thought of as a square matrix as follows: Fix a location $(R_{\text{row}}, R_{\text{col}}, R_{\text{shared.row}}, R_{\text{shared.col}}, k$ in the new proof. Split k into k_{row} and k_{col}. The rows of the matrix are indexed by $(R_{\text{row}}, R_{\text{shared.row}}, k_{\text{row}})$, and the columns are indexed by $(R_{\text{col}}, R_{\text{shared.col}}, k_{\text{col}})$. Indeed, with this definition, it is possible to find the row-parts (resp., column-parts) of the sampler-neighbors based on $(R_{\text{row}}, R_{\text{shared}}, k)$ (resp., $(R_{\text{row}}, R_{\text{shared}}, k)$) thanks to RNL.

Following is a detailed description of this construction.

Algorithm 1.32. Fix the original verifier V_{old}.

1. Sample a coin sequence R.
2. For each $k \in [q]$, construct the sampler of the neighborhood of (R, k) and check consistency of its sampler-neighborhood *in a rectangular way* as follows: Denote the randomness partition by of R by $(R_{\text{row}}, R_{\text{col}}, R_{\text{shared}})$.
   a) Find the "row parts": Compute $L_{\text{row}} \coloneqq A_{\text{row}}(R_{\text{row}}, R_{\text{shared}}, k)$ where A_{row} is the neighbor listing agent. Construct a canonical $(\mu/3q)$-sampler on the set of $|L_{\text{row}}|$ vertices, one corresponding to every entry in the list L_{row}. From the index of (R, k) in the list L_{row}, find the indices of the sampler-neighbors of (R, k), and output their "row-part" $(R'_{\text{row}}, R'_{\text{shared.row}}, k'_{\text{row}})$. In addition, output the row-part of (R, k).
   b) Find the "column part": Similarly, compute $L_{\text{col}} \coloneqq A_{\text{col}}(R_{\text{col}}, R_{\text{shared}}, k)$, construct a canonical $(\mu/3q)$-sampler on the set of $|L_{\text{col}}|$ vertices, find the indices of the sampler-neighbors of (R, k), and output their "column-part" $(R'_{\text{col}}, R'_{\text{shared.col}}, k'_{\text{col}})$. In addition, output the column-part of (R, k).

3. Let $\Delta - 1$ denote the degree of the sampler above. Note that we are making $\Delta \cdot q$ many queries. Feed the $\Delta \cdot q$ bits queried from the proof π_{new} to a circuit that first checks consistency between every sampler-neighborhood. That is, it checks that in each of the q blocks of Δ bits, all the Δ bits are equal. If an inconsistency is spotted, the circuit immediately rejects. Otherwise, feed the first bit in every block to the decision circuit of the original verifier V_{old} (along with the p parity-checks on the randomness) and output its answer.

1.3.2 Proof of Theorem 1.31

Rectangularity. The randomness of the new verifier is split exactly the same as the original verifier into R_{row}, R_{col} and $R_{\text{shared}} = (R_{\text{shared.row}}, R_{\text{shared.col}})$. It follows from the description of the algorithm that R_{row} and R_{shared} determine the row-part of each of the $q \cdot \Delta$ queries. Similarly, R_{col} and R_{shared} determine the column-part of each of the $q \cdot \Delta$ queries.

ROP. The new decision predicate can be implemented by taking a circuit that checks equality on each of the q sampler-neighborhoods constructed in Item 2, and ANDing its answer with the output of original decision circuit (fed an arbitrary representative of each sampler-neighborhood, as well as the randomness parity checks). Therefore, the τ-ROP is preserved.

Query and decision complexities. By Fact 1.21, the size of each sampler-neighborhood (in a $(\mu/3q)$-sampler) is $\mathsf{poly}(q/\mu)$. A sampler-neighborhood is queried for each of the q original, so the query complexity is $\mathsf{poly}(q/\mu)$. As described in the ROP analysis, the new decision circuit can be obtained by ANDing the original decision circuit (of size d) to $\mathsf{poly}(q/\mu)$ equality checks. Thus, the size of the new circuit is $d + \mathsf{poly}(q/\mu)$.

Runtime complexity. The new verifier emulates the original one. In addition, for each of the q queries it invokes RNL agents and finds a neighborhood in the sampler. Invoking RNL agents takes t_{RNL} time. Constructing the explicit sampler on the configuration's neighborhood and finding its sampler-neighborhood takes time at most $\mathsf{poly}(t_{\text{RNL}})$ time (we upper bound the size of each list with the runtime of each agent).

With rectangularity out of the way, we can describe the run of the new verifier given proof π_{new} a more succinct way:

Algorithm 1.33 (Algorithm 1.32, simplified)**.** Given input x and proof π_{new}, the new verifier V_{new} runs as follows:

1. Sample $R \in \{0,1\}^r$.
2. For each $k \in [q]$, query π_{new} for (R, k) as well as its sampler-neighbors.

3. Feed the $q \cdot \Delta$ bits queried from the proof π_{new} to a circuit that first checks consistency between every sampler-neighborhood. That is, it checks that in each of the q blocks of Δ bits, all the Δ bits are equal. If an inconsistency is spotted the circuit immediately rejects. Otherwise, feed the first bit in every block to the decision circuit of the original verifier V_{old} (along with the p parity-checks on the randomness) and output its answer.

Indeed, Algorithm 1.32 and Algorithm 1.33 describe the same verifier precisely due to RNL: for any random coin sequence R and query index k, Item 2 of Algorithm 1.32 indeed queries all neighbors of (R, k) in the sampler (and then checks their consistency). We now show that this verifier is sound and smooth.

Smoothness. For each $R \in \{0,1\}^r$ and $k \in [q]$, let $\bar{\Gamma}(R, k)$ denote the closed sampler-neighborhood of (R, k), i.e., the union of the sampler-neighborhood of (R, k) and the singleton containing it $\{(R, k)\}$. Recall that the sampler is a regular graph with constant degree $\Delta - 1$, and thus $\Delta = \left|\bar{\Gamma}(R, k)\right|$ for each R, k.

Now recall how the new verifier determines its queries (see Algorithm 1.33): sample $R \in \{0,1\}^r$, and for each $k \in [q]$ query all the Δ locations in $\bar{\Gamma}(R, k)$. We must now show why this procedure is equally likely to query each location in π_{new}.

Fix a location (R', k') in the new proof. Notice that

$$\Pr_{\substack{R \in \{0,1\}^r \\ k \in [q]}} \left[(R', k') \in \bar{\Gamma}(R, k)\right] = \frac{\Delta}{2^r \cdot q}, \tag{1.2}$$

where both R and k are distributed uniformly and random. Recall that the new verifier makes $q \cdot \Delta$ queries to the proof, where the (k, j)-th query, for $k \in [q]$ and $j \in [\Delta]$, is $\bar{\Gamma}(R, k)[j]$. We thus get

$$\Pr_{\substack{R \in \{0,1\}^r \\ k \in [q], j \in [\Delta]}} \left[(R', k') = \bar{\Gamma}(R, k)[j]\right] = \frac{1}{2^r \cdot q}$$

showing that each location (R', k') is equally likely to be queried by the new verifier.

The fact that the samplers we construct are Δ-regular graphs implies that every location in the new proof is read with exactly the same probability.

Soundness. One way to see soundness (as well as smoothness) would be to observe Algorithm 1.32 is the same as the verifier of Dinur and Harsha, 2013, Theorem 5.1, and is therefore sound (and smooth). Since the latter theorem and its proof are described in the "label cover" view of PCPs whereas our work takes the "proof systems" view, we present an alternative proof in the latter view next.

Let $\alpha := \mu/3q$. Recall that V_{new} denotes the new smooth verifier, and V_{old} denotes the original verifier. Fix an input $x \notin L$ and an alleged proof π_{new} for V_{new}. We will show that V_{new} rejects x and π_{new} with probability at least $1 - s - \mu$.

We let $i^{(k)}(R)$ denote the location of the k-th query of V_{old} when sampling random coins R. Recall that locations in π_{new} are indexed by full configurations (R, k), that can be partitioned into m disjoint α-samplers, where the j-th sampler connects all configurations (R, k) such that $i^{(k)}(R) = j$. We derive a proof π_{old} for the original verifier V_{old} by assigning $\pi_{\text{old}}(j)$ the majority value of π_{new} on the j-th sampler. Formally,

$$\pi_{\text{old}}(j) \coloneqq \mathsf{Maj}_{R,k}\big\{\pi_{\text{new}}(R, k) \;\big|\; i^{(k)}(R) = j\big\}.$$

The soundness of V_{new} follows from the following claim.

Claim 1.34. $\Pr_R[V_{\text{new}} \text{ accepts } \pi_{\text{new}}] \leq s + \mu$.

Proof. For $j \in [m]$, let C_j be the set of all the configurations (R, k) such that $i^{(k)}(R) = j$. We say that C_j (or the sampler defined on C_j) has consistency η if η-fraction of the configurations in C_j satisfy $\pi_{\text{new}}(R, k) = \pi_{\text{old}}(j)$. We partition the set of random strings $\{0,1\}^r$ into B_1, B_2, B_3 as follows.

1. A string $R \in B_1$ iff there exists a k such that (R, k) is in C_j whose consistency is at most $1 - 2\alpha$.
2. A string $R \in B_2$ iff $R \notin B_1$ and there exists k such that $\pi_{\text{new}}(R, k) \neq \pi_{\text{old}}(j)$ where $i^{(k)}(R) = j$, and
3. $B_3 = \{0,1\}^r \setminus (B_1 \cup B_2)$.

We start with writing the probability as follows:

$$\Pr_R[V_{\text{new}} \text{ accepts } \pi_{\text{new}}] = \sum_{i=1}^{3} \Pr[R \in B_i] \cdot \Pr[V_{\text{new}} \text{ accepts } \pi_{\text{new}} \mid R \in B_i]$$

Consider a sampler with consistency at most $1 - 2\alpha$. The average value[15] of π_{new} on this sampler is between 2α and $1 - 2\alpha$. Since this is a α-sampler, it holds that for at least $(1 - \alpha)$-fraction of configurations (named error configurations), the average value of π_{new} on the sampler-neighborhood of each configuration is between α and $1 - \alpha$. In particular, π_{new} assigns inconsistent values to the sampler-neighbors of each error configuration and hence V_{new} rejects on R whenever (R, k) is an error configuration for some $k \in [q]$. Form this we conclude that

$$\begin{aligned}\Pr[V_{\text{new}} \text{ accepts } \pi_{\text{new}} \mid R \in B_1] &\leq \Pr_{R \in B_1}[\forall k \in [q], (R, k) \text{ is an not an error configuration}] \\ &\leq \alpha q. \end{aligned} \tag{1.3}$$

To see the last inequality, let $\mathcal{C}$ be the set of samplers where the consistency of each of these samplers is at most $1 - 2\alpha$. Let $B_1' \subseteq B_1$ be the set of random strings R such that

[15] Recall that π_{new} is over $\{0,1\}$ and we interpret these as real numbers.

$\forall k \in [q], (R,k)$ is an not an error configuration. We are interested in the ratio $|B_1'|/|B_1|$. As for each $C_j \in \mathcal{C}$, at least $(1-\alpha)$-fraction of configurations in C_j are error configurations, we have $|B_1'| \leq \alpha \sum_{C_j \in \mathcal{C}} |C_j|$. Furthermore, we have $\sum_{C_j \in \mathcal{C}} |C_j| \leq q|B_1|$, as each random string R can only occur at most q times in any sampler. Combining this with the previous inequality gives $|B_1'|/|B_1| \leq \alpha q$, as required.

Next, we calculate $\Pr_R[R \in B_2]$. For every $R \in B_2$, as $R \notin B_1$, all its configurations belong to the a sampler with consistency at least $1-2\alpha$. Furthermore, $|B_2| \leq 2\alpha q \cdot 2^r$ as, by definition, at most 2α fraction of the configurations (R',k') from a sampler on C_j with consistency at least $1-2\alpha$ satisfy $\pi_{\text{new}}(R',k') \neq \pi_{\text{old}}(j)$ where $i^{(k')}(R') = j$. Therefore,

$$\Pr_R[R \in B_2] \leq 2\alpha q. \tag{1.4}$$

Finally, for $R \in B_3$, we have

$$\begin{aligned} \Pr[R \in B_3] \cdot \Pr[V_{\text{new}} \text{ accepts } \pi_{\text{new}} \mid R \in B_3] &= \Pr[V_{\text{new}} \text{ accepts } \pi_{\text{new}} \cap (R \in B_3)] \\ &\leq \Pr[V_{\text{old}} \text{ accepts } \pi_{\text{old}} \cap (R \in B_3)] \\ &\leq \Pr[V_{\text{old}} \text{ accepts } \pi_{\text{old}}] \\ &\leq s. \end{aligned} \tag{1.5}$$

Here, the first inequality follows from the fact that when $R \in B_3$, we have $\pi_{\text{new}}(R,k) = \pi_{\text{old}}(j)$ where $i^{(k)}(R) = j$, for every $k \in [q]$ and hence if V_{new} accepts π_{new} on R, then V_{old} also accepts π_{old} on R.

Combining (1.3), (1.4), and (1.5), we get

$$\Pr_R[V_{\text{new}} \text{ accepts } \pi_{\text{new}}] \leq \alpha q + 2\alpha q + s = \mu + s,$$

as required. □

1.4 A many-query robust PCP with RNL

In this section, we prove that the Reed–Muller-based PCP of Ben-Sasson et al. (2006) and Ben-Sasson et al. (2005) has RNL. This PCP issues many queries, but is *robust* – which will become useful later in the composition stage (Section 1.6) to reduce its query complexity. In particular, we modify the many-query robust PCP of Ben-Sasson et al. (2006) and Ben-Sasson et al. (2005) to obtain the following PCP with RNL.[16]

Theorem 1.35 (Strengthening Theorem 3.1 of Ben-Sasson et al. (2006), simplifying Ben-Sasson et al. 2005)**.** *Suppose that L is a language in $\mathsf{NTIME}(T(n))$ for some non-decreasing function $T\colon \mathbb{N} \to \mathbb{N}$. There exists a universal constant c such for all odd integers $m \in \mathbb{N}$ and $s \in (0,1/2)$ satisfying $T(n)^{1/m} \geq m^{cm}/s^6$ and $\min\{1/c \log n, s^3/m^{cm}\} \geq (T(n)^{1/m} \cdot \mathsf{poly}\log T(n))^{-1/2}$, L has a robust PCP with the following parameters:*

[16] Ben-Sasson et al. (2005) used an object they called a *verifier specification*. For the sake of simplicity, we do not use this object, and this costs us an extra $q(n)$ factor in the running time. This loss of $q(n)$ has no effect on our application to rigid matrices.

1. *Alphabet* $\{0,1\}$.
2. *Randomness complexity* $r(n) = (1 - \frac{1}{m}) \log T(n) + O(m \log \log T(n)) + O(\log(1/s))$.
3. *Decision and Query complexity* $d(n) = q(n) = T(n)^{1/m} \cdot \mathsf{poly}(\log T(n), 1/s)$.
4. *Robust soundness error* s *with robustness parameter* $\Theta(s)$.
5. *Runtime complexity* $t(n) = q(n) \cdot \mathsf{poly}(n, \log T(n))$.
6. *The PCP verifier has* τ*-RNL with running time* $t_{\mathrm{RNL}}(n)$ *where*

$$\tau \cdot r(n) = r_{\mathrm{shared}} = \frac{4}{m} \log T(n) + O(m \log \log T(n)) + O(\log(1/s)),$$
$$t_{\mathrm{RNL}}(n) = \mathsf{poly}(\log T(n)).$$

Remark 1.36. *Items 1 to 4 are exactly as in the statement of Ben-Sasson et al., 2006, Theorem 3.1, the outer robust PCP construction of Ben-Sasson et al. while Item 5 (the verifier running time) is obtained by the efficient PCP verifiers of Ben-Sasson et al. (2005). The main difference between the two works of Ben-Sasson et al. (2006) and Ben-Sasson et al. (2005) is that the latter uses a reduction from Succinct-SAT to Succinct-Multivariate-Algebraic-CSP. To show that these PCPs have RNL, we need to analyze the query and predicate of the corresponding PCP verifiers. Since these are almost identical in both the constructions (i.e., the original robust PCP construction and the subsequent efficient version of it), we work with the robust PCP of Ben-Sasson et al. (2006) and just observe that these modifications can be carried out efficiently as in Ben-Sasson et al. (2005).*

Remark 1.37. *We remark that the runtime complexity of the Ben-Sasson et al. (2006) and Ben-Sasson et al. (2005) verifier is in fact* $q(n) \cdot \mathsf{polylog}(T(n)) + O(n)$ *(as observed by Chen et al. 2020). This improvement is obtained by constructing a robust PCP of proximity variant of Theorem 1.35 (which constructs only a robust PCP), and then converting it to a robust PCP using a standard transformation based on linear time-encodable error-correcting codes. However, this improvement is not needed for our main result.*

The robust PCP verifiers of Ben-Sasson et al. (2006) and Ben-Sasson et al. (2005) already have the properties listed in Items 1 to 5. In this section we show that it also has RNL as stated in Item 6 above. This is done in two steps: First (Section 1.4.1), we show that the robust PCP from Ben-Sasson et al., 2006, Section 8.2.1 has RNL, albeit over a large alphabet. Second (Section 1.4.2), we reduce the alphabet size to binary while preserving RNL.

1.4.1 The robust RNL PCP verifier over a large alphabet

Lemma 1.38. *There exists a robust PCP verifier for a language in* $\mathsf{NTIME}(T(n))$ *with the properties mentioned in Theorem 1.35, but over an alphabet of size* $2^{\mathsf{polylog}(T(n))}$ *(instead of the Boolean alphabet) and with expected robustness.*[17]

For convenience, we remind the reader of the PCP verifier from Ben-Sasson et al. (2005), following its presentation in Section 8.2.1 of Ben-Sasson et al. (2006) (see Remark 1.40 for a minor difference). That same work shows completeness and expected robustness of this verifier. In Section 1.4.1, we show how to convert PCP with RNL that has expected robustness ρ to a PCP with RNL that has robustness parameter $\Omega(\rho)$, thereby proving Theorem 1.35 albeit with an alphabet of size $2^{\mathsf{polylog}(T(n))}$. Keeping this issue of expected robustness aside for now, we first show that this PCP has RNL, so it suffices for us to recall its *query patterns* without detailing the way its *decision (predicate)* is made based on these queries.

Algorithm 1.39 (Verifier query pattern Ben-Sasson et al., 2006, Section 8.2.1)**.** Let $\mathcal{F}$ be a field of size $|\mathcal{F}| = T(n)^{1/m} \cdot \mathsf{poly}\log T(n)$ where m is an odd integer. The proof oracle is a map $\Pi\colon \mathcal{F}^m \to \mathcal{F}^d$ where $d = m \cdot \mathsf{poly}(\log T(n))$. Let $\mathsf{shift}\colon \mathcal{F}^m \to \mathcal{F}^m$ denote the linear transformation that cyclically shifts each coordinate to the left; i.e., $\mathsf{shift}(x_1, \ldots, x_m) \coloneqq (x_2, x_3, \ldots, x_m, x_1)$. Let $S_\lambda \subseteq \mathcal{F}^m$ be a λ-biased set of size $O((m \log |\mathcal{F}|/\lambda)^2)$ from Lemma 1.24, for $\lambda = \min\{1/c \log n, s^3/m^{cm}\}$ where c is a large constant. Note that by the condition specified in Theorem 1.35, we have $\lambda \geq \frac{1}{\sqrt{|\mathcal{F}|}}$.

The queries will be based on *lines* through $\mathcal{F}^m$, and their shifts. Recall that the line $\mathcal{L}$ with intercept $x \in \mathcal{F}^m$ and direction $y \in \mathcal{F}^m$ is the set $\mathcal{L} \coloneqq \{x + ty\colon t \in \mathcal{F}\}$. The verifier queries proceeds as follows:

1. Sample $x \in \{0\} \times \mathcal{F}^{m-1}$ uniformly at random, and let $\mathcal{L}_0$ denote the (first-axis parallel) line with intercept x and direction $y = (1, 0, \ldots, 0)$. Query the proof oracle on $\mathcal{L}_0$ and $\mathsf{shift}(\mathcal{L}_0)$.

2. Sample a direction y' from the λ-biased set S_λ. Note that $y'_1 \neq 0$. Let $\mathcal{L}_1$ be the line with direction y' and intercept x (from the previous step). Query the proof oracle on $\mathcal{L}_1$.

Remark 1.40 (Differences in the query pattern of Ben-Sasson et al. 2006; Ben-Sasson et al. 2005)**.** *The only difference in the robust PCP construction of Ben-Sasson et al. (2006) and its efficient counterpart in Ben-Sasson et al. (2005) is the reduction from* $\mathsf{NTIME}(T(N))$ *to (succinct) Multivariate-Algebraic-CSP (which in turn uses the reduction of Pippenger and Fischer (1979)) Ben-Sasson et al., 2005, Definition 6.3, Theorem 6.4. This causes the field size to increase from* $O(m^2 \cdot T(n)^{1/m})$ *to* $T(n)^{1/m} \cdot \mathsf{poly}\log T(n)$.

[17] i.e., in expectation over the PCP randomness, we need to change at least a ρ-fraction of bits that the verifier reads in order to make it accept.

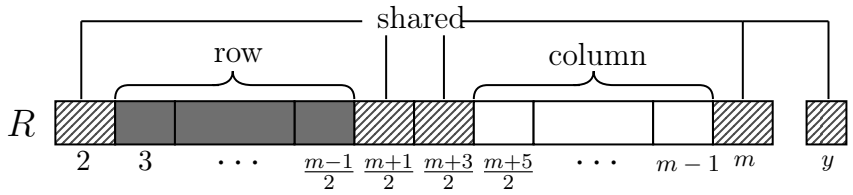


Figure 1.1: The partition of the randomness of Algorithm 1.39.

Proof of Lemma 1.38. We prove that the verifier of Algorithm 1.39 has RNL. First, note that the verifier uses a total of $\log(|S_\lambda|) + (m-1)\log(|\mathcal{F}|)$ random bits to sample a first-axis parallel line $\mathcal{L}_0$ and a canonical pseudorandom line $\mathcal{L}_1$, reusing the random bits between these two lines. [18]

The randomness used for sampling x is partitioned into $(m-1)$ parts of equal length, denoted by $(R_2, R_3, \ldots, R_m)$ where $|R_i| = \log(|\mathcal{F}|)$, and R_i determines x_i for each $i \in \{2, \ldots, m\}$ (recall that $x_1 = 0$ always). The randomness used to sample a direction from S_λ is denoted by R_y. Recall that there are two-types of lines, canonical and axis-parallel lines. In both cases the direction of the line y is a function of the bits R_y (in the axis-parallel line the direction is just the constant function).

The row, column and shared parts of the randomness are portrayed in Fig. 1.1. Formally, partition $R_y = (R_{y.\text{row}}, R_{y.\text{col}})$ arbitrarily, and let

$$\begin{aligned} R_{\text{row}} &\coloneqq \left(R_3, \ldots, R_{(m-1)/2}\right) \\ R_{\text{col}} &\coloneqq \left(R_{(m+5)/2}, \ldots, R_{m-1}\right) \\ R_{\text{shared.row}} &\coloneqq (R_2, R_{(m+1)/2}, R_{y.\text{row}}) \\ R_{\text{shared.col}} &\coloneqq (R_{(m+3)/2}, R_m, R_{y.\text{col}}) \\ R_{\text{shared}} &\coloneqq (R_{\text{shared.row}}, R_{\text{shared.col}}). \end{aligned}$$

Thus, the randomness has parts of length $r_{\text{row}} \coloneqq |R_{\text{row}}|$, $r_{\text{col}} \coloneqq |R_{\text{col}}|$, $r_{\text{shared}} \coloneqq |R_{\text{shared}}|$, with total randomness $r = r_{\text{row}} + r_{\text{col}} + r_{\text{shared}}$ and $\tau \cdot r = r_{\text{shared}}$.

Rectangular Neighbor-Listing (RNL). The BGHSV verifier makes $4 \cdot |\mathcal{F}|$ queries. We index the queries by $k := (b_1, b_2, t) \in \{0,1\}^2 \times \mathcal{F}$ as follows.

- $b_1 = 0$ indicates the query is to a line. $b_1 = 1$ indicates it is to a shifted line.
- $b_2 = 0$ indicates the query is to a first-axis parallel line. $b_2 = 1$ indicates it is to a canonical line.
- $t \in \mathcal{F}$ indicates the position on the line.

[18] The size of the set S_λ and a detailed description of how we derived this query pattern based on the tests of the verifier of Ben-Sasson et al., 2006, Section 8.2.1 can be found in Bhangale et al., 2024, Appendix A.

Given a configuration (R, k) that results in a query to location $z \in \mathcal{F}^m$, we ought to show how to list all neighboring configurations (i.e., configurations that lead to location z) in a rectangular way. We first show who are the neighboring configurations, and then show how to rectangularly (and synchronously) list them by describing the listing agents A_{row} and A_{col}. This shows that the verifier has RNL.

Neighbors of (R, k). A full configuration $(R, k) = (R_{\text{row}}, R_{\text{col}}, R_{\text{shared}}, k)$ specifies $k = (b_1, b_2, t)$, $x = (0, R_2, \ldots, R_m)$, and $R_y \in [|S_\lambda|]$. R_y and b_2 determine $y \in \mathcal{F}^m$ as follows: if $b_2 = 0$ then $y = (1, 0, \ldots, 0)$, otherwise $y = S_\lambda[R_y]$.

Recall that $\mathsf{shift}(x)$ denotes cyclic shift of x one step to the left. Given the full configuration specified by $k = (b_1, b_2, t)$, x and R_y, the location of the kth query will be $x + t \cdot y$ if $b_1 = 0$, or $\mathsf{shift}(x + t \cdot y)$ if $b_1 = 1$. More concisely, letting $\mathsf{shift}_j(x)$ denote the cyclic shift of x by j steps for any $j \in \mathbb{Z}$, the location of the kth query is $\mathsf{shift}_{b_1}(x + t \cdot y)$.

Thus, any other configuration (R', k') that specifies $k' = (b'_1, b'_2, t')$, x' and R'_y would query the same location if and only if

$$\mathsf{shift}_{b'_1}(x' + t' \cdot y') = \mathsf{shift}_{b_1}(x + t \cdot y). \tag{1.6}$$

In other words, (R', k') neighbors (R, k) if and only if it satisfies Eq. (1.6).

The neighbor listing agents. Rearranging Eq. (1.6), we see that $(x, b_1, b'_1, t, t', y, y')$ uniquely determines x' by the equation

$$x' = \mathsf{shift}_{b_1 - b'_1}(x + t \cdot y) - t' \cdot y', \tag{1.7}$$

Recall that any configuration (R', k') must fulfill the condition $x'_1 = 0$. For any possible $k' = (b'_1, b'_2, t')$ and y', fulfillment of this condition is determined only by $k = (b_1, b_2, t)$, y and (x_2, x_m). Thus, we say that a partial configuration $\left(R'_y, k'\right)$ is $\mathsf{realizable}$ (for (R_{shared}, k)) if and only if $x'_1 = 0$.

Furthermore, notice that for any (R'_y, b'_1, b'_2) there is a unique t' such that Eq. (1.7) has $x'_1 = 0$, and finding such t' can be done in a constant number of arithmetic operations over $\mathcal{F}$. This will come in handy shortly, when we construct the listing agents.

We can now present two algorithms listing all neighbors of a given configuration in a rectangular and synchronized fashion. First the row neighbor-listing agent A_{row}. On input row configuration $(R_{\text{row}}, R_{\text{shared}}, k)$,

1. Obtain t, b_1, b_2, y, $x_{[2,(m+3)/2]}$ and x_m from the input. Initialize an empty list L_{row}.
2. For each $\left(R'_y, b'_1, b'_2\right)$:
   a) Find $t' \in \mathcal{F}$ such that $\left(R'_y, k'\right)$ is realizable (as previously explained).
   b) Compute $x'_{\text{row}} \coloneqq x'_{[2,(m+1)/2]}$ as in Eq. (1.7). Append $\left(x'_{\text{row}}, R'_y, k'\right)$ to the list L_{row}.

3. Sort L_{row} according to $\left(R'_y, k'\right)$. Output L_{row}.

The column neighbor-listing agent runs similarly. On input column configuration $(R_{\text{col}}, R_{\text{shared}}, k)$,

1. Obtain t, b_1, b_2, y, x_2 and $x_{[(m+1)/2,m]}$ from the input. Initialize an empty list L_{col}.
2. For each $\left(R'_y, b'_1, b'_2\right)$:
   a) Find $t' \in \mathcal{F}$ such that $\left(R'_y, k'\right)$ is realizable (as previously explained).
   b) Compute $x'_{\text{col}} := x'_{[(m+3)/2,m]}$ as in Eq. (1.7). Append $\left(x'_{\text{col}}, R'_y, k'\right)$ to the list L_{col}.
3. Sort L_{col} according to $\left(R'_y, k'\right)$. Output L_{col}.

Note that both agents give lists indexed and sorted by all realizable (R'_y, k'). Thus, both arrays are of the same length and ordering. Moreover, the ordering of L_{row} and L_{col} is the same when the agents are given as inputs any two neighboring configurations (R, k) and (R', k'). Thus, Item 2 of Definition 1.12 is satisfied.

For each $i \in [|L_{\text{row}}|]$, concatenating $L_{\text{row}}[i]$ and $L_{\text{col}}[i]$ gives a tuple $\left(x'_{\text{row}}, R'_y, k', x'_{\text{col}}, R'_y, k'\right)$. By splitting R'_y arbitrarily into $R'_{y.\text{row}}$ and $R'_{y.\text{col}}$ we have that L_{row} and L_{col}, when appropriately "zipped" (as in Item 1 of Definition 1.12) give the list L of all neighboring configurations to (R, k).

To see Item 3 of Definition 1.12, note that both A_{row} and A_{col} can identify the index of (R, k) in L since it corresponds to the index of the entry whose two last elements are (R_y, k) in L_{row} and in L_{col}.

Lastly, we calculate t_{RNL}, which is the runtime of A_{row} (the case of A_{col} is analogous). Item 1 takes $\mathsf{poly}(m \cdot \log|\mathcal{F}| \cdot \log(1/s))$ time; this includes getting $y \in S_\lambda$ from a random string R_y using the efficient construction of λ-biased set in Lemma 1.24. For each (R'_y, b'_1, b'_2), Item 2a takes $\mathsf{poly}(\log|\mathcal{F}|)$ time, and Item 2b takes $\mathsf{poly}(m \cdot \log|\mathcal{F}|)$ time: using Eq. (1.7) for each $i \in [m]$, x'_i can be computed using one coordinate from x, y and y' and performing addition/multiplication over the field $\mathcal{F}$. The output list length is upper bounded by $4|S_\lambda| = \mathsf{poly}(m \cdot \log|\mathcal{F}| \cdot \log(1/s))$, so sorting it in Item 3 takes at most $\mathsf{poly}(m \cdot \log|\mathcal{F}| \cdot \log(1/s))$ time. Thus, overall the running time is dominated by $\mathsf{poly}(m \cdot \log|\mathcal{F}| \cdot \log(1/s))$. By the choice of parameters in Theorem 1.35 and size of $\mathcal{F}$, this is asymptotically equal to $\mathsf{poly}(\log T(n))$. □

Getting the Robust Soundness Error

In order to get the strong robust soundness (instead of expected robustness), we need to modify the PCP verifier as follows (as suggested by Ben-Sasson et al. (2006)). Consider a d-regular expander graph G on $\{0,1\}^r$, the set of random strings, as the vertex set. On randomness $R \sim \{0,1\}^r$, the new verifier looks at all the neighbors of R in G (in total, we have $(d+1)$ strings) and runs the original verifier on all these strings as the verifier's randomness. The new verifier accepts iff the original verifier accepts on all the $(d+1)$ strings. Thus,

- As shown by Ben-Sasson et al. (2006), the transformation maintains the robust soundness error and the robustness parameter up to a constant factor (the constant factor can be taken to be 1/2) by choosing a good expander with $d = \mathsf{poly}(1/s)$.
- The randomness complexity of the new verifier stays the same.
- The decision and query complexity blows up by a multiplicative factor of $(d + 1) = \mathsf{poly}(1/s)$ and therefore we get the parameters as stated in Lemma 1.38.

Instead of using any expander, we can use an expander which is a tensor product of 4 expanders. This will facilitate the RNL property for the modified outer PCP verifier. The tensor product of two graphs G_1 and G_2, is the graph denoted by $G_1 \times G_2$, with vertex set $V(G_1 \times G_2) = V(G_1) \times V(G_2)$ and any two of its vertices (u_1, v_1) and (u_2, v_2) are adjacent, whenever u_1 is adjacent to u_2 in G_1 and v_1 is adjacent to v_2 in G_2.Let $G = G_{\text{row}} \times G_{\text{shared.row}} \times G_{\text{shared.col}} \times G_{\text{col}}$ be a degree-d expander graph that is used to transform the PCP as stated above. With this, it can be shown that the RNL property is preserved. More formally, we can show the following.

Lemma 1.41. *Suppose a language L has a PCP with verifier V as described in Table 1.2, then L has a PCP with verifier V' as described in Table 1.2. Furthermore, if V has τ-RNL, then so does V'.*

Complexity	V	V'
Alphabet	Σ	Σ
Robust soundness error	s	$\Omega(s)$
Robustness parameter	ρ	$\Omega(\rho)$
Randomness	r	r
Query	q	$q \cdot \mathsf{poly}(1/s)$
Proof length	m	m
Decision	d	$d \cdot \mathsf{poly}(1/s)$
Runtime	t	$t \cdot \mathsf{poly}(1/s)$
RNL agent runtime	t_{RNL}	$t_{\text{RNL}} \cdot \mathsf{poly}(1/s)$

Table 1.2: The complexities of original verifier V and the new verifier V'.

Proof. The modified verifier V' is as described at the beginning of this section with $G = G_{\text{row}} \times G_{\text{shared.row}} \times G_{\text{shared.col}} \times G_{\text{col}}$ as an expander with degree d. Let d_1, d_2, d_3 and d_4 be the degrees of the graphs $G_{\text{row}}, G_{\text{shared.row}}, G_{\text{shared.col}}$ and G_{col}, respectively with $d_i = \Theta(d)$ for all $i \in [4]$. Note that $d = d_1 \cdot d_2 \cdot d_3 \cdot d_4$. By setting each of the graphs $G_{\text{row}}, G_{\text{shared.row}}, G_{\text{shared.col}}$ and G_{col} to be an expander with the second largest eigenvalue λ (of the normalized adjacency matrix of the graphs), it follows that G will be an expander with the second largest eigenvalue

of the normalized adjacency matrix of G at most λ. As we need $d = \mathsf{poly}(1/s)$, we can use such a tensor product expander, in which we incur polynomial loss in the degree compared to the expansion parameter, in the modified outer verifier V'.

We can associate a one-to-one map ζ from $[d]$ to $[d_1] \times [d_2] \times [d_3] \times [d_4]$ in a natural way. It makes sense to index the queries of V' by a pair (k, i) where $k \in [q]$ and $i \in \{0, 1, ..., d\}$. The queries $(k, 0)$ are exactly the original queries. We can also assume that the expanders are consistently labeled, so if u is the i-th neighbor of v, then v is also the i-th neighbor of u. We note that $(R, (k, i))$ is equivalent to $(P, (k', i'))$ (meaning they both result in looking at the same index of the proof) iff (R', k) and (P', k') are equivalent according to the original PCP, where R' is the i-th neighbor of R in the expander G we introduced, and P' is the i'-th neighbor of P in G. Once this is observed, it is easier to verify that the new PCP has an RNL algorithm, assuming G has a tensor product structure. We now proceed to a formal proof.

Let A_{row} and A_{col} are the row and column neighbor-listing agents of V, respectively. Here are the new algorithms A'_{row} and A'_{col} for the modified outer verifier V'. Recall that the verifier V' is issuing $(d+1)q$ queries on randomness R. These queries correspond to the queries of V on randomness from the set $R \cup_{i \in [d]} R_i$, where R_i is the ith neighbor of R in G (based on a specified ordering of the neighbors in G). For simplicity, let $k \leq q$ ($k > q$ can be handled in a similar way, but just to make things easier to understand, we restrict the description here to the case when $k \leq q$ here).

First the row neighbor-listing agent A'_{row}. On input row configuration $(R_{\text{row}}, R_{\text{shared}}, k)$,

1. Run the original A_{row} on the input. Let L_1 be the list generated by A_{row}. Let $L'_{\text{row}} \leftarrow L_1$.

2. For each $(R'_{\text{row}}, R'_{\text{shared.row}}, k')$ in the list L_1,
   For each $(i, j) \in [d_1] \times [d_2]$,
   For each $(i', j') \in [d_3] \times [d_4]$,
   Go to the (i, j)th neighbor $(\tilde{R}_{\text{row}}, \tilde{R}_{\text{shared.row}})$ of $(R'_{row}, R'_{\text{shared.row}})$ in $G_{\text{row}} \times G_{\text{shared.row}}$ and append $(\tilde{R}_{\text{row}}, \tilde{R}_{\text{shared.row}}, q + k' + \zeta^{-1}((i, j)^{-1}, (i', j')^{-1}))$ to L'_{row}. Here $(i, j)^{-1}$ is the pair (x, y) such that $(R'_{\text{row}}, R'_{\text{shared.row}})$ is the (x, y)th neighbor of $(\tilde{R}_{\text{row}}, \tilde{R}_{\text{shared.row}})$ in $G_{row} \times G_{\text{shared.row}}$.

To verify the correctness of the row neighbor-listing agent, in addition to the entries from the list L_1 above, the final list should also contain the random string $\tilde{R}$ which is a neighbor of R' in G where R' is in the joint list generated by $(A_{\text{row}}, A_{\text{col}})$ along with the proper query index. To achieve this, in Step 2. the agent goes over all the entries $(R'_{\text{row}}, R'_{\text{shared.row}}, k')$ in the list L_1 and adds all the row-part of the randomness from the neighbors of R' in G. The tensor product structure of the expander G allows the agent to list all such row-part of the randomness without knowing R' fully. The query index is calculated by noting

that if a location in the proof is queried by V on the configuration (R', k'), then the same location is queried by V' on the configuration $(\tilde{R}, i+k')$ where $\tilde{R}$ is the ith neighbor of R' in G.

For completeness, we state the algorithm A'_{col} which on randomness $(R_{\text{col}}, R_{\text{shared}})$ and k works as follows:

1. Run the original A_{col} on the input. Let L_1 be the list generated by A_{col}. Let $L'_{\text{col}} \leftarrow L_1$.
2. For each $(R'_{\text{col}}, R'_{\text{shared.col}}, k')$ in the list L_1,
   For each $(i, j) \in [d_1] \times [d_2]$,
   For each $(i', j') \in [d_3] \times [d_4]$,
   Go to the (i', j')th neighbor $(\tilde{R}_{\text{shared.col}}, \tilde{R}_{\text{col}})$ of $(R'_{\text{shared.col}}, R'_{\text{col}})$ in $G_{\text{shared.col}} \times G_{\text{col}}$ and append $(\tilde{R}_{\text{col}}, \tilde{R}_{\text{shared.col}}, q + k' + \zeta^{-1}((i,j)^{-1}, (i',j')^{-1}))$ to L'_{col}.

It is easy to observe that the lists $L'_{\text{row}}, L'_{\text{col}}$ satisfy the properties listed in the RNL property of the PCP verifier V' based on the discussion above. □

1.4.2 Alphabet Reduction

Lemma 1.41 gives a robust PCP with RNL *over a large alphabet*, but the final construct requires a PCP over the *Boolean alphabet*. Next, we show that standard alphabet reduction (Forney, 1965) preserves RNL. Namely, each symbol is replaced with its encoding in a binary error correcting code. We use a constant rate and constant distance code such that the decoding and encoding time is linear.

Lemma 1.42. *Suppose language L has a PCP with verifier V as described in Table 1.3, then L has a PCP with verifier V' as described in Table 1.3. Furthermore, if V has τ-RNL, then so does V'.*

Complexity	V	V'
Alphabet	Σ	$\{0,1\}$
Robust soundness error	s	s
Robustness parameter	ρ	$\Omega(\rho)$
Randomness	r	r
Query	q	$O(q \cdot \log \lvert\Sigma\rvert)$
Proof length	m	$O(m \cdot \log \lvert\Sigma\rvert)$
Decision	d	$d \cdot \mathsf{polylog}(\lvert\Sigma\rvert)$
Runtime	t	$t \cdot \mathsf{polylog}(\lvert\Sigma\rvert)$
RNL agent runtime	t_{RNL}	$O(t_{\text{RNL}} \cdot \log\log \lvert\Sigma\rvert)$

Table 1.3: The complexities of original verifier V and the Boolean verifier V'.

Proof. Fix a linear-time (in the RAM model) computable and decodable error correcting code of constant rate and distance, denoted $\mathrm{Enc}\colon \Sigma \to \{0,1\}^{\sigma}$ for $\sigma = O(\log|\Sigma|)$ from Theorem 1.19. Note that since the code is *systematic*, the first $\log|\Sigma|$ bits in the encoding are the binary representation of the (non-binary) message. The new (Boolean) proof is written in a natural way: each non-binary symbol is replaced with its encoding under Enc.

The Boolean PCP verifier V' emulates the non-Boolean verifier V as follows: when the non-Boolean verifier queries a location $b \in [m]$, the Boolean verifier queries the whole block $[(b-1)\sigma+1, b\sigma]$ and checks if it is a valid encoding of the symbol specified by the first $\log|\Sigma|$ bits (if not, reject). Once all the queries are decoded correctly, the verifier V' does the verification on the decoded values as V.

It is easy to observe that the query complexity and the proof length increase by a factor of $O(\log|\Sigma|)$. Since the new verifier has to perform the decoding of an error correcting code, this adds a multiplicative overhead of $\mathsf{polylog}(|\Sigma|)$ (on a multi-tape turing machine) in the running time and the decision complexity.

The importance of using the error correcting code is to make sure that the new verifier V' is still robust. This follows from Ben-Sasson et al., 2006, Lemma 2.13, where it was shown that the soundness error remains the same and the robustness parameter decreases by a constant factor.

It is also easy to observe that RNL is preserved with the same partition of the randomness. Fix a j-th location from the block $[(b-1)\sigma+1, b\sigma]$. First observe the following proposition:

Proposition (a): If a full configuration (R, k) queries a location $b \in [m]$ in the original proof, then the configuration $(R, \sigma(k-1)+j)$ queries the j-th location from the block $[(b-1)\sigma+1, b\sigma]$ in the new proof and vice-versa.

Let A_{row} and A_{col} are the row and column agents of the non-Boolean verifier V. The row agent $A'_{\mathrm{row}}(R_{\mathrm{row}}, R_{\mathrm{shared.row}}, \tilde{k}) \to L'_{\mathrm{row}}$ and the column agent $A'_{\mathrm{col}}(R_{\mathrm{col}}, R_{\mathrm{shared.col}}, \tilde{k}) \to L'_{\mathrm{col}}$ of the new verifier V' are as follows: Both the agents first compute the block number $k = \lceil \tilde{k}/\sigma \rceil$ and the index $j = \tilde{k} - \sigma(k-1)$. Next, the agents compute

$$L'_{\mathrm{row}}[i] \coloneqq (R'_{\mathrm{row}}, R'_{\mathrm{shared.row}}, \sigma(k'-1)+j) \quad s.t. \quad (R'_{\mathrm{row}}, R'_{\mathrm{shared.row}}, k') = L_{\mathrm{row}}[i]$$
$$L'_{\mathrm{col}}[i] \coloneqq (R'_{\mathrm{col}}, R'_{\mathrm{shared.col}}, \sigma(k'-1)+j) \quad s.t. \quad (R'_{\mathrm{col}}, R'_{\mathrm{shared.col}}, k') = L_{\mathrm{col}}[i]$$

where $L_{\mathrm{row}} \leftarrow A_{\mathrm{row}}(R_{\mathrm{row}}, R_{\mathrm{shared.row}}, k)$ and $L_{\mathrm{col}} \leftarrow A_{\mathrm{col}}(R_{\mathrm{col}}, R_{\mathrm{shared.col}}, k)$. To see the correctness, suppose the full configuration $(R_{\mathrm{row}}, R_{\mathrm{col}}, R_{\mathrm{shared.row}}, R_{\mathrm{shared.col}}, k)$ queries the location $b \in [m]$ from the original proof. By RNL of V, the full configuration $(R'_{\mathrm{row}}, R'_{\mathrm{col}}, R'_{\mathrm{shared.row}}, R'_{\mathrm{shared.col}}, k')$ given by $L_{\mathrm{row}}[i]$ and $L_{\mathrm{col}}[i]$ leads to the same location $b \in [m]$. Using Proposition (a), we can conclude that the verifier V' when given the full configuration $(R'_{\mathrm{row}}, R'_{\mathrm{col}}, R'_{\mathrm{shared.row}}, R'_{\mathrm{shared.col}}, \sigma(k'-1)+j)$, queries the location $(b-1)\sigma+j$ in the new proof. As $\tilde{k} = \sigma(k-1)+j$, again using Proposition (a), the input full configuration to the agents $(R'_{\mathrm{row}}, R'_{\mathrm{col}}, R'_{\mathrm{shared.row}}, R'_{\mathrm{shared.col}}, \tilde{k})$ also leads to the same location $(b-1)\sigma+j$ in the new proof. Therefore, every full configuration given by $(L'_{\mathrm{row}}[i], L'_{\mathrm{col}}[i])$ leads to the location $(b-1)\sigma+j$ in the new proof. Furthermore, since the number of full configurations on which V queries $b \in [m]$ is the same as the number

of full configurations on which V' queries any fixed location from the block $[(b-1)\sigma+1, b\sigma]$ (in fact, there is a bijection given by Proposition (a)), the list $(L'_{\text{row}}, L'_{\text{col}})$ is exhaustive. □

We now finish the proof of Theorem 1.35.

Proof of Theorem 1.35. Lemma 1.41 shows the existence of a verifier with the additional τ-RNL over a large alphabet. The alphabet reduction technique from Lemma 1.42 converts the PCP to a PCP over the Boolean alphabet. Since, the original PCP is over an alphabet of size $2^{\mathsf{polylog}(T(n))}$, this conversion increases the proof length, query complexity, decision complexity and the verifier's running time by a multiplicative factor of $\mathsf{polylog}(T(n))$. With all these changes, these four parameters of the new verifier are asymptotically same as the ones mentioned in Theorem 1.35.

In the whole process, the robustness parameter of the verifier changes by a constant multiplicative factor and this change is irrelevant in proving the lemma. □

1.5 Adding randomness oblivious predicates (ROP) to a robust PCP

One way at looking at the verification procedure is as follows: On sampling the randomness R, the verifier constructs a circuit $D := D(R)$ and a subset $I := I(R)$ of proof locations of size q. The verifier outputs the verdict of $D(\pi|_I)$. In this abstract way, the circuit D depends on the full randomness R. However, for our application we need the verifier to have randomness-oblivious predicates (ROP).

Recall that the ROP states that the decision predicate depends only on a small fraction of the randomness, but may take as input a limited number of parity checks on the entire randomness. We generalize robust soundness to the ROP setting in the natural way, measuring the distance of both the bits read by the verifier *as well as the randomness parity checks* from satisfying the decision predicate. This definition will be useful when we compose a robust PCP having ROP with a PCPP (in Section 1.6).

Definition 1.43 (robust soundness for ROP verifier)**.** *For functions $s, \rho : \mathbb{N} \to [0,1]$, a PCP verifier V for a language L with τ-ROP and parity check complexity p has* robust-soundness error *s with* robustness parameter *ρ if the following holds for every $x \notin L$: For every oracle π, with probability strictly less than s, the input to the decision predicate (that consists of bits read by the verifier and parities of the randomness) are ρ-close to being accepted. Formally,*

$$\forall \pi \quad \Pr_{(I,D,P) \overset{R}{\sim} V(x)} [\exists a, b \text{ s.t. } D(ab) = 1 \text{ and } \delta(ab, \pi|_I P) \leq \rho] < s(|x|).$$

Lemma 1.44. *There exists a constant $C \geq 1$ such that if the language L has a PCP with verifier V as described in Table 1.4, then L has a PCP with verifier V' as described in Table 1.4 with 0-ROP. Furthermore, if V has τ-RNL, then so does V'.*

Complexity	V	V'
Robust soundness error	s	s
Robustness parameter	ρ	$\Omega(\rho)$
Randomness	r	r
Query	$q \geq Cr$	q
Proof length	m	m
Decision	d	$\widetilde{O}(t)$
Parity-check	$-$	q
Runtime	t	$\widetilde{O}(t) + \mathsf{poly}(q)$
RNL agent runtime	t_{RNL}	t_{RNL}

Table 1.4: The complexities of the original verifier V and the 0-ROP verifier V'.

Proof. We can replace the circuit D of V, which depends on the randomness R, with another circuit D_{ROP} such that $D_{\mathrm{ROP}}(x, R) = D(x)$ for all $x \in \{0,1\}^q$ and randomness R. In other words, we give R as an explicit input to the circuit D_{ROP} and therefore remove the dependence of D_{ROP} on R. Note however that the output of D_{ROP} depends on both the randomness R as well as the proof locations $\pi|_I$.

It is easy to see that this preserves the completeness and soundness of the PCP. However, this transformation might lose the guarantee on the *robust soundness* when we look at the input to the circuit. This is because even if $\pi|_I$ is far from satisfying D, it might be the case that $(\pi|_I, R)$ is close to satisfying D_{ROP} (in this case when measuring the distance from satisfying answers, changes in R are also allowed).

In order to overcome this issue we encode the randomness with a good error code. By using the code from Theorem 1.19, we have a linear error correcting code $\mathrm{Enc}\colon \{0,1\}^r \to \{0,1\}^q$ with constant relative distance. Based on its linear time decoder, two circuits are constructed:

- Circuit Dec that on input $y \in \{0,1\}^q$ outputs $x \in \{0,1\}^r$ such that $\mathrm{Enc}(x) = y$ if such x exists, and an arbitrary value otherwise.[19]
- Circuit Test that on input $y \in \{0,1\}^q$ outputs 1 if y is a codeword (i.e., in the image of Enc), and 0 if it is not.

The actual decision circuit generated by V' is $D'(z, y) \coloneqq D_{\mathrm{ROP}}(z, \mathrm{Dec}(y)) \wedge \mathrm{Test}(y)$. The parity checks generated by V' are simply $\mathrm{Enc}(R)$ – indeed, since the encoding is linear then $\mathrm{Enc}(R)_1, \ldots, \mathrm{Enc}(R)_q$ are linear functions in R.

[19] In fact, the circuit only needs decode valid codewords, which is easier than decoding noisy codewords.

Decision and runtime complexities. The original verifier V constructs D in time t. The circuit D_{ROP} can emulate this using $\widetilde{O}(t)$ gates (Cook, 1988). By Theorem 1.19 (See Spielman 1996), the circuits Dec and Test can be constructed using at most $\widetilde{O}(q) = \widetilde{O}(t)$ gates. All in all, the size of the new decision circuit D' is at most $\widetilde{O}(t)$.

Apart from constructing D', the verifier V' also produces the queries of V and the parity checks $\mathrm{Enc}(R)$. Queries are constructed in time at most t, and again by Theorem 1.19 computing $\mathrm{Enc}(R)$ takes time $\mathsf{poly}(q)$ on a multi-tape machine. Thus the runtime of V' is at most $\widetilde{O}(t) + \mathsf{poly}(q)$.

Robust soundness. The original verifier has robust-soundness error s with robustness parameter ρ, so with probability at least $1 - s$ the input $\pi|_I$ to the circuit $D := D(R)$ is at least ρ-far from satisfying D. This means that with probability at least $1 - s$, $(\pi|_I, E(R))$ is at least $\min\{\rho/2, \Omega(1)\}$ far from satisfying D'. To see that, note that to satisfy D' either the first half of the input $\pi|_I$ needs to be changed in $q \cdot \rho$ locations, or the second half of the input $E(R)$ needs to be changed to another legal encoding $E(R')$ which by the properties of E requires $\Omega(q)$ bit-changes. Thus, the robustness parameter of V' is $\min\{\rho/2, \Omega(1)\} \geq \Omega(\rho)$, and the robust soundness error remains s. □

1.6 RNL-preserving PCP composition

In this section, we strengthen the composition theorem of Ben-Sasson et al. (2006) by showing that it preserves RNL and (to some extent) ROP of a robust PCP.

1.6.1 The composition theorem

Ben-Sasson et al. (2006) show that the composition of a robust PCP V_{out} and a PCP of proximity V_{in} with suitable parameters yields a sound $\mathsf{composite}$ PCP (described in Fig. 1.2) that enjoys the inner query complexity and (roughly) the outer randomness complexity.

Theorem 1.45 (Ben-Sasson et al., 2006, Theorem 2.7)**.** *Suppose language L has a robust PCP verifier V_{out}, and that CVP has a PCPP verifier V_{in}, with parameters described in Table 1.5 such that $\rho_{\mathrm{out}} \geq \delta_{\mathrm{in}}$. Then, language L has a PCP verifier V_{comp} as described in Table 1.5.*

Our goal is to reduce the query complexity of the PCP of Section 1.4 by composing it with a constant-query inner PCPP (see Section 1.7). Towards this end, we adapt the composition to consider ROP (described in Fig. 1.3), and strengthen Theorem 1.45 by showing that it preserves RNL of the outer PCP, and (to some extent) its ROP.

Lemma 1.46. *In the setting of Theorem 1.45, assume further that V_{out} has 0-ROP with parity-check complexity p_{out}, and τ-RNL with listing agent runtime of t_{RNL}. Then V_{comp} has*

Hardwired: Outer verifier V_{out} and inner verifier V_{in}.

Input: Explicit input x, outer proof Π and inner proofs $\{\pi_{R_{\text{out}}} \mid R_{\text{out}} \in \{0,1\}^{r_{\text{out}}}\}$.

1. Sample $R_{\text{out}} \in \{0,1\}^{r_{\text{out}}}$.
2. Run $V_{\text{out}}(x; R_{\text{out}})$ to obtain $I_{\text{out}} = (i_1, i_2, \ldots, i_{q_{\text{out}}})$ and D_{out}.
3. Sample $R_{\text{in}} \in \{0,1\}^{r_{\text{in}}}$.
4. Run $V_{\text{in}}(D_{\text{out}}, R_{\text{in}})$ to obtain $I_{\text{in}} = ((b_1, j_1), \ldots, (b_{q_{\text{in}}}, j_{q_{\text{in}}}))$ and D_{in}.
5. Initialize $I_{\text{comp}} := \emptyset$, and $D_{\text{comp}} := D_{\text{in}}$
6. For each $k \in [q_{\text{in}}]$, determine the queries of the composite verifier:
   a) If $b_k = 0$, let $\hat{i}_k$ be the i_{j_k}th location of Π. Append $\hat{i}_k$ to I_{comp}.
   b) If $b_k = 1$, let $\hat{i}_k$ be the j_kth location of $\pi_{R_{\text{out}}}$. Append $\hat{i}_k$ to I_{comp}.

Output $(I_{\text{comp}}, D_{\text{comp}})$.

Figure 1.2: The composite verifier V_{comp} of Ben-Sasson et al. (2006).

$\hat{\tau}$-ROP with parity-check complexity p_{out}, and $\hat{\tau}$-RNL with listing agent runtime of $t_{\text{RNL}} \cdot 2^{r_{\text{in}}} \cdot q_{\text{in}} \cdot t_{\text{in}}$, where

$$\hat{\tau} = \frac{r_{\text{in}} + \tau \cdot r_{\text{out}}}{r_{\text{in}} + r_{\text{out}}}.$$

Furthermore, the shared and aware parts of the randomness are the same.

Remark 1.47. *We stress that the outer verifier's ROP (as well as RNL) is used in showing RNL of the composite verifier. Briefly, this is so that the composite listing agents can emulate the inner verifier so as to obtain its queries, without having access to the outer randomness.*

Proof. First, some names and notation. In the following proof, V_{out}, V_{in} and V_{comp} will be called the outer, inner and composite verifiers (respectively). We affix the terms outer, inner and composite when discussing components of the respective verifiers. For example, the outer randomness refers to the randomness used by the outer verifier. We denote the outer randomness by R_{out}, the inner randomness by R_{in}, and the composite by $\hat{R}$ – this will avoid double-indices when we further partition the outer and composite randomness.

Complexity	V_{out}	V_{in}	V_{comp}
Soundness error	(Robust:) s_{out}	s_{in}	$s_{\text{out}} + s_{\text{in}}$
Proximity parameter	-	δ_{in}	-
Robustness parameter	ρ_{out}	-	-
Randomness	r_{out}	r_{in}	$r_{\text{out}} + r_{\text{in}}$
Query	q_{out}	q_{in}	q_{in}
Decision	d_{out}	d_{in}	d_{in}
Runtime	t_{out}	t_{in}	$t_{\text{out}} + t_{\text{in}}$

Table 1.5: The complexities of V_{out}, V_{in} and V_{comp}. The complexities of each verifier are taken with respect to its input; that is, the complexities of the outer and composite verifier are with respect to n, while those of the inner verifier are with respect to $d_{\text{out}}(n)$. For example, $r_{\text{out}} + r_{\text{in}}$ refers to $r_{\text{out}}(n) + r_{\text{in}}(d_{\text{out}}(n))$.

Soundness. Note that the outer verifier V_{out} on randomness R_{out} queries the outer proof Π at q_{out} many locations I. The verifier then computes p_{out} many parities

$$P_{\text{out}} \coloneqq (C_1(R_{\text{out}}), C_2(R_{\text{out}}), \ldots, C_{p_{\text{out}}}(R_{\text{out}}))$$

and evaluates the circuit $D_{\text{out}}(\Pi_{|I}, P_{\text{out}})$. By the robust soundness property of V_{out} with probability at least $(1 - s_{\text{out}})$ over R_{out}, the string $(\Pi_{|I_{\text{out}}}, P_{\text{out}})$ is at least ρ_{out}-far from satisfying D_{out}. Therefore, with probability at least $(1 - s_{\text{out}})$, the inner verifier V_{in} gets $(D_{\text{out}}, (\Pi_{|I_{\text{out}}}, P_{\text{out}}))$ instance of circuit value problem where $(\Pi_{|I_{\text{out}}}, P)$ is ρ_{out}-far[20] from satisfying D_{out}. Since, the proximity parameter δ_{in} of V_{in} satisfies $\delta_{\text{in}} \leq \rho_{\text{out}}$, V_{in} rejects any such proofs with probability at least $1 - s_{\text{in}}$. Therefore, the composite verifier rejects the proof with probability at least $(1 - s_{\text{out}})(1 - s_{\text{in}})$ and hence the soundness error is at most $1 - (1 - s_{\text{out}})(1 - s_{\text{in}}) \leq s_{\text{out}} + s_{\text{in}}$.

RNL. To show RNL of V_{comp}, we show that there is a partition of the composite randomness, as well as a row neighbor-listing agent $\hat{A}_{\text{row}}$ and column neighbor-listing agent $\hat{A}_{\text{col}}$.

Let the RNL partition of the outer randomness R_{out} be given by

$$R_{\text{out}} = (R_{\text{row}}, R_{\text{col}}, R_{\text{shared.row}}, R_{\text{shared.col}}) \in \{0,1\}^{r_{\text{out}}}.$$

The partition of the composite randomness $\hat{R}$ (which consists of the randomness of both outer and inner verifier) is rather simple: the composite row and column parts are the

[20]The guarantee on the input to the circuit is stronger than just saying that it is ρ_{out}-far from satisfying D_{out}. This is because the inner verifier always gets the *honest* parities $C_i(R_{\text{out}})$ when needed, unlike the proof queries which can be arbitrary in soundness case (See Item 7b in Figure 1.3). However, we do not need this structure in proving the lemma.

Hardwired: Outer verifier V_{out} and inner verifier V_{in}.

Input: Explicit input x, outer proof Π and inner proofs $\{\pi_{R_{\text{out}}} | R_{\text{out}} \in \{0,1\}^{r_{\text{out}}}\}$.

1. Obtain outer verifier circuit D_{out} from $V_{\text{out}}(x)$.[a]
2. Sample $R_{\text{out}} \in \{0,1\}^{r_{\text{out}}}$.
3. Run $V_{\text{out}}(x; R_{\text{out}})$ to obtain $I_{\text{out}} = (i_1, i_2, \ldots, i_{q_{\text{out}}})$ and the parity-checks $P_{\text{out}} = (C_1, \ldots, C_{p_{\text{out}}})$.
4. Sample $R_{\text{in}} \in \{0,1\}^{r_{\text{in}}}$.
5. Run $V_{\text{in}}(D_{\text{out}}, R_{\text{in}})$ to obtain $I_{\text{in}} = ((b_1, j_1), \ldots, (b_{q_{\text{in}}}, j_{q_{\text{in}}}))$ and D_{in}.
6. Initialize $I_{\text{comp}}, P_{\text{comp}} \coloneqq \emptyset$, and $D_{\text{comp}} \coloneqq D_{\text{in}}$
7. For each $k \in [q_{\text{in}}]$, determine the queries of the composite verifier:
   a) If $b_k = 0$ and $j_k \leq i_{q_{\text{out}}}$, let $\hat{i}_k$ be the i_{j_k}th location of Π. Append $\hat{i}_k$ to I_{comp}.
   b) If $b_k = 0$ and $j_k > i_{q_{\text{out}}}$. Append $C_{j_k - q_{\text{out}}}$ to P_{comp}.
   c) If $b_k = 1$, let $\hat{i}_k$ be the j_kth location of $\pi_{R_{\text{out}}}$. Append $\hat{i}_k$ to I_{comp}.

Output $(I_{\text{comp}}, P_{\text{comp}}, D_{\text{comp}})$.

[a] We use 0-ROP of the outer verifier to obtain the circuit before sampling the outer randomness.

Figure 1.3: The composite verifier V_{comp} of Fig. 1.2, adapted to preserve ROP.

same as the outer's, and the shared part is formed of the outer's shared part as well as the entire inner randomness. Formally, partition $R_{\text{in}} =: (R_{\text{in.row}}, R_{\text{in.col}})$ arbitrarily. Then, the composite randomness $\hat{R}$ is partitioned into

$$\begin{aligned}
\hat{R}_{\text{row}} &\coloneqq R_{\text{row}} \\
\hat{R}_{\text{col}} &\coloneqq R_{\text{col}} \\
\hat{R}_{\text{shared.row}} &\coloneqq (R_{\text{shared.row}}, R_{\text{in.row}}) \\
\hat{R}_{\text{shared.col}} &\coloneqq (R_{\text{shared.col}}, R_{\text{in.col}}).
\end{aligned}$$

Indeed, we have

$$\hat{\tau} \coloneqq \frac{|R_{\text{in}}| + |R_{\text{shared.row}}| + |R_{\text{shared.col}}|}{|R_{\text{in}}| + |R|} = \frac{r_{\text{in}} + \tau \cdot r_{\text{out}}}{r_{\text{in}} + r_{\text{out}}}.$$

We construct the row and column agents of V_{comp}, denoted $\hat{A}_{\text{row}}$ and $\hat{A}_{\text{col}}$ (respectively). Fix a configuration $(\hat{R}, k) := (R_{\text{out}}, R_{\text{in}}, k)$ where the outer randomness is R_{out} and the inner randomness is R_{in}. We describe only the row agent $\hat{A}_{\text{row}}$, as the column agent $\hat{A}_{\text{col}}$ is analogous. The outcome depends on whether the query was issued to the outer proof or to one of the inner proofs. (In the notation of Fig. 1.3, this is determined by the value of b_k.)

Case 1 The k-th query is to an inner proof (i.e., $b_k = 1$).

Observe that if a configuration $(R'_{\text{out}}, R'_{\text{in}}, k')$ results in a query to the same location, then it must have the same outer randomness as $\left(\hat{R}, k\right)$; that is, $R'_{\text{out}} = R_{\text{out}}$. If indeed $R_{\text{out}} = R'_{\text{out}}$, checking whether the location queried by the two configurations is the same depends only on the inner randomness and the query index, and these are known given the composite shared randomness. In such a case, $\hat{A}_{\text{row}}$ lists all neighboring configurations by finding all possible (R'_{in}, k') that lead to the same location via "brute-force" enumeration. Thus, in this case, on input $(R_{\text{row}}, R_{\text{shared}}, R_{\text{in}})$, the row agent runs as follows:

1. Initialize a list $\hat{L}_{\text{row}}$.
2. For each $R'_{\text{in}} \in \{0,1\}^{r_{\text{in}}}$ and $k' \in [q_{\text{in}}]$ such that $b_{k'} = 1$:
   a) Check if the inner configuration (R'_{in}, k') results in the same query location as the configuration (R_{in}, k). If so, add $(R_{\text{row}}, R_{\text{shared.row}}, R'_{\text{in}}, k')$ to the list $\hat{L}_{\text{row}}$.
3. Output $\hat{L}_{\text{row}}$.

Case 2 The k-th query is to the outer proof (i.e., $b_k = 0$).

Let $(R'_{\text{out}}, R'_{\text{in}}, k')$ be a configuration that potentially leads to reading the same proof location as $(R_{\text{out}}, R_{\text{in}}, k)$. Denote by $(b'_{k'}, j'_{k'})$ the k'-query of the inner verifier on $(D_{\text{out}}, R'_{\text{in}})$.

In this case, we have that $(R_{\text{out}}, R_{\text{in}}, k)$ and $(R'_{\text{out}}, R'_{\text{in}}, k')$ are neighboring configurations if and only if $b'_{k'} = 0$ and the underlying configurations to V_{out}, namely (R_{out}, j_k) and $(R'_{\text{out}}, j'_{k'})$, are neighboring configurations with respect to V_{out}.

Thus, we start by listing all neighboring configurations of (R_{out}, j_k) with respect to V_{out} in a rectangular and synchronized fashion. Then, we extend each such neighbor (R'_{out}, j') to a list of all configurations $(R'_{\text{out}}, R'_{\text{in}}, k')$ to V_{comp} that read the same location in the outer proof.

Thus, in this case, on input $(R_{\text{row}}, R_{\text{shared}}, R_{\text{in}})$, the row agent runs as follows:

1. Initialize a list $\hat{L}_{\text{row}}$.
2. Compute j_k based on D_{out} and R_{in}.
3. Invoke the outer row agent $A_{\text{row}}(R_{\text{row}}, R_{\text{shared}}, j_k)$ to obtain a list L_{row}.
4. For each $(R'_{\text{row}}, R'_{\text{shared.row}}, j')$ in L_{row}:

a) For each $R'_{\text{in}} \in \{0,1\}^{r_{\text{in}}}$ and $k' \in [q_{\text{in}}]$:
   i. Emulate V_{in} on input D_{out} and randomness R'_{in}. If the k'-th query is to location j' in the input oracle, i.e., $b_{k'} = 0$ and $j_{k'} = j'$, add $(R'_{\text{row}}, R'_{\text{shared.row}}, R'_{\text{in}}, k')$ to $\hat{L}_{\text{row}}$.

5. Output $\hat{L}_{\text{row}}$.

It is simple to verify that these lists are synchronized and canonical, and that the zip $\hat{L}$ of the two lists $\hat{L}_{\text{row}}$ and $\hat{L}_{\text{col}}$ lists all neighboring configurations of $(\hat{R}, k)$.

We show that $\hat{A}_{\text{row}}$ knows the index of the full given configuration $(\hat{R}, k)$ in $\hat{L}_{\text{row}}$ (an analogous statement holds for $\hat{A}_{\text{col}}$). In case 2, observe that running A_{row} on $(R_{\text{row}}, R_{\text{shared}}, j_k)$ gives the unique index of the entry corresponding to (R_{out}, j_k) in L_{row}. Furthermore, from the shared randomness, $\hat{A}_{\text{row}}$ knows (R_{in}, k). Together this identifies uniquely the list entry in $\hat{L}_{\text{row}}$ that corresponds to (R, k). In case 1, this is even simpler, since from the shared randomness $\hat{A}_{\text{row}}$ knows (R_{in}, k) and can thus identify the unique corresponding list entry in $\hat{L}_{\text{row}}$.

The running time of both agents is dominated by the double enumeration in Item 4 of Case 2, which is dominated by

$$t_{\text{RNL}} \cdot 2^{r_{\text{in}}} \cdot q_{\text{in}} \cdot t_{\text{in}}. \tag{1.8}$$

ROP. We prove that the composite verifier has $\hat{\tau}$-ROP, where the aware and shared part of the randomness are the same. Since the outer verifier has 0-ROP, then the description of D_{out} is indeed independent of the randomness. Also, the set of queries $I_{\text{in}} = ((b_1, j_1), \ldots, (b_{q_{\text{in}}}, j_{q_{\text{in}}}))$ and D_{in} depend only on R_{in}, which is a part of the aware randomness. Thus, once the aware randomness is fixed, the query corresponding to (b_c, j_c) where $b_c = 0$ and $j_c > i_{q_{\text{out}}}$, is a fixed parity of the outer randomness R_{out}. Since the aware part of randomness contains the shared outer randomness, the j_c-th input to D_{in} is a fixed affine (rather than linear) function of R_{obliv}.

This shows that the predicate of the composite verifier depends only on the aware randomness, which is the same as the shared randomness. Additionally, the randomness parity checks fed into the predicate are determined by the aware part of the randomness. □

1.7 The final construct: Short, efficient, smooth, and rectangular PCPs

In this section, we obtain our final *short*, *efficient* and *constant-query* PCP for languages in $\mathsf{NTIME}(T(n))$ that is smooth, rectangular and has ROP.

We will use the randomness-efficient, constant-query PCPP of Mie in the inner level of our composition. Recall Definitions 1.16 and 1.17: A pair language $L \subseteq \{0,1\}^* \times \{0,1\}^*$ consists of pairs (x, y), where, in the context of a PCPP, x is given explicitly to the verifier, and y is given implicitly (oracle access). We use $n \coloneqq |x|$ and $K \coloneqq |y|$.

Theorem 1.48 (Mie 2009)**.** *Suppose L is a pair language in $\mathsf{NTIME}(T(n))$ for some non-decreasing function $T(n)$. Then, for every constant $0 < s, \delta < 1$, L has a PCPP verifier with the following parameters:*

- *randomness complexity $r_{\text{in}}(n+K) \coloneqq \log T(n+K) + O(\log\log T(n+K))$.*
- *query complexity $q_{\text{in}}(n+K) \coloneqq O_{s,\delta}(1)$,*
- *verification time $t_{\text{in}}(n, K) \coloneqq \mathsf{poly}(n, \log K, \log T(n+K))$.*
- *soundness error s and proximity parameter δ.*

We can now prove the following main theorem.

Theorem 1.49. *Let L be a language in $\mathsf{NTIME}(T(n))$ for some non-decreasing function $T\colon \mathbb{N} \to \mathbb{N}$. There exists a universal constant c such that for all odd integers $m \in \mathbb{N}$, and any constant $s \in (0, \frac{1}{2})$ satisfying $T(n)^{1/m} \geq m^{cm}/s^6$ and $\min\{1/c \log n, s^3/m^{cm}\} \geq (T(n)^{1/m} \cdot \mathsf{poly}\log T(n))^{-1/2}$, L has a PCP verifier with the following parameters:*

- *Alphabet $\{0,1\}$.*
- *Randomness complexity $r(n) = \log T(n) + O(m \log\log T(n)) + O(\log n)$.*
- *Soundness error s.*
- *Decision, query and parity-check complexities all $O_s(1)$.*
- *Verifier runtime $t(n) = \mathsf{poly}(n, T(n)^{1/m})$.*
- *The verifier has τ-ROP and is τ-rectangular, where*

$$\tau \cdot r(n) = \frac{5}{m} \log T(n) + O(m \log\log T(n)) + O(\log n).$$

Furthermore, the shared and the aware parts of the randomness are the same.

Proof. We start with the robust PCP verifier over the Boolean alphabet for L given by Theorem 1.35, with the following complexities:

- Randomness complexity $r_{\text{out}}(n) = (1 - \frac{1}{m}) \log T(n) + O(m \log\log T(n)) + O(\log(1/s))$.
- Query and Decision complexity $q_{\text{out}}(n) = d_{\text{out}}(n) = T(n)^{1/m} \cdot \mathsf{poly}(\log T(n), 1/s)$.
- Verifier running time $t_{\text{out}}(n) = q_{\text{out}}(n) \cdot \mathsf{poly}(n, \log T(n))$.
- τ_{out}-RNL, with $\tau_{\text{out}} \cdot r_{\text{out}}(n) = \frac{4}{m} \log T(n) + O(m \log\log T(n)) + O(\log(1/s))$, and listing-agents runtime $t_{\text{RNL,out}}(n) = \mathsf{poly}(\log T(n))$.
- Robust soundness error $s/3$ with robustness parameter $\rho = \Theta(s)$.

We next add the τ_{out}-ROP to this PCP verifier using Lemma 1.44. This step increases the decision complexity to $\widetilde{O}(t_{\text{out}}(n))$, verifier's running time to $\widetilde{O}(t_{\text{out}}(n)) + \mathsf{poly}(q_{\text{out}}(n)) = \mathsf{poly}(t_{\text{out}}(n))$, and adds the parity-check complexity $q_{\text{out}}(n)$. It also reduces the robustness parameter by a constant factor to $\Omega(\rho)$.

Next, we wish to compose this (outer) robust PCP with the inner PCPP of Theorem 1.48. Since the decision complexity of the outer verifier is $\widetilde{O}(t_{\text{out}}(n))$ the inner PCPP ought to verify the pair-language $\mathsf{CVP} \in \mathsf{NTIME}(\widetilde{O}(t_{\text{out}}(n)))$, where the length of the implicit input is $K = q_{\text{out}}(n) = \widetilde{O}(t_{\text{out}}(n))$. Therefore, we compose this robust PCP with the PCPP verifier of Mie given in Theorem 1.48, for the pair language $\mathsf{CVP} \in \mathsf{NTIME}(\widetilde{O}(t_{\text{out}}(n)))$, with soundness error $s/3$ and proximity parameter which is greater than the robustness parameter $\Omega(\rho)$ of the outer verifier. The PCPP has query complexity $O_s(1)$, randomness complexity $r_{\text{in}}(\widetilde{O}(t_{\text{out}}(n))) = \frac{1}{m}\log T(n) + O(\log\log T(n)) + O(\log n) + O(\log(1/s))$ and verification time $t_{\text{in}}(\widetilde{O}(t_{\text{out}}(n))) + \mathsf{poly}(t_{\text{out}}(n)) = \mathsf{poly}(t_{\text{out}}(n)) = \mathsf{poly}(n, T(n)^{1/m}, 1/s)$.

By Theorem 1.45 and Lemma 1.46, the composite PCP verifier has soundness error $2s/3$, and query and parity-check complexities both $O_s(1)$. The randomness complexity of the composite verifier, denoted by $r(n)$, is $r_{\text{out}}(n) + r_{\text{in}}(\widetilde{O}(t_{\text{out}}(n)))$. The running time of the composite verifier, denoted $t_{\text{comp}}(n)$, is $\widetilde{O}(t_{\text{out}}(n)) + t_{\text{in}}(\widetilde{O}(t_{\text{out}}(n))) = \mathsf{poly}(t_{\text{out}}(n)) = \mathsf{poly}(n, T(n)^{1/m})$.

The composite verifier has $\hat{\tau}$-RNL and $\hat{\tau}$-ROP where

$$\hat{\tau} = \frac{r_{\text{in}}(\widetilde{O}(t_{\text{out}}(n))) + \tau_{\text{out}} \cdot r_{\text{out}}(n)}{r_{\text{in}}(\widetilde{O}(t_{\text{out}}(n))) + r_{\text{out}}(n)} \leq \tau.$$

Furthermore, the shared and the aware parts of the randomness are indeed the same. The running time of the RNL agents, denoted $t_{\text{RNL}}(n)$, is at most

$$O(t_{\text{RNL,out}}(n) \cdot 2^{r_{\text{in}}(\widetilde{O}(t_{\text{out}}(n)))} \cdot t_{\text{in}}(\widetilde{O}(t_{\text{out}}(n))) \leq \mathsf{poly}(n, T(n)^{1/m}) \ .$$

Finally, we use Theorem 1.31 with $\mu = s/3$ to make the composite verifier smooth and rectangular. This step increases the soundness error by $s/3$ and thus the overall soundness error is s as required. The PCP verifier becomes smooth and is τ-rectangular and has τ-ROP, with the shared and the aware parts of the randomness still the same. This conversion keeps the randomness and parity-check complexities the same. The decision and query complexities remain the same up to constants. Finally, the running time of the verifier is $t(n) = q \cdot \mathsf{poly}(t_{\text{RNL}}(n)) + t_{\text{comp}}(n) = \mathsf{poly}(n, T(n)^{1/m})$, as asserted. □

Remark 1.50. *We remark that the randomness complexity of the composed verifier in Theorem 1.49 is actually only* $\log T(n) + O(m \log\log T(n))$ *for* $T(n) = \Omega(n^m)$. *This improvement is obtained by the improved verifier running time* $t_{\text{out}}(n) = q(n) \cdot \mathsf{polylog} T(n) + O(n) = T(n)^{1/m} \cdot \mathsf{poly}(\log T(n)) + O(n)$ *mentioned in Remark 1.37. Note that when* $T(n) = \Omega(n^m)$, $t_{\text{out}}(n) = T(n)^{1/m} \cdot \mathsf{poly}(\log T(n))$. *Plugging this value of* $t_{\text{out}}(n)$ *in the above proof yields the above randomness complexity. As in the case of Remark 1.37, this improvement is not needed for our construction.*

Chapter 2

Models That Prove Their Own Correctness

Bob is studying for his algebra exam and stumbles upon a question Q that he cannot solve. He queries a Large Language Model (LLM) for the answer, and it responds with a number: 42. Bob is aware of recent research showing that the LLM attains a 90% score on algebra benchmarks (cf. Frieder et al. 2023), but should he trust that the answer to his particular question Q is indeed 42?

Bob could ask the LLM to explain its answer in natural language. Though he must proceed with caution, as the LLM might try to convince him of an incorrect answer (Turpin et al., 2023). Moreover, even if 42 is the correct answer, the LLM may fail to produce a convincing proof (Wang, Yue, and Sun, 2023). If only the LLM could formally prove its answer, Bob would verify the proof and be convinced.

This chapter initiates the study of *Self-Proving models* (Fig. 2.1) that prove the correctness of their answers via an Interactive Proof system (Goldwasser, Micali, and Rackoff, 1985). Self-Proving models successfully convince a verification algorithm V with *worst-case soundness guarantees*: for any question, V rejects all incorrect answers with high probability over the interaction. This guarantee holds even against provers that have access to V's specification, and unbounded computational power.

Our contributions are as follows.

- We define Self-Proving models (Section 2.1).
- We propose two methods for learning Self-Proving models in Section 2.2. The first, *Transcript Learning (TL)*, relies on access to transcripts of accepting interactions and is the focus of this chapter; we prove convergence bounds for TL under convexity and Lipschitzness assumptions. The second method, *Reinforcement Learning from Verifier Feedback (RLVF)*, trains a model by emulating interaction with the verifier. We also present variants of these algorithms that use *Annotations* to improve learning in practice.

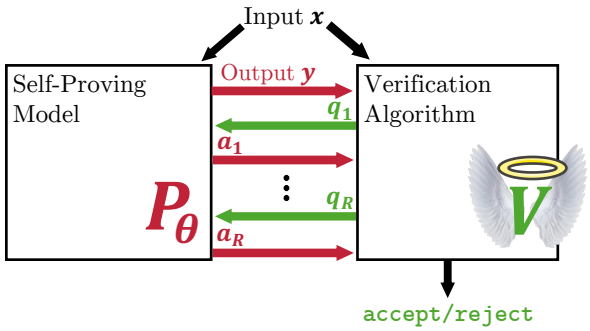


Figure 2.1: **Self-Proving models.** For input x, Self-Proving model P_θ generates an output y and sends it to a Verification Algorithm V. Then, over $i \in [R]$ rounds, V sends query q_i, and receives an answer a_i from P_θ. Finally, V decides ("accept/reject") whether it is convinced that y is a correct output for x.

	Guarantee	Type	Def.
V	Completeness & Soundness	Worst-case $\forall x, y$	2.2
P_θ	Verifiability	Average-case $x \sim \mu,\ y \sim P_\theta(x)$	2.4

Table 2.1: **Formal guarantees.** Completeness and soundness are fundamental guarantees of a verification algorithm V. Verifiability (novel in this work) is a feature of a model P_θ with respect to a verifier V and input distribution μ. Importantly, V's soundness holds for any input x and output y.

- We empirically study TL and Annotated-TL (ATL) for training Self-Proving transformers that compute the Greatest Common Divisor (GCD) of two integers. Table 2.2 demonstrates the efficacy of our methods, with additional experiments in Section 2.3. Our results may be of independent interest for research on the arithmetic capabilities of transformers (e.g. Charton 2024; Lee et al. 2024). Code, data and models are available at `https://github.com/orrp/self-proving-models`.

Scope. This chapter contains a theory of learned models that prove their own correctness via an Interactive Proof system. The fascinating and well-studied question of *which* settings are verifiable in an Interactive Proof system is beyond our scope. Our theory is general in that it pertains to *any* such setting, e.g., any decision problem solvable in polynomial space (Shamir, 1992). See Goldreich (2008c) for a primer on Proof systems more broadly.

2.1 Defining Self-Proving models

We introduce and formally define our learning framework in which models prove the correctness of their output. We start with preliminaries from the learning theory and proof systems literatures in Section 2.1.1. We then introduce our main definition in Section 2.1.2.

Learning method	Correctness (%)	Verifiability (%)
GPT (baseline)	99.8	-
GPT+TL	98.8	60.3
GPT+TL+RLVF	98.9	78.3
GPT+Annotated TL	98.6	96.0

Table 2.2: **Self-Proving transformers computing the GCD.** We train a 6.3M parameter GPT to compute the GCD of two integers sampled log-uniformly from $[10^4]$. Vanilla GPT correctly generates the GCD for almost all inputs, but does not prove correctness to a simple verification algorithm. GPT trained with Transcript Learning (GPT+TL) proves its answer 60.3% of the time; adding Reinforcement Learning from Verifier Feedback (+RLVF) increases this to 78.3%; training with Annotated Transcript Learning (GPT+ATL) gives the highest Verifiability score of 96%. See Section 2.3 for details.

2.1.1 Preliminaries

Let Σ be a finite set of tokens and Σ^* denote the set of finite sequences of such tokens. We consider sequence-to-sequence models $F_\theta \colon \Sigma^* \to \Sigma^*$, which are total functions that produce an output for each possible input sequence. A model is parameterized by a real-valued, finite dimensional vector θ. We consider models as *randomized* functions, meaning that $F_\theta(x)$ is a random variable over Σ^*, of which samples are denoted by $y \sim F_\theta(x)$.

Before we can define models that prove their own correctness, we must first define correctness. Correctness is defined with respect to an input distribution μ over Σ^*, and a ground-truth F^* that defines correct answers. For simplicity of presentation, we focus on the case that each input $x \in \Sigma^*$ has exactly one correct output $F^*(x) \in \Sigma^*$, and a zero-one loss function on outputs (the general case is deferred to Definition 2.9). The fundamental goal of machine learning can be thought of as learning a model of the ground-truth F^*. Formally,

Definition 2.1 (Correctness)**.** *Let μ be a distribution of input sequences in Σ^* and let $F^* \colon \Sigma^* \to \Sigma^*$ be a fixed (deterministic) ground-truth function. For any $\alpha \in [0, 1]$, we say that model F_θ is α-correct (with respect to μ) if*

$$\Pr_{\substack{x \sim \mu \\ y \sim F_\theta(x)}} [y = F^*(x)] \geq \alpha.$$

An *interactive proof system* (Goldwasser, Micali, and Rackoff, 1985) is a protocol carried out between an efficient *verifier* and a computationally unbounded *prover*. The prover attempts to convince the verifier of the correctness of some assertion, while the verifier accepts only correct claims. The prover is powerful yet untrusted; in spite of this, the verifier must reject false claims with high probability.

In the context of this work, it is important to note that the verifier is *manually-defined* (as opposed to learned). Formally, the verifier is a probabilistic polynomial-time algorithm tailored to a particular ground-truth capability F^*. Informally, the verifier is the anchor of trust: think of the verifier as an efficient and simple algorithm, hosted in a trustworthy environment.

Given an input $x \in \Sigma^*$, the model F_θ "claims" that $y \sim F_\theta(x)$ is correct. We now define what it means to *prove* this claim. We will use P_θ to denote Self-Proving models, noting that they are formally the same object[1] as non-Self-Proving ("vanilla") models F_θ. This notational change is to emphasize that P_θ first outputs $y \sim P_\theta(x)$ *and is then prompted by the verifier*, unlike F_θ who only generates an output $y \sim F_\theta(x)$.

A Self-Proving model proves that $y \sim P_\theta(x)$ is correct to a verifier V over the course of R rounds of interaction (Figure 2.1). In each round $i \in [R]$, verifier V queries P_θ on a sequence $q_i \in \Sigma^*$ to obtain an answer $a_i \in \Sigma^*$; once the interaction is over, V accepts or rejects. For fixed $x, y \in \Sigma^*$, the decision of V after interacting with P_θ is a random variable over V's decision (accept/reject), determined by the randomness of V and P_θ. The decision random variable is denoted by $\langle V, P_\theta \rangle (x, y)$.

We present a definition of Interactive Proofs restricted to our setting.

Definition 2.2. *Fix a soundness error $s \in (0, 1)$, a finite set of tokens Σ and a ground-truth $F^* \colon \Sigma^* \to \Sigma^*$. A* verifier V (in an Interactive Proof) for F^* *is a probabilistic polynomial-time algorithm that is given explicit inputs $x, y \in \Sigma^*$ and black-box (oracle) query access to a prover P.*[2] *It* interacts *with P over R rounds (see Figure 2.1) and outputs a decision $\langle V, P \rangle (x, y) \in \{\text{reject}, \text{accept}\}$. Verifier V satisfies the following two guarantees:*

- Completeness: *There exists an* honest prover P^* *such that, for all $x \in \Sigma^*$,*

$$\Pr[\langle V, P^* \rangle (x, F^*(x)) \ \text{ accepts }] = 1,$$

  where the probability is over the randomness of V.[3]

- Soundness: *For all P and for all $x, y \in \Sigma^*$, if $y \neq F^*(x)$ then*

$$\Pr[\langle V, P \rangle (x, y) \ \text{ accepts }] \leq s,$$

  where the probability is over the randomness of V and P, and s is the soundness error.

The efficiency of an interactive proof is usually measured with respect to four parameters: the round complexity R, the communication complexity (the overall number of bits transferred during the interaction), P^*'s efficiency and V's efficiency. These complexity measures scale with the computational complexity of computing the ground-truth F^*. For example, an interactive proof for a complex F^* may require multiple rounds of interaction.

[1] Both are randomized mappings from Σ^* to Σ^*.

[2] We intentionally write P rather than P_θ: Interactive Proofs are defined with respect to all possible provers, not just parameterized ones.

[3] WLOG, the honest prover is deterministic by fixing the optimal randomness of a randomized prover.

Remark 2.3 (Verifier efficiency)**.** *Definition 2.2 requires that V is a polynomial-time algorithm whereas provers are unbounded. This captures a requirement for* efficient verification*. We chose polynomial time as a measure of efficiency because it is common Proof systems literature. That said, one could adapt Definition 2.2 to fit alternative efficiency measures, such as space complexity (Condon and Lipton, 1989) or circuit depth (Goldwasser et al., 2007). Regardless of which measure is taken, to avoid a trivial definition it is crucial that V should be more efficient than the honest prover P^*; else, V can simply execute P^* to perform the computation itself.*

By definition, the soundness error s of a verifier V bounds the probability that it is mistakenly convinced of an incorrect output; in that sense, the smaller s, the "better" the verifier V. In our setting, we think of a manually-defined verifier V who is formally proven (by a human) to have a small soundness error by analysis of V's specification.

As depicted in Figure 2.1, each of the model's answers depends on all previous queries and answers in the interaction. This captures the setting of *stateful models*, e.g. a session with a chatbot.

Towards defining Self-Proving models (Section 2.1.2), let us observe the following. Completeness and soundness are *worst-case guarantees*, meaning that they hold for all possible inputs $x \in \Sigma^*$. In particular, completeness implies that for all $x \in \Sigma^*$, the honest prover P^* convinces V of the correctness of $F^*(x)$; in classical proof systems there is no guarantee that an "almost honest" prover can convince the verifier (cf. Paradise 2021b). Yet, if we are to *learn* a prover P_θ, we cannot expect it to agree with P^* perfectly, nor can we expect it to always output $F^*(x)$. Indeed, Self-Proving models will have a *distributional guarantee* with respect to inputs $x \sim \mu$.

2.1.2 The Definition

We define the *Verifiability* of a model P_θ with respect to an input distribution μ and a verifier V. Intuitively, Verifiability captures the ability of the model to prove the correctness of its answer $y \sim P_\theta(x)$, when the input x is sampled from μ. We refer to models capable of proving their own correctness as *Self-Proving models*. Notice that, as in Definition 2.2, the verifier is fixed and agnostic to the choice of the Self-Proving model.

Definition 2.4 (Self-Proving model)**.** *Fix a verifier V for a ground-truth $F^*\colon \Sigma^* \to \Sigma^*$ as in Definition 2.2, and a distribution μ over inputs Σ^*. The* Verifiability *of a model $P_\theta\colon \Sigma^* \to \Sigma^*$ is defined as*

$$\mathrm{ver}_{V,\mu}(\theta) \coloneqq \Pr_{\substack{x\sim\mu \\ y\sim P_\theta(x)}} \left[\langle V, P_\theta\rangle (x, y) \ \text{ accepts } \right]. \tag{2.1}$$

*We say that model P_θ is β-*Self-Proving *with respect to V and μ if $\mathrm{ver}_{V,\mu}(\theta) \geq \beta$.*

Remark 2.5 (Verifiability $\implies$ correctness)**.** *Notice that the ground-truth F^* does not appear in Definition 2.4 except for the first sentence. Indeed, once it is established that V is a verifier for F^* (as per Definition 2.2), then Verifiability w.r.t V implies correctness*

w.r.t F^*: Consider any input distribution μ, ground-truth F^*, and a verifier V for F^* with soundness error s. By a union bound, if model P_θ is β-Verifiable, then it is $(\beta - s)$-correct. That is to say, Verifiability is formally a stronger guarantee than correctness when V has small soundness error s.

As depicted in Figure 2.1, a Self-Proving model P_θ plays a dual role: first, it generates an output $y \sim P_\theta(x)$, and then it proves the correctness of this output to V. Note also that Self-Provability is a feature of a *model*, unlike completeness and soundness which are features of a *verifier* (see Table 2.1).

The benefit of Verifiability over correctness is captured by the following scenario. Alice wishes to use a model P_θ to compute some functionality F^* on an input x_0 in a high risk setting. Alice generates $y_0 \sim P_\theta(x_0)$. Should Alice trust that y_0 is correct? If Alice has a held-out set of labeled samples, she can estimate P_θ's average correctness on μ. Unfortunately, (average) correctness provides no guarantee regarding the correctness of the particular (x_0, y_0) that Alice has in hand. If, however, Alice has access to a verifier V for which P_θ is Self-Proving, then she can trust the model on an input-by-input (rather than average-case) basis: Alice can execute V on (x_0, y_0) and black-box access to P_θ. Soundness of V guarantees that if y_0 is incorrect, then V rejects with high probability, in which case Alice should either generate $P_\theta(x_0)$ again—or find a better model.

2.1.3 A More General Definition

We present variants of Self-Proving models (Definition 2.4) generalized to one-to-many relations, and general bounded loss functions. While these generalizations provide a richer framework that may accommodate a wider range of applications, the remainder of this chapter focuses on the forgoing Definition 2.4, which captures the essential properties while remaining mathematically manageable. Readers primarily interested in this chapter's main results may proceed directly to Section 2.2. Those interested in extending this work may find the generalizations presented here valuable for future research directions.

General (bounded) loss functions. In Definition 2.1 we implicitly use the 0-1 loss when measuring the correctness of a model: For any $x \in X$, we measure only whether the model generated the correct output $y = F^*(x)$, but not how "far" the generated y was from $F^*(x)$. It is often the case in machine learning that we would be satisfied with models that generate a "nearly-correct" output. This is formalized by specifying a loss function $\ell \colon \Sigma^* \times \Sigma^* \to [0, 1]$ and measuring the probability that $\ell(x, y)$ is smaller than some threshold $\lambda \in [0, 1)$, where x is drawn from the input distribution μ, and y is generated by the model when given input x.

In the context of language modeling, different loss function allow for a more fine-grained treatment of the *semantics* of a given task. As an example, consider the *prime-counting task*:

- Given an integer $x < 10^9$, output the number of primes less than or equal to x.

In the notation of Section 2.1, the prime-counting task would be captured by the ground-truth function

$$F^*(x) \coloneqq |\{p \in \mathbb{N} \mid p \leq x,\ p \text{ is prime}\}|\,.^4$$

Per Definition 2.1, any output other than $F^*(x)$ is "just as incorrect" as any other. Yet, we might prefer outputs that are closer to the correct answer, say, in L_1 norm. This preference can be captured by the following bounded loss function

$$\ell_1(x, y) \coloneqq \begin{cases} |y - F^*(x)| \cdot 10^{-9} & \text{if } y \leq 10^9 \\ 1 & \text{else.} \end{cases}$$

In particular, if we are interested in knowing the answer only up to some additive constant C, we could say that an output y is "correct-enough" if $\ell_1(x, y) \leq C \cdot 10^{-9}$.

More generally, we relax Definition 2.1 to capture approximate correctness as follows.

Definition 2.6 (Approximate correctness)**.** *Let μ be a distribution over input sequences in Σ^* and let $\ell\colon \Sigma^* \times \Sigma^* \to [0, 1]$ be a loss function. For any $\alpha, \lambda \in [0, 1]$, we say that model F_θ is (α, λ)-correct with respect to μ if*

$$\Pr_{\substack{x \sim \mu \\ y \sim F_\theta(x)}} [\ell(x, y) \leq \lambda] \geq \alpha.$$

One-to-many-relations. In Section 2.1, we focused on the setting of models of a ground-truth function $F^*\colon \Sigma^* \to \Sigma^*$. That is, when each input x has exactly one correct output, namely $F^*(x)$. A more general setting would be to consider a ground-truth *relation* $L \subseteq \Sigma^* \times \Sigma^*$. Then, we say that y is a correct output for x if $(x, y) \in L$. Importantly, this allows a single x to have many possible correct outputs, or none at all.

Note that we must take care to choose a loss function ℓ that captures correctness with respect to the relation L, i.e., $\ell(x, y) = 0$ if and only if $(x, y) \in L$. Equivalently, any loss function ℓ induces a relation $L \coloneqq \{(x, y) \mid \ell(x, y) = 0\}$. Therefore, our relaxation to approximate-correctness Definition 2.6 already captures the setting of one-to-many relations, since an input x may have multiple y^* such that $\ell(x, y^*) = 0$.

The general definition. We first present a relaxed definition of Interactive Proof systems for verifying approximate-correctness.

Definition 2.7 (Definition 2.2, generalized)**.** *Fix a soundness error $s \in (0, 1)$, a threshold $\lambda \in [0, 1)$, a finite set of tokens Σ, and a loss function $\ell\colon \Sigma^* \times \Sigma^* \to [0, 1]$. A* verifier *$V$ for ℓ with threshold λ is a probabilistic polynomial-time algorithm that is given explicit inputs $x, y \in \Sigma^*$ and black-box (oracle) query access to a prover P. It* interacts *with P over R rounds (see Figure 2.1) and outputs a decision $\langle V, P\rangle(x, y) \in \{\text{reject}, \text{accept}\}$. Verifier V satisfies the following two guarantees:*

[4] Formally, the input and output are strings in Σ^* representing integers (e.g. in decimal representation). See Section 2.3.6 for a concrete instantiation used in our experiments.

- Completeness: *There exists an* honest prover P^* *such that, for all* $x, y \in \Sigma^*$*, if* $\ell(x, y) = 0$ *then*

$$\Pr[\langle V, P^* \rangle (x, y) \quad \text{accepts }] = 1,$$

  where the probability is over the randomness of V*.*

- Soundness: *For all* P *and for all* $x, y \in \Sigma^*$*, if* $\ell(x, y) > \lambda$ *then*

$$\Pr[\langle V, P \rangle (x, y) \quad \text{accepts }] \leq s,$$

  where the probability is over the randomness of V *and* P*, and* s *is the soundness error.*

Indeed, for a given ground-truth function $F^* \colon \Sigma^* \to \Sigma^*$, Definition 2.2 can be recovered by choosing the 0-1 loss

$$\ell_{F^*}(x, y) \coloneqq \begin{cases} 1 & \text{if } x \neq F^*(y) \\ 0 & \text{else.} \end{cases}$$

and any threshold $\lambda \in [0, 1)$.

Remark 2.8 (Connection to Interactive Proofs of Proximity)**.** *Definition 2.7 can be seen as a slight generalization of (perfect completeness) Interactive Proofs of Proximity (IPPs, Rothblum, Vadhan, and Wigderson 2013). An IPP for a relation* $L \subseteq \Sigma^* \times \Sigma^*$ *with proximity parameter* λ *is obtained by instantiating Definition 2.7 with the loss function* ℓ_{Hamming} *defined by*

$$\ell_{\text{Hamming}}(x, y) \coloneqq \min \left\{ \frac{\#\{i \mid y_i \neq y_i^*\}}{|y|} \;\middle|\; (x, y^*) \in L, \; |y^*| = |y| \right\},$$

that is, $\ell_{\text{Hamming}}(x, y)$ *is the fraction of tokens in* y *that must be changed so as obtain an output* y^* *with* $(x, y^*) \in L$*. However, the motivation of Rothblum, Vadhan, and Wigderson (2013) was studying sublinear time verification, whereas ours is to relax the requirements of traditional Interactive Proofs towards meeting common desiderata in machine learning.*

With this relaxed notion of Interactive Proofs in hand, we are now ready to define Self-Proving models for general (bounded) loss functions.

Definition 2.9 (Definition 2.4, generalized)**.** *Fix a loss function* $\ell \colon \Sigma^* \times \Sigma^* \to [0, 1]$*, a verifier* V *for* ℓ *with threshold* $\lambda \in [0, 1)$ *as in Definition 2.7, and a distribution* μ *over inputs* Σ^**. The* Verifiability *of a model* $P_\theta \coloneqq \Sigma^* \to \Sigma^*$ *is defined as*

$$\text{ver}_{V,\mu}(\theta) \coloneqq \Pr_{\substack{x \sim \mu \\ y \sim P_\theta(x)}} [\langle V, P_\theta \rangle (x, y) \quad \text{accepts }].$$

We say that model P_θ *is* β-Self-Proving *with respect to* V *and* μ *if* $\text{ver}_{V,\mu}(\theta) \geq \beta$*.*

Analogously to Remark 2.5, we observe that Verifiability (Definition 2.9) implies approximate-correctness: Suppose P_θ is β-Self-Proving model with respect to a verifier V that has soundness error s and threshold parameter λ for loss function ℓ. Then by a union bound,

$$\Pr_{\substack{x\sim\mu \\ y\sim P_\theta(x)}} [\ell(x, y) \leq \lambda] \geq \beta - s.$$

Importantly, as emphasized throughout this chapter, soundness of V implies that for *all* inputs x, any output y such that $\ell(x, y) > \lambda$ is rejected with high probability $(1 - s)$.

2.2 Learning Self-Proving autoregressive models

With a sound verifier V at hand, obtaining Self-Proving models with respect to V holds great promise: a user that prompts the model with input x does not need to take it on good faith that $P_\theta(x)$ is correct; she may simply verify this herself by executing the verification protocol. How, then, can we learn models that are not just approximately-correct, but Self-Proving as well?

The challenge is to align the model with a verifier. We assume that the learner has access to input samples $x \sim \mu$ and correct outputs $F^*(x)$, as well as the verifier specification (code). Additionally, the learner can emulate the verifier, as the latter is computationally efficient (Remark 2.3).

Our focus is on autoregressive sequence-to-sequence (Self-Proving) models P_θ. Such models generate their output by recursively prompting a randomized sampling from a base distribution p_θ over tokens Σ. For an input $z \in \Sigma^*$, the output $w \sim P_\theta(z)$ is generated as follows:

- Sample $w_1 \sim p_\theta(z)$.
- Let $j = 1$. While w_j is not the end-of-sequence token $\texttt{EOS} \in \Sigma$:
  - Sample $w_{j+1} \sim p_\theta(z w_1 \cdots w_j)$.
  - Update $j \coloneqq j + 1$.
- Output $w = w_1 w_2 \cdots w_j$.

For any $z \in \Sigma^*$, it is useful to consider the vector of log-probabilities over Σ, denoted by $\log p_\theta(z) \in \mathsf{R}^{|\Sigma|}$. We assume that each coordinate in this vector is differentiable with respect to θ.

Our general approach is inspired by Reinforcement Learning from Human Feedback (Christiano et al., 2017), a method for aligning models with human preferences, which has recently been used to align sequence-to-sequence models (Ouyang et al., 2022). However, there are two important differences between humans and algorithmic verifiers: (1) Verifiers are efficient algorithms which may be emulated by the learner. This is unlike humans, whose

preferences are costly to obtain. On the other hand, (2) verifiers make a single-bit decision at the end of an interaction, but cannot guide the prover (model) in intermediate rounds. In RL terms, this is known as the *exploration problem* for sparse reward signals (e.g. Ladosz et al. 2022).

Section 2.2.1 introduces *Transcript Learning* (TL), a learning algorithm that overcomes the exploration problem mentioned in the second point under the assumption that the learner has access to transcripts of interactions in which the verifier accepts. We prove convergence bounds, and then (Section 2.3) evaluate it through experiments.

Access to accepting transcripts is a reasonable assumption, for example, when there is an efficient honest prover that can generate such transcripts (Goldwasser, Kalai, and Rothblum, 2015). When there is no access to accepting transcripts, we propose *Reinforcement Learning from Verifier Feedback* (Section 2.2.2).

Specification of the learning model. We must first fully specify the theoretical framework in which our results reside. Continuing from Section 2.1, we define a *learner* as an algorithm Λ with access to a family of autoregressive models $\{P_\theta\}_\theta$ and samples from the input distribution $x \sim \mu$. In our setting of Self-Proving models (and in consistence with the Interactive Proofs literature), we give the learner the full specification of the verifier V. More formally,

Definition 2.10 (Self-Proving model learner)**.** *A* (Self-Proving model) learner *is a probabilistic oracle Turing Machine Λ with the following access:*

- *A family of* autoregressive models $\{P_\theta\}_{\theta\in\mathbb{R}^d}$ *where $d \in \mathbb{N}$ is the number of parameters in the family. Recall (Section 2.2) that for each θ and $z \in \Sigma^*$, the random variable $P_\theta(z)$ is determined by the logits $\log p_\theta(z) \in \mathbb{R}^{|\Sigma|}$. For any $z \in \Sigma^*$ and $\sigma \in \Sigma$, the learner Λ can compute the gradient of the σ^{th} logit, that is, $\nabla_\theta \log \Pr_{\sigma'\sim p_\theta(z)}[\sigma = \sigma']$. In particular, $\log \Pr_{\sigma'\sim p_\theta(z)}[\sigma = \sigma']$ is always differentiable in θ.*
- *Sample access to the* input distribution μ*. That is, Λ can sample $x \sim \mu$.*
- *The full specification of the verifier V, i.e., the ability to emulate the verification algorithm V. More specifically, Λ is able to compute V's decision after any given interaction; that is, given input x, output y, and a sequence of queries and answers $(q_i, a_i)_{i=1}^R$, the learner Λ can compute the decision of V after this interaction.*

Throughout this section, we will refer to the *transcript* of an interaction between a verifier and a prover (see Figure 2.1). We will denote this transcript by $\pi = (y, q_1, a_1, \ldots, q_R, a_R)$, and for any index $s \in [|\pi|]$ we will write $\pi_{<s} \in \Sigma^{s-1}$ to denote the s-long prefix of π.

2.2.1 Transcript Learning

We present an algorithm for learning Self-Proving models which uses access to a distribution of accepting transcripts. This is a reasonable assumption to make when the honest prover

P^* (see Definition 2.2) is efficient, as in the case of public-coin Doubly-Efficient Interactive Proof systems as defined by Goldwasser, Kalai, and Rothblum (2015) and developed in other theoretical (e.g. Goldreich and Rothblum 2018) and applied (e.g. Zhang et al. 2021) works. In this case, an honest prover P^* can be run by the learner during training to collect accepting transcripts without incurring heavy computational cost. Alternatively, the learner may collect a dataset of accepting transcripts prior to learning (see Figure 2.2).

The intuition behind Transcript Learning is that the interaction of the verifier and prover can be viewed as a sequence itself, which is called the *transcript* $\pi \in \Sigma^*$. The idea is to learn a model not just of $x \mapsto y^*$ for a correct output y^*, but of $x \mapsto y^*\pi^*$, where π^* is a transcript of an interaction in which the verifier accepted.

Transcript Learning assumes access to a *transcript generator*—a random variable over transcripts that faithfully represents the interaction of the verifier with some prover for a given input. An *honest transcript generator* is one who is fully supported on transcripts accepted by the verifier. Formally,

Definition 2.11 (Transcript generator)**.** *Fix a verifier V in a proof system of $R \in \mathbb{N}$ rounds. A transcript generator $\mathcal{T}_V$ for V is a randomized mapping from inputs $x \in \Sigma^*$ to transcripts $\pi = (y, q_1, a_1, \ldots, q_R, a_R) \in \Sigma^*$. For any input x, $\mathcal{T}_V(x)$ satisfies that for each $r \leq R$, the marginal of $\mathcal{T}_V(x)$ on the r^{th} query q_r agrees with the corresponding marginal of the query generator $(V_q)_r$.*[5]

A transcript generator $\mathcal{T}_V^* \coloneqq \mathcal{T}_V$ is honest *if it is fully supported on transcripts π^* for which the verifier accepts.*

Notice that for any verifier V, there is a one-to-one correspondence between transcript generators and (possibly randomized) provers. We intentionally chose *not* to specify a prover in Definition 2.11 to emphasize that transcripts can be "collected" independently of the honest prover (see completeness in Definition 2.2), and in fact can be collected "in advance" prior to learning (see Figure 2.2). As long as the generator is fully supported on honest transcripts, it can be used for Transcript Learning (Algorithm 1 described next).

TL trains a Self-Proving model by autoregressively optimizing towards generating accepting transcripts. At a very high level, it works by repeatedly sampling $x \sim \mu$ and transcript $y^*\pi^* \sim \mathcal{T}^*(x)$, and updating the logits $\log p_\theta$ towards agreeing with $y^*\pi^*$ via Gradient Ascent. We prove that, under certain conditions, it is expected to output a Self-Proving model.

Theorem 2.12. *Fix a verifier V, an input distribution μ, an autoregressive model family $\{P_\theta\}_{\theta \in \mathbb{R}^d}$, and a norm $||\cdot||$ on $\mathbb{R}^d$. Fix an honest transcript generator $\mathcal{T}_V^*$, and assume that*

[5] A *query generator* V_q corresponding to V takes as input a partial interaction and samples from the distribution over next queries by V. Formally, for any $r \leq R$, given input x, output y, and partial interaction $(q_i, a_i)_{i=1}^r$, $V_q(x, y, q_1, a_1, \ldots, q_r, a_r)$ is a random variable over Σ^{L_q}. For completeness' sake, we can say that when prompted with any sequence z that does not encode an interaction, $V_q(z)$ is fully supported on a dummy sequence $\perp \cdots \perp \in \Sigma^{L_q}$.

Algorithm 1: Transcript Learning (TL)

Hyperparameters: Learning rate $\lambda \in (0, 1)$ and number of samples $N \in \mathbb{N}$.

Input: An autoregressive model family $\{P_\theta\}_{\theta \in \mathbb{R}^d}$, verifier specification (code) V, and sample access to an input distribution μ and an accepting transcript generator $\mathcal{T}^*_V(\cdot)$.

Output: A vector of parameters $\bar{\theta} \in \mathbb{R}^d$.

Initialize $\theta_0 \coloneqq \vec{0}$.

for $i = 0, \ldots, N-1$ **do**

  Sample $x \sim \mu$ and $\pi^* = (y^*, q_1^*, a_1^*, \ldots, q_R^*, a_R^*) \sim \mathcal{T}^*_V(x)$. Denote $a_0 \coloneqq y^*$.

  foreach *Round of interaction* $r = 0, \ldots, R$ **do**

    Let $S(r)$ denote the indices of the r^{th} answer a_r in π^*, and let $\pi_{<s}$ denote the prefix of the partial transcript $(y, q_1^*, a_1^*, \ldots, q_r^*)$.

    for $s \in S(r)$ **do**

      Compute `# Forwards and backwards pass`

$$\alpha_s(\theta_i) \coloneqq \Pr_{\sigma \sim p_{\theta_i}(x\pi_{<s})}[\sigma = \pi_s^*]$$

$$\vec{d}_s(\theta_i) \coloneqq \nabla_\theta \log \alpha_s(\theta_i) = \nabla_\theta \log \Pr_{\sigma \sim p_{\theta_i}(x\pi_{<s})}[\sigma = \pi_s^*].$$

  Update

$$\theta_{i+1} \coloneqq \theta_i + \lambda \cdot \prod_{\substack{r \in [R] \cup \{0\} \\ s \in S(r)}} \alpha_s(\theta_i) \cdot \sum_{\substack{r \in [R] \cup \{0\} \\ s \in S(r)}} \vec{d}_s(\theta_i).$$

Output $\bar{\theta} \coloneqq \frac{1}{N} \sum_{i \in [N]} \theta_i$.

the agreement function

$$A(\theta) \coloneqq \Pr_{\substack{x \sim \mu \\ \pi^* \sim \mathcal{T}^*_V(x) \\ \pi \sim \mathcal{T}^\theta_V(x)}} [\pi = \pi^*]$$

is concave in θ, where the verifier queries are the same in π^* and π. For any $\varepsilon > 0$, let B_{Norm}, B_{Lip} and C be upper-bounds such that the following conditions hold.

- *There exists $\theta^* \in \mathsf{R}^d$ with $||\theta^*|| < B_{\text{Norm}}$ such that $A(\theta^*) \geq 1 - \varepsilon/2$.*
- *For all θ, the logits of P_θ are B_{Lip}-Lipschitz in θ. That is,*

$$\sup_{\substack{\theta \in \mathsf{R}^d \\ z \in \Sigma^*}} ||\nabla_\theta \log p_\theta(z)|| \leq B_{\text{Lip}}.$$

- *In the proof system defined by V, the total number of tokens (over all rounds) is at most C.*

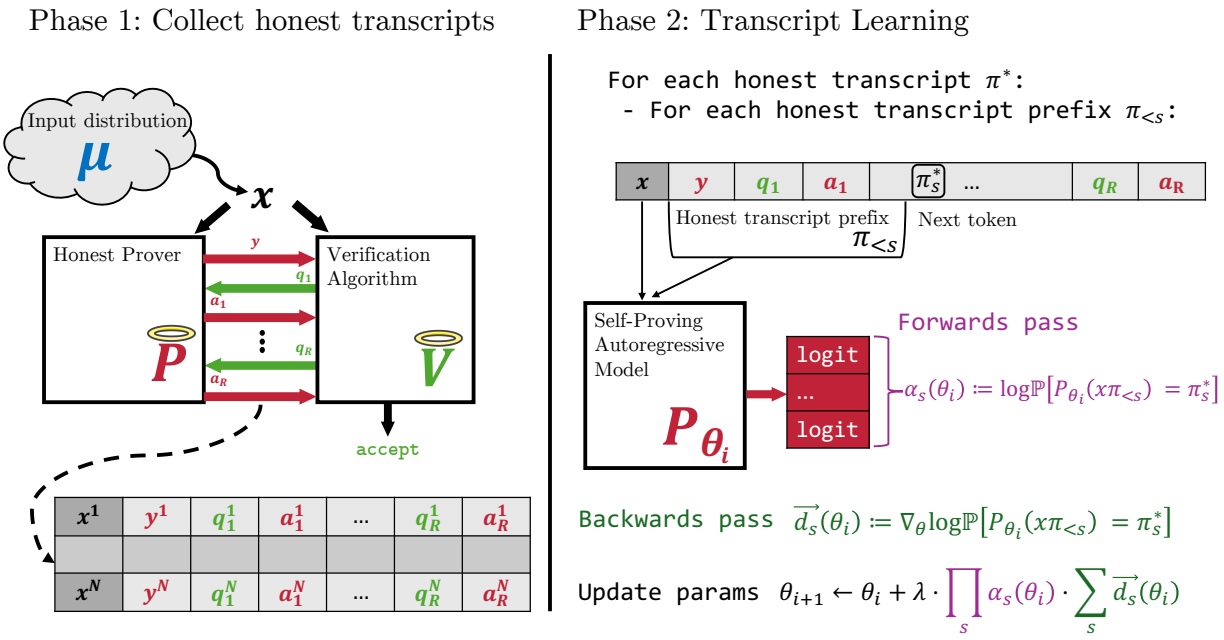


Figure 2.2: **Transcript Learning, visualized.** To understand Algorithm 1, consider the above visualization. In Phase 1, N honest transcripts are collected by letting an Honest Prover interact with the Verification Algorithm; these will be the samples from the honest transcript generator $\mathcal{T}_V^*(x)$. Phase 2 describes the execution of Algorithm 1 itself: For each honest transcript π^* (lines 2-3), and for each prefix π_s of this transcript (lines 4-6), the $\alpha_s(\theta_i)$ and $\vec{d}_s(\theta_i)$ are computed via forwards and backwards passes, respectively (line 7). After iterating through all prefixes, the parameters θ_i are updated (line 8).

Denote by $\bar{\theta}$ the output of TL running for number of iterations N where

$$N \geq 4 \cdot C^2 \cdot \frac{B_{\text{Norm}}^2 \cdot B_{\text{Lip}}^2}{\varepsilon^2} \tag{2.2}$$

and learning rate $\lambda = B_{\text{Norm}}/CB_{\text{Lip}}\sqrt{N}$. Then the expected Verifiability (over the randomness of the samples collected by TL) of $\bar{\theta}$ is at least $1-\varepsilon$. That is,

$$\mathop{\mathbb{E}}_{\bar{\theta}}[\text{ver}_{V,\mu}(\bar{\theta})] \geq 1 - \varepsilon.$$

The conditions for Theorem 2.12 can be split into two. First, the standard conditions

used to prove SGD convergence: convexity,[6] B_{Norm}-boundedness, and B_{Lip}-Lipschitzness. Second, there is a bound C on the *communication complexity* of the prover in the Interactive Proof system.

Quantitatively, the efficiency of TL is captured by the *number of iterations* N. It is desirable to minimize N, which is also the *number of samples* needed from the distribution μ and the transcript generator $\mathcal{T}^*$. The bound on N in Equation (2.2) can be decomposed into the complexity of SGD ($B^2_{\text{Norm}} B^2_{\text{Lip}}/\varepsilon^2$), and communication complexity of the proof system $O(C^2)$. Minimizing communication complexity has been an overarching goal in the study of proof systems (e.g. Goldreich and Håstad 1998; Goldreich, Vadhan, and Wigderson 2002; Reingold, Rothblum, and Rothblum 2021). Theorem 2.12 formally shows the benefit of communication-efficient proof systems in the context of Self-Proving models.

The proof of Theorem 2.12 goes by reduction to Stochastic Gradient Descent (SGD). We show (Lemma 2.13) that the learner can use its only available tools—sampling honest transcripts, emulating the verifier, and differentiating the logits—to optimize the agreement $A(\theta)$. Specifically, this is done by accumulating gradients from the cross-entropy loss computed at each token. Since $A(\theta)$ lower bounds the Verifiability of P_θ, the former can be used as a surrogate for the latter.

Essentially, we are tasked with proving that TL estimates a surrogate of the Verifiability-gradient of its model P_θ. More precisely, TL estimates the gradient of a function that bounds the Verifiability from below. Maximizing this function therefore maximizes the Verifiability.

The lower-bounding function is the agreement of the answers generated by P_θ with the answers provided by the honest transcript generator $\mathcal{T}^*_V$. More formally, we let $\mathcal{T}^\theta_V$ denote the transcript generator induced by the model P_θ when interacting with V: for each x, $\mathcal{T}^\theta_V(x)$ is the distribution over transcripts of interactions between V and P_θ on input x. We stress that $\pi^* \sim \mathcal{T}^*_V(x)$ and $\pi \sim \mathcal{T}^\theta_V(x)$ are transcripts produced when interacting *with the same verifier queries*; we can think of the verifier as simultaneously interacting with the honest prover and with the model P_θ.[7] In what follows, we use $\pi^* \sim \mathcal{T}^*_V(x)$ and $\pi \sim \mathcal{T}^\theta_V(x)$ to denote two transcripts that share the same queries. That is, taking $\pi^* = (y^*, q^*_1, a^*_1, \ldots, q^*_R, a^*_R)$ to denote an accepting transcript sampled from $\mathcal{T}^*_V(x)$, and $\pi = (y, q^*_1, a_1, \ldots, q^*_R, a_R)$ to denote a random transcript sampled from $\mathcal{T}^\theta_V(x)$, we say that π and π^* *agree* if they agree on the prover answers, namely if:

$$(y, a_1, \ldots, a_R) = (y^*, a^*_1, \ldots, a^*_R).$$

This definition implicitly uses the independence of the verifier and model's randomness. We first prove that TL correctly estimates the gradient of $A(\theta)$ in its update step.

[6] Convexity does not hold in general LLM training. Yet, Theorem 2.12 provides useful theoretical analysis in a simplified setting, which we empirically validate in the non-convex setting in Section 2.3.

[7] The way it is presented in the algorithm (and implemented in the experiments), first the verifier is called by $\mathcal{T}^*_V$ and outputs queries $(q^*_1, \ldots q^*_R)$, and then the model is prompted with the verifier queries one a time. This maintains soundness, since a proof system is sound as long as the prover does not know the verifier's queries in advance.

Lemma 2.13 (TL gradient estimation)**.** *Fix an input distribution μ over Σ^* and a verifier V with round complexity R and answer length L_a. Fix an honest transcript generator $\mathcal{T}_V^*$. Let θ be the parameters of a model P_θ and let*

$$A(\theta) \coloneqq \Pr_{\substack{x\sim\mu \\ \pi^*\sim\mathcal{T}_V^*(x) \\ \pi\sim\mathcal{T}_V^\theta(x)}} [\pi = \pi^*] .$$

Then,

$$\nabla A(\theta) = \mathop{\mathbb{E}}_{\substack{x\sim\mu \\ \pi^*\sim\mathcal{T}_V^*}} \left[\prod_{\substack{r\in[R]\cup\{0\} \\ s\in S(r)}} \alpha_s(\theta) \cdot \sum_{\substack{r\in[R]\cup\{0\} \\ s\in S(r)}} \vec{d}_s(\theta) \right]$$

where $S(r)$, $\alpha_s(\theta)$ and $\vec{d}_s(\theta)$ are as defined in Algorithm 1.

Note that Lemma 2.13 is true for *any* model P_θ. Moreover, the random vector over which the expectation is taken (in the right hand side) is precisely the direction of the update performed in Algorithm 1. We now prove Lemma 2.13, from which we derive Theorem 2.12.

Proof. Throughout this proof, expectations and probabilities will be over the same distributions as in the lemma statement. First, we use the law of total probability together with the autoregressive property of P_θ (Section 2.2) to switch from probabilities on transcripts, to products of next-token probabilities. Formally, consider a fixed input x, an honest transcript $\pi^* = (y^*, q_1^*, a_1^*, \ldots, q_R^*, a_R^*)$, and denote a random transcript sampled from $\mathcal{T}_V^\theta(x)$ when using the same verifier queries by $\pi = (y, q_1^*, a_1, \ldots, q_R^*, a_R)$. For any $r \in [R]$ denote the random variable $\mathcal{T}_V^{\theta,<r} \coloneqq \mathcal{T}_V^\theta(yq_1^*a_1 \cdots a_{r-1}q_r^*)$. Then,

$$\begin{aligned}
\Pr_\pi [\pi = \pi^*] &= \Pr_\pi[(y, a_1, \ldots, a_R) = (y^*, a_1^*, \ldots, a_R^*)] && (2.3) \\
&= \Pr_{y\sim P_\theta(x)}[y = y^*] \cdot \prod_{r\in[R]} \Pr_{a\sim\mathcal{T}_V^{\theta,<r}}[a = a_r^*] \\
&= \Pr_{y\sim P_\theta(x)}[y = y^*] \cdot \prod_{\substack{r\in[R] \\ s\in S(r)}} \Pr_{\sigma\sim p_\theta(\pi^*_{<s})}[\sigma = \pi_s^*] && (2.4) \\
&= \prod_{\substack{r\in[R]\cup\{0\} \\ s\in S(r)}} \alpha_s(\theta), && (2.5)
\end{aligned}$$

where, as noted above, Equation (2.3) uses the independence of the verifier and model's randomness, Equation (2.4) uses the autoregressive property of P_θ (Definition 2.10), and Equation (2.5) is by definition of α_s and of a_0. Next, a basic calculus identity gives

$$\nabla_\theta \left(\Pr_\pi [\pi = \pi^*] \right) = \Pr_\pi [\pi = \pi^*] \cdot \nabla_\theta \log \left(\Pr_\pi [\pi = \pi^*] \right) . \tag{2.6}$$

This implicitly assumes that $\Pr_\pi[\pi = \pi^*]$ is differentiable in θ; indeed, this follows from Definition 2.10, where the logits of the model were assumed to by differentiable. Let us focus on the rightmost factor. By Equation (2.5),

$$\nabla_\theta \log\left(\Pr_\pi[\pi = \pi^*]\right) = \nabla_\theta \log\left(\prod_{\substack{r\in[R]\cup\{0\}\\ s\in S(r)}} \alpha_s(\theta)\right) = \sum_{\substack{r\in[R]\cup\{0\}\\ s\in S(r)}} \nabla_\theta \log \alpha_s(\theta) = \sum_{\substack{r\in[R]\cup\{0\}\\ s\in S(r)}} \vec{d}_s(\theta) \tag{2.7}$$

where the last equality is by definition of $\vec{d}_s(\theta)$. Combining Equation (2.5) and Equation (2.6) gives

$$\nabla_\theta\left(\Pr_\pi[\pi = \pi^*]\right) = \prod_{\substack{r\in[R]\cup\{0\}\\ s\in S(r)}} \alpha_s(\theta) \cdot \sum_{\substack{r\in[R]\cup\{0\}\\ s\in S(r)}} \vec{d}_s(\theta).$$

By the law of total probability and the linearity of the gradient,

$$\mathop{\mathbb{E}}_{x,\pi^*}\left[\nabla_\theta\left(\Pr_\pi[\pi = \pi^*]\right)\right] = \nabla_\theta\left(\mathop{\mathbb{E}}_{x,\pi^*}\left[\Pr_\pi[\pi = \pi^*]\right]\right) = \nabla_\theta\left(\Pr_{x,\pi^*,\pi}[\pi = \pi^*]\right) = \nabla_\theta A(\theta).$$

which concludes the proof. □

We a gradient-estimation lemma at hand, we can now derive the convergence result.

Proof pf Theorem 2.12. Our strategy is to cast TL as Stochastic Gradient Ascent and apply Fact 2.17. Let ε, B_{Norm}, B_{Lip} and C as in the theorem statement be given. Let θ^* be such that $A(\theta^*) \geq 1 - \varepsilon/2$ and $||\theta^*|| \leq B_{\text{Norm}}$.

First, notice that

$$\mathop{\mathbb{E}}_{\bar{\theta}}\left[\text{ver}_{V,\mu}(\bar{\theta})\right] \geq \mathop{\mathbb{E}}_{\bar{\theta}}[A(\bar{\theta})],$$

This is because, for any x and model P_θ, whenever the transcript generated by $\mathcal{T}^\theta(x)$ agrees with π^*, then the verifier accepts (because π^* is honest). Therefore, to prove the theorem it suffices to show that

$$\mathop{\mathbb{E}}_{\bar{\theta}}[A(\bar{\theta})] \geq 1 - \varepsilon.$$

Following the notation in Algorithm 1, in every iteration $i \in [N]$ the norm of the update step is

$$\left\|\prod_{\substack{r\in[R]\cup\{0\}\\ s\in S(r)}} \alpha_s(\theta_i) \cdot \sum_{\substack{r\in[R]\cup\{0\}\\ s\in S(r)}} \vec{d}_s(\theta_i)\right\| = \left|\prod_{\substack{r\in[R]\cup\{0\}\\ s\in S(r)}} \alpha_s(\theta_i)\right| \cdot \left\|\sum_{\substack{r\in[R]\cup\{0\}\\ s\in S(r)}} \vec{d}_s(\theta_i)\right\|$$
$$\leq 1 \cdot \sum_{\substack{r\in[R]\cup\{0\}\\ s\in S(r)}} \left\|\vec{d}_s(\theta_i)\right\|,$$

where the inequality is because $\alpha_s(\theta_i)$ are probabilities, so ≤ 1. Continuing, we have

$$\sum_{\substack{r\in[R]\cup\{0\}\\ s\in S(r)}} \left\|\vec{d}_s(\theta_i)\right\| \leq \sum_{\substack{r\in[R]\cup\{0\}\\ s\in S(r)}} B_{\text{Lip}} \leq C \cdot B_{\text{Lip}}.$$

The first inequality is by definition of B_{Lip} as an upper-bound on the gradient of P_θ's logits. The second is because, by definition, C is an upper-bound on the number of tokens sent by the prover in the proof system, which is exactly the number of terms in the sum: r indexes rounds, and s indexes tokens sent in each round.

To conclude, Lemma 2.13 shows that TL samples from a gradient estimator for $A(\theta)$, while the above equation shows that the gradient is upper-bounded by $C \cdot B_{\text{Lip}}$. We can therefore apply Fact 2.17 to obtain

$$\mathop{\mathbb{E}}_{\bar{\theta}}\left[A\left(\bar{\theta}\right)\right] \geq A(\theta^*) - \varepsilon/2 \geq (1-\varepsilon/2) - \varepsilon/2 = 1-\varepsilon,$$

where the inequality is by definition of θ^*.

□

2.2.2 Reinforcement Learning from Verifier Feedback (RLVF)

As mentioned in Section 2.2.1, Transcript Learning uses access to an honest transcript generator to estimate gradients of (a lower bound on) the Verifiability of a model P_θ.

Reinforcement Learning from Verifier Feedback (RLVF, Algorithm 2) estimates this gradient without access to a transcript generator. RLVF can be viewed as a modification of TL in which the learner emulates the interaction of the verifier with its own model P_θ.

Before we continue with formal analysis of Algorithm 2, let us make a few observations.

Firstly, the parameters are updated (line 11) only when an accepting transcript was generated. This means that the learner can first fully generate the transcript (lines 6-7), and then take backwards passes (line 9) only if the transcript was accepted by V. This is useful in practice (e.g. when using neural models) as backwards passes are more computationally expensive than forwards passes.

On the other hand, this means that RLVF requires the parameter initialization θ_0 to have Verifiability bounded away from 0, so that accepting transcripts are sampled with sufficient probability. Fortunately, such a Self-Proving base model can be learned using TL. This gives a learning paradigm in which a somewhat-Self-Proving base model is learned with TL (with Verifiability $\delta > 0$), and then "amplified" to a fully Self-Proving model using RLVF. This can be seen as an adaptation of the method of Nair et al. (2018) to the setting of Self-Proving models.

Secondly, in comparing Algorithms 1 and 2, we see that the latter (RLVF) does not keep track of the probabilities α_s. This is because, in RL terms, RLVF is an *on-policy* algorithm; it generates transcripts using the current learned model, unlike TL that samples them from

Algorithm 2: Reinforcement Learning from Verifier Feedback (RLVF)

Hyperparameters: Learning rate $\lambda \in (0, 1)$ and number of samples $N \in \mathbb{N}$.
Input: An autoregressive model family $\{P_\theta\}_{\theta \in \mathbb{R}^d}$, initial parameters $\theta_0 \in \mathbb{R}^d$, verifier specification (code) V, and sample access to an input distribution μ.
Output: A vector of parameters $\bar{\theta} \in \mathbb{R}^d$.

for $i = 0, \ldots, N-1$ **do**
  Sample $x \sim \mu$.
  Initialize $a_0 \coloneqq y \sim P_{\theta_i}(x)$.
  foreach *Round of interaction* $r = 1, \ldots R$ **do**
    Sample the r^{th} query `# Emulate the verifier`

$$q_r \sim V_q(x, a_0, q_1, a_1, \ldots, q_{r-1}, a_{r-1}).$$

    Sample the r^{th} answer `# Forwards pass`

$$a_r \sim P_{\theta_i}(x, a_0, q_1, a_1, \ldots, q_r).$$

    Let $\tau_r \coloneqq (a_0, q_1, \ldots, a_{r-1}, q_r)$.
    for $s \in [L_a]$ **do**
      Let $a_{r,s}$ denote the s^{th} token in a_r. Compute `# Backwards pass`

$$\vec{d}_s(\theta_i) \coloneqq \nabla_\theta \log \Pr_{\sigma \sim p_{\theta_i}(x\tau_r)}[\sigma = a_{r,s}].$$

  if $V(x, y, q_1, a_1, \ldots, q_R, a_R)$ accepts **then**
    Update

$$\theta_{i+1} \coloneqq \theta_i + \lambda \cdot \sum_{\substack{r \in [R] \cup \{0\} \\ s \in [L_a]}} \vec{d}_s(\theta_i).$$

Output $\bar{\theta} \coloneqq \frac{1}{N} \sum_{i \in [N]} \theta_i$.

a distribution whose parameterization is unknown to the learner. Hence, the update step in RLVF is simpler than TL.

We now prove that the update step in RLVF maximizes the Verifiability of P_θ; this is analogous to Lemma 2.13 for TL. We leave it for future work to use Lemma 2.14 to obtain convergence bounds on RLVF (analogous to Theorem 2.12). RLVF can be derived by viewing Self-Proving as a reinforcement learning problem in which the agent (prover) is rewarded when the verifier accepts. Indeed, RLVF is the Policy Gradient method (Sutton et al., 1999) for a verifier-induced reward. Convergence bounds for Policy Gradient methods are a challenging and active area of research (e.g. Agarwal et al. 2021), and so we leave the full analysis to future work.

Lemma 2.14 (RLVF gradient estimation)**.** *Fix an input distribution μ over Σ^* and a verifier V with round complexity R and answer length L_a. For any transcript $(x, y, q_1, \ldots, a_R)$ we let $\mathrm{Acc}_V(x, y, q_1, \ldots, a_R)$ denote the indicator random variable which equals 1 if and only if V accepts the transcript. For any model P_θ, denote by $\mathrm{ver}(\theta)$ the verifiability of P_θ with respect to V and μ (Definition 2.4). Then, for any θ,*

$$\nabla_\theta \mathrm{ver}(\theta) = \mathop{\mathbb{E}}_{\substack{x\sim\mu \\ y\sim P_\theta(x) \\ (q_r,a_r)_{r=1}^R}} \left[\mathrm{Acc}_V(x, y, q_1, \ldots, a_R) \cdot \sum_{\substack{r\in[R]\cup\{0\} \\ s\in[L_a]}} \vec{d}_s(\theta) \right]$$

where $(q_r, a_r)_{r=1}^R$ are as sampled in lines 5-6 of Algorithm 2, and $\vec{d}_s(\theta)$ is as defined in line 8 therein.

Proof. Recall the *transcript generator of P_θ*, denoted by $\mathcal{T}_V^\theta$ (see Lemma 2.13). By the definitions of Verifiability in Definition 2.4 and $V(x, y, q_1, \ldots, a_R)$ in the lemma statement,

$$\begin{aligned} \mathrm{ver}(\theta) &\coloneqq \Pr_{\substack{x\sim\mu \\ y\sim P_\theta(x)}} \left[\langle V, P_\theta\rangle (x, y) \ \text{ accepts } \right] \\ &= \mathop{\mathbb{E}}_{\substack{x\sim\mu \\ y\sim P_\theta(x) \\ (q_r,a_r)_{r=1}^R}} \left[\mathrm{Acc}_V(x, y, q_1, \ldots, a_R)\right] \\ &= \mathop{\mathbb{E}}_{x\sim\mu} \left[\Pr_{\pi\sim\mathcal{T}_V^\theta(x)} \left[\mathrm{Acc}_V(x, \pi) \right] \right] \end{aligned} \tag{2.8}$$

Now, for every input x, let $\Pi^*(x) \subset \Sigma^*$ denote the set of accepting transcripts:

$$\Pi^*(x) \coloneqq \left\{\pi^* \in \Sigma^* : \mathrm{Acc}_V(x, \pi^*) = 1\right\}.$$

We can assume that $\Pi^*(x)$ has finite cardinality, since V's running time is bounded and hence the number of different transcripts that it can read (and accept) is finite. For any fixed input x, we can express its acceptance probability by the finite sum:

$$\Pr_{\pi\sim\mathcal{T}_V^\theta(x)} [\mathrm{Acc}_V(x, \pi)] = \sum_{\pi^*\in\Pi^*(x)} \Pr_{\pi\sim\mathcal{T}_V^\theta(x)} [\pi = \pi^*]. \tag{2.9}$$

We will use Equations (2.3) through (2.7) in the proof of Lemma 2.13. Up to a change in index notation, these show that, for any π^*,

$$\nabla_\theta \Pr_{\pi\sim\mathcal{T}^\theta(x)} [\pi = \pi^*] = \Pr_{\pi\sim\mathcal{T}^\theta(x)} [\pi = \pi^*] \cdot \sum_{\substack{r\in R\cup\{0\} \\ s\in[L_a]}} \nabla_\theta \vec{d}_s(\theta).$$

Combining Equations (2.8) and (2.9), by linearity of expectation we have that

$$\begin{aligned}
\nabla_\theta \mathrm{ver}(\theta) &= \mathop{\mathbb{E}}_{x\sim\mu}\left[\sum_{\pi^*\in\Pi^*(x)} \nabla_\theta \Pr_{\pi\sim\mathcal{T}^\theta(x)}[\pi=\pi^*]\right] \\
&= \mathop{\mathbb{E}}_{x\sim\mu}\left[\sum_{\pi^*\in\Pi^*(x)} \Pr_{\pi\sim\mathcal{T}^\theta(x)}[\pi=\pi^*]\cdot \sum_{\substack{r\in R\cup\{0\}\\ s\in[L_a]}} \nabla_\theta \vec{d}_s(\theta)\right] \\
&= \mathop{\mathbb{E}}_{x\sim\mu}\left[\mathop{\mathbb{E}}_{\pi\sim\mathcal{T}^\theta(x)}\left[\mathrm{Acc}_V(x,\pi)\cdot \sum_{\substack{r\in R\cup\{0\}\\ s\in[L_a]}} \nabla_\theta \vec{d}_s(\theta)\right]\right] \\
&= \mathop{\mathbb{E}}_{\substack{x\sim\mu\\ \pi\sim\mathcal{T}^\theta(x)}}\left[\mathrm{Acc}_V(x,\pi)\cdot \sum_{\substack{r\in R\cup\{0\}\\ s\in[L_a]}} \nabla_\theta \vec{d}_s(\theta)\right] \\
&= \mathop{\mathbb{E}}_{\substack{x\sim\mu\\ y\sim P_\theta(x)\\ (q_r,a_r)_{r=1}^R}}\left[\mathrm{Acc}_V(x,y,q_1,\ldots,a_R)\cdot \sum_{\substack{r\in R\cup\{0\}\\ s\in[L_a]}} \nabla_\theta \vec{d}_s(\theta)\right],
\end{aligned}$$

where in the last equality, the probability is over (q_r, a_r) sampled as in Algorithm 2, and it follows from the definition of the transcript generator $\mathcal{T}^\theta(x)$. □

2.2.3 Learning from annotated transcripts

To minimize the length of messages exchanged in an Interactive Proof system, the honest prover is designed to send the shortest possible message to the verifier, containing only essential information.

However, when training Self-Proving model, it may be useful for it to first generate an "annotated" answer $\widetilde{a}$ which is then trimmed down to the actual answer a to be sent to the verifier. We adapt Sections 2.1 and 2.2 to this setting by considering *Annotated Transcripts*. The TL and RLVF algorithms naturally extend to annotated transcripts as well. Table 2.2 shows that annotations significantly improve performance of TL.

Formally, we define a *transcript annotator* and an *answer extractor* incorporated into the training and inference stages, respectively.

Fix a verifier V in an R-round proof system with question length L_q and answer length L_a. An *annotation system* with annotation length $\widetilde{L_a}$ consists of a *transcript annotator* A, and an *answer extractor* E.

In terms of efficiency, think of the annotator as an algorithm of the same computational resources as an honest prover in the system (see Definition 2.2), and the answer extractor as an extremely simple algorithm (e.g., trim a fixed amount of tokens from the annotation).

To use an annotation system the following changes need to be made:

- At training time, an input x and transcript π is annotated to obtain $\widetilde{\pi} \coloneqq A(x, \pi)$, e.g. before the forwards backwards pass in TL (line 3 in Algorithm 1).
- At inference time (i.e., during interaction between V and P_θ), the prover keeps track of the annotated transcript, but in each round passes the model-generated (annotated) answer through the extractor E before it is sent to the verifier. That is, in each round $r \in [R]$, the prover samples

$$\widetilde{a_r} \sim P_\theta(x, y, q_1, \widetilde{a_1}, \ldots, \widetilde{a_{r-1}}, q_r).$$

  The prover then extracts an answer $a_r \coloneqq E(\widetilde{a_r})$ which is sent to the verifier.

Remark 2.15. *Annotations can be viewed as adding Chain-of-Thought (Wei et al., 2022). As a concrete example, consider our experiments on computing the GCD. As detailed in Section 2.3.4, a proof π in this setting is the output of an iterative process—the extended Euclidean algorithm—starting from the input x: $x \mapsto \pi_1 \mapsto \pi_2 \mapsto \cdots \mapsto \pi$. The annotation of the proof π consists the first T steps $(\pi_1, \ldots, \pi_T)$ up to some fixed cutoff T. These are prepended to the proof and shown to the model during TL training. At inference time, the model is evaluated only on whether it generated the proof π correctly.*

2.2.4 Background on Stochastic Gradient Descent

For convenience of the reader, we provide a description of Stochastic Gradient Ascent and quote a theorem on its convergence. We adapt the presentation of Shalev-Shwartz and Ben-David (2014), noting that they present Stochastic Gradient Descent (SGD) in its more general form for non-differentiable unbounded functions. This section may be skipped by readers familiar with SGD convergence.

Stochastic Gradient Ascent (SGA) is a fundamental technique in concave optimization. Given a concave function $f\colon \mathsf{R}^d \to [0, 1]$, SGA starts at $w_0 = \vec{0} \in \mathsf{R}^d$ and tries to maximize $f(w)$ by taking a series of "steps." Than directly differentiating f, SGA instead relies on an estimation $\nabla f(w)$: in each iteration, SGA takes a step in a direction that estimates $\nabla f(w)$.

Definition 2.16 (Gradient estimator)**.** *Fix a differentiable function $f\colon \mathsf{R}^d \to \mathsf{R}$ for some d. A* gradient estimator *for f is a randomized mapping $D_f\colon \mathsf{R}^d \to \mathsf{R}^d$ whose expectation is the gradient of f. That is, for all $w \in \mathsf{R}^d$,*

$$\mathop{\mathbb{E}}_{v \sim D_f(w)}[v] = \nabla f(w).$$

Note that this is an equality between d-dimensional vectors.

Algorithm 3: Stochastic Gradient Ascent

Hyperparameters: Learning rate $\lambda > 0$ and number of iterations $N \in \mathbb{N}$.
Input: A function $f\colon \mathsf{R}^d \to \mathsf{R}$ to maximize and a gradient estimator D_f for f.
Output: A vector $\bar{w} \in \mathbb{R}^d$.

Initialize $w_0 \coloneqq \vec{0} \in \mathsf{R}^d$.
for $i = 1, \ldots, N-1$ **do**
  Sample $v_i \sim D_f(w_{i-1})$.
  Update $w_i \coloneqq w_{i-1} + \lambda \cdot v_i$.
Output $\bar{w} \coloneqq \frac{1}{N} \sum_{i \in [N]} w_i$.

Theorem 14.8 of Shalev-Shwartz and Ben-David (2014) implies the following fact.

Fact 2.17. *Fix a concave $f\colon \mathsf{R}^d \to [0,1]$, a norm $||\cdot||$ on R^d, and upper-bounds $B_{\text{Norm}}, B_{\text{Lip}} > 0$. Let*

$$w^* \in \operatorname*{argmax}_{w : ||w|| < B_{\text{Norm}}} f(w),$$

and let $\bar{w}$ denote the output of Algorithm 3 run for N iterations with learning rate

$$\lambda = \frac{B_{\text{Norm}}}{B_{\text{Lip}}\sqrt{N}}.$$

If at every iteration it holds that $||v_i|| < B_{\text{Lip}}$, then

$$\mathop{\mathbb{E}}_{\bar{w}}[f(\bar{w})] \geq f(w^*) - \frac{B_{\text{Norm}} \cdot B_{\text{Lip}}}{\sqrt{N}}.$$

Learning with Stochastic Gradient Ascent/Descent. Fact 2.17 captures the general case of using SGA for maximization of concave problems. It is more common for the literature to discuss the equivalent setting of Stochastic Gradient Descent (SGD) for minimization of convex problems. Specifically, a common application of SGD is for the task of *Risk Minimization*: given a loss function and access to an unknown distribution of inputs, the goal is to minimize the expected loss with respect to the distribution. Assuming that the loss function is differentiable, the gradient of the loss serves as a gradient estimator (see Definition 2.16) for the risk function. We refer the reader to Shalev-Shwartz and Ben-David (2014, Section 14.5.1) for a complete overview of SGD for risk minimization.

For the sake of completeness, we formulate Transcript Learning (TL, Algorithm 1) in the framework of Risk Minimization for Supervised Learning. Although multiple loss functions may achieve our ultimate goal—learning Self-Proving models—in what follows we define the loss that corresponds to TL. Fix a verifier V and let $\mathcal{T}_V^*$ denote a distribution over accepting transcripts. We define

$$\mathsf{loss}\left(\theta, (x, \pi^*)\right) \coloneqq \Pr_{\pi \sim \mathcal{T}_V^\theta(x)}\left[\pi \neq \pi^*\right], \tag{2.10}$$

where π^* and π share the same verifier messages (as in Lemma 2.13) so the inequality is only over the prover's messages, namely $\Pr_{\pi\sim\mathcal{T}_V^\theta(x)}[\pi \neq \pi^*] = \Pr_{\pi\sim\mathcal{T}_V^\theta(x)}[(y, a_1, \ldots, a_R) \neq (y^*, a_1^*, \ldots, a_R^*)]$.[8]

The risk function is the expected value of the loss over the joint distribution of inputs and accepting transcripts $\mu \times \mathcal{T}_V^*(\mu)$:

$$\mathsf{Risk}(\theta) \coloneqq \mathop{\mathbb{E}}_{\substack{x\sim\mu \\ \pi^*\sim\mathcal{T}_V^*}} [\mathsf{loss}(\theta, (x, \pi^*))],$$

which means that the *agreement function* defined in Theorem 2.12

$$A(\theta) = \Pr_{\substack{x\sim\mu \\ \pi^*\sim\mathcal{T}_V^*(x) \\ \pi\sim\mathcal{T}_V^\theta(x)}} [\pi = \pi^*]$$

satisfies $A(\theta) = 1 - \mathsf{Risk}(\theta)$.

Thus, maximizing the agreement is equivalent to minimizing the risk. The hypothesis class over which the optimization is performed is the ball of radius B_{Norm}, i.e., $\left\{\theta \in \mathsf{R}^d : ||\theta|| < B_{\text{Norm}}\right\}$. The assumption that A is concave in θ implies that the loss function is convex in θ, which is the required assumption for using SGD for risk minimization.

Indeed, TL uses the natural gradient estimator for this setting, the gradient of the "complement" of the loss: $\Pr_\pi[\pi = \pi^*]$, since TL maximizes the agreement instead of minimizing the risk. The proof of Lemma 2.13, i.e., $\nabla_\theta A(\theta) = \mathbb{E}_{x,\pi^*}[\nabla_\theta(\Pr_\pi[\pi = \pi^*])]$, follows from the above discussion.

2.3 Training a Self-Proving transformer for the GCD

We describe our experimental setup, and present ablation studies that shed additional light on the effect of *annotation* and *representation* on Verifiability.

2.3.1 Setup: Training transformers to predict the GCD of two integers

Charton (2024) empirically studies the power and limitations of learning GCDs with transformers. We follow their setup and two conclusions on settings that make for faster learning: Training from the log-uniform distribution, and choosing a base of representation with many prime factors.

We fix a base of representation $B = 210$ and use $\mathbf{x}$ to denote an integer x encoded as a B-ary string.[9] For sequences of integers, we write $(\mathbf{x_1}\mathbf{x_2})$ to denote the concatenation of $\mathbf{x_1}$

[8] This loss is not to be confused with those discussed in Definition 2.9. Here, we are simply explaining how TL can be viewed as a supervised risk minimizer for the loss function defined in Equation (2.10).

[9] $B = 210$ is chosen following Charton (2024) to be an integer with many prime factors.

with $\mathbf{x_2}$, delimited by a special token. The vocabulary size needed for this representation is $|\Sigma| \approx 210$.

We choose the input distribution μ to be the log-uniform distribution on $[10^4]$, and train the transformer on sequences of the form $(\mathbf{x_1 x_2 y})$, where $x_1, x_2 \sim \mu$ and $y = GCD(x_1, x_2)$. This is a scaling-down of Charton (2024), to allow single GPU training of Self-Proving transformers. In all of our experiments, we use a GPT model (Vaswani et al., 2017) with 6.3M parameters trained on a dataset of 1024K samples in batches of 1024. Full details are deferred to Section 2.3.6.

2.3.2 Setup: Proving correctness of the GCD

Following Charton (2024) as a baseline, we find that transformers can correctly compute the GCD with over 99% probability over $(x_1, x_2) \sim \mu$. To what extent can they *prove* their answer? To answer this question, we first devise a natural proof system based on Bézout's theorem.

Before we dive in, let us clarify what we mean by *a proof system for the GCD*. Prover Paul has two integers 212 and 159; he claims that $GCD(212, 159) = 53$. An inefficient way for Verifier Veronica to check Paul's answer is by executing the Euclidean algorithm on $(212, 159)$ and confirm that the output is 53. In an efficient proof system, Veronica asks Paul for a short string π^* (describing two integers) with which she can easily compute the answer—without having to repeat Paul's work all over. On the other hand, if Paul were to claim that "$GCD(212, 159) = 51$" (it does not), then for any alleged proof π, Veronica would detect an error and reject Paul's claim.

The verifier in the proof system relies on the following fact.

Claim 2.18 (Bézout's identity (Bezout, 1779))**.** *Let $x_0, x_1 \in \mathbb{N}$ and $z_0, z_1 \in \mathbb{Z}$. If $z_0 \cdot x_0 + z_1 \cdot x_1$ divides both x_0 and x_1, then $z_0 \cdot x_0 + z_1 \cdot x_1 = GCD(x_0, x_1)$.*

Any coefficients z_0, z_1 satisfying the assumption of Claim 2.18 are known as *Bézout coefficients* for (x_0, x_1). Claim 2.18 immediately gives our simple proof system: For input $x = (x_0, x_1)$ and alleged GCD y, the honest prover sends (alleged) Bézout coefficients (z_0, z_1). The Verifier accepts if and only if $y = z_0 \cdot x_0 + z_1 \cdot x_1$ and y divides both x_0 and x_1.

In this proof system the Verifier does not need to make any query; to fit within Definition 2.2, we can have the verifier issue a dummy query. Furthermore, by Claim 2.18 it is complete and has soundness error $s = 0$. Lastly, we note that the Verifier only needs to perform two multiplications, an addition, and two modulus operations; in that sense, verification is more efficient than computing the GCD in the Euclidean algorithm as required by Remark 2.3.

Annotations. To describe how a proof $z = (z_0, z_1)$ is annotated, let us first note how it can be computed. The Bézout coefficients can be found by an extension of the Euclidean algorithm. It is described in Algorithm 4.

Algorithm 4: Extended Euclidean algorithm

Input: Nonzero integers $x_0, x_1 \in \mathbb{N}$.

Output: Integers (y, z_0, z_1), such that $y = GCD(x_0, x_1)$ and (z_0, z_1) are Bézout coefficients for (x_0, x_1).

Initialize $r_0 = x_0$, $r_1 = x_1$, $s_0 = 1$, $s_1 = 0$, and $q = 0$.
while $r_1 \neq 0$ **do**
  Update $q \coloneqq \lfloor r_0/r_1 \rfloor$.
  Update $(r_0, r_1) \coloneqq (r_1, r_0 - q \times r_1)$.
  Update $(s_0, s_1) \coloneqq (s_1, s_0 - q \times s_1)$.
Output GCD $y = r_0$ and Bézout coefficients $z_0 \coloneqq s_0$ and $z_1 \coloneqq (r_0 - s_0 \cdot x_0)/x_1$.

Referring to Algorithm 4, the annotation of a proof $z = (z_0, z_1)$ will consist of intermediate steps in its computation. Suppose that in each iteration of the While-loop, the algorithm stores each of r_0, s_0 and q in an arrays $\vec{r_0}$, $\vec{s_0}$ and $\vec{q}$. The annotation $\tilde{z}$ of z is obtained by concatenating each of these arrays. In practice, to avoid the transformer block (context) size from growing too large, we fix a cutoff T and first trim each array to its first T elements.

We formalize this in the terminology of Section 2.2.3 by defining a Transcript Annotator and Answer Extractor. Note that, since our proof system consists only of one "answer" z send from the prover to the verifier, the entire transcript π is simply $z = (z_0, z_1)$. Since the verification is deterministic, this means that the proof system is of an NP type (however, note that the search problem of finding the "NP-witness" $z = (z_0, z_1)$ is in fact in P).

- *Transcript Annotator A:* For a fixed cutoff T and given input $x = (x_0, x_1)$ and transcript $z = (z_0, z_1)$, A executes Algorithm 4 on input $x = (x_0, x_1)$. During the execution, A stores the first T intermediate values of r_0, s_0 and q in arrays $\vec{r_0}$, $\vec{s_0}$ and $\vec{q}$. It outputs $A(x, z) \coloneqq (\vec{r_0}, \vec{s_0}, \vec{q}, z)$.
- *Answer Extractor E:* Given an annotated transcript $\tilde{z} = (\vec{r_0}, \vec{s_0}, \vec{q}, z)$, outputs $E(\tilde{z}) \coloneqq z$.

We note that the computational complexity of A is roughly that of the honest prover, i.e., Algorithm 4 (up to additional space due to storing intermediate values). As for E, it can be implemented in logarithmic space and linear running time in $|\tilde{z}|$, i.e., the length of the description.[10]

2.3.3 Experimental results

To measure Verifiability, we train a Self-Proving transformer using Transcript Learning on sequences $(\mathbf{x_1 x_2 y} \pi)$ and estimate for how many inputs $x_1, x_2 \sim \mu$ does the model generate

[10] That is, if integers are represented by n-bits, then E has space complexity $O(\log n + \log T)$ and running time $O(n \cdot T)$.

both the correct GCD $\mathbf{y}$ and a valid proof π. We test on 1000 pairs of integers $x'_1, x'_2 \sim \mu$ held-out of the training set, prompting the model with $(\mathbf{x'_1 x'_2})$ to obtain $(\mathbf{y'}\pi')$, and testing whether $V(\mathbf{x'_1 x'_2 y'}\pi')$ accepts.

Table 2.2 shows our main experimental result, which has the following key takeaways:

1. Transcript Learning (TL) for 100K iterations (≈100M samples) results in a Self-Proving transformer that correctly proves 60.3% of its answers.

2. A base Self-Proving Model with fairly low Verifiability of 40% can be improved to 79.3% via Reinforcement Learning from Verifier Feedback (RLVF). Although it does not rely on honest transcripts, RLVF trains slowly: this nearly-twofold improvement took four million iterations.

3. Most efficient is Annotated Transcript Learning, which yielded a model with 96% Verifiability in 100K iterations.

We further investigate the effect of annotations next.

2.3.4 Models generalize beyond annotations

Recall that a proof π is annotated by including intermediate steps in its computation: roughly speaking, the proof π for input $(\mathbf{a}, \mathbf{b})$ is obtained as the last element in a sequence $\mathbf{a}, \mathbf{b}, \pi_\mathbf{1}, \pi_\mathbf{2}, \ldots$ computed by the Euclidean algorithm. We annotate the proof π by prepending to it the sequence of *Euclidean steps* $(\pi_\mathbf{1}, \ldots, \pi_\mathbf{T})$ up to some fixed cutoff T.

Figure 2.3 shows how T affects the Verifiability of the learned model. As suggested by Lee et al. (2024), training the model on more intermediate steps results in better performance; in our case, increasing the number of intermediate steps T yields better Self-Proving models. One might suspect that models only learn to execute the Euclidean algorithm in-context. To rule out this hypothesis, we derive an upper bound on the possible efficacy of such limited models. This bound is based on the *Euclidean depth* of integers (x_1, x_2), which we define as the number of intermediate steps that the Euclidean algorithm makes before terminating on input (x_1, x_2). Indeed, a model that only learns to compute (in-context) the simple arithmetic of the Euclidean algorithm would only be able to prove the correctness of inputs (x_1, x_2) whose depth does not exceed the annotation cutoff T.

Figure 2.3 tells a different story: For each cutoff T, we estimate the probability that integers $x_1, x_2 \sim \mu$ have Euclidean depth at most T on 10^5 sampled pairs. Larger annotation cutoff T increases Verifiability, but all models exceed their corresponding Euclidean depth bound.

2.3.5 Base of representation

As mentioned previously, Charton (2024) concludes that, for a given base of representation B, transformers correctly compute the GCD of integers x_1, x_2 that are products of primes

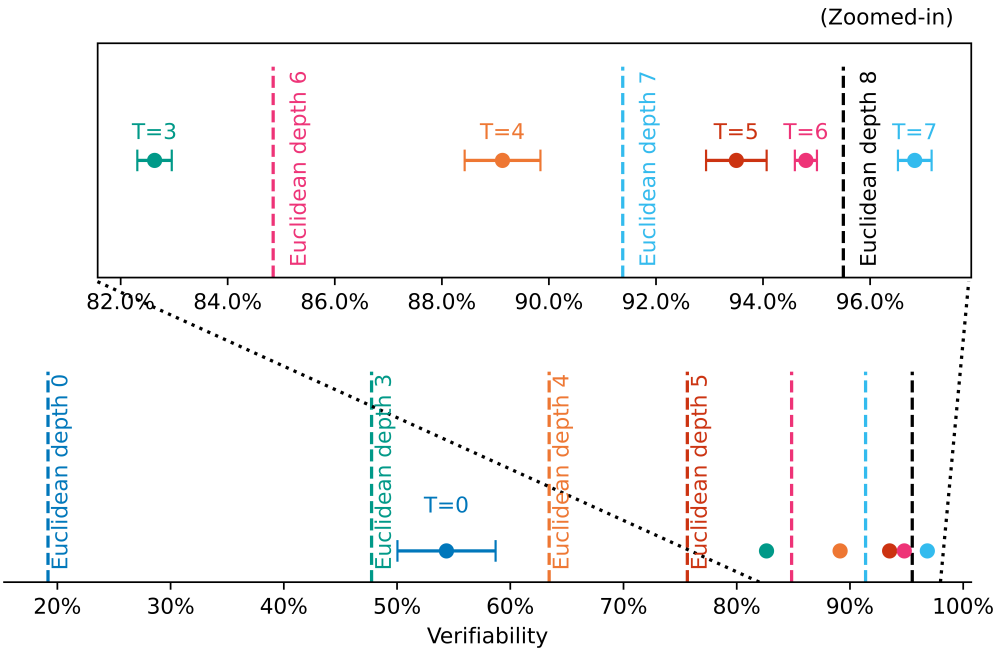


Figure 2.3: **Verifiability with increasing amounts of annotation**. T is the number of steps added in Annotated Transcript Learning. Dashed lines indicate *Euclidean depth*, that bound the Verifiability of models that prove *only* for integers up to a certain number of steps. Each T was run with three seeds, with mean $\pm$ standard error depicted. The upper graph provides a zoomed-in view of the 82% to 98% range from the lower graph, which spans a broader scale from 20% to 100%.

dividing B. Simply put, choosing a base B with many different prime factors yields models with better correctness (accuracy), which suggests why base $B = 210 = 2 \cdot 3 \cdot 5 \cdot 7$ yielded the best results.

To test whether the factorization of B has a similar effect on Verifiability as well, we train transformers on 68 bases varying the number of prime divisors $\omega(B)$ from $\omega(B) = 1$ (i.e., B is a prime power) to $\omega(B) = 4$. Figure 2.4 shows that $\omega(B)$ correlates not just with correctness (Charton, 2024), but also with Verifiability. Although the finding is statistically significant (no overlapping error margins), the overall difference is by a few percentage points; we attribute this to the smaller (10%) number of samples on which models were trained, relative to our other experiments.

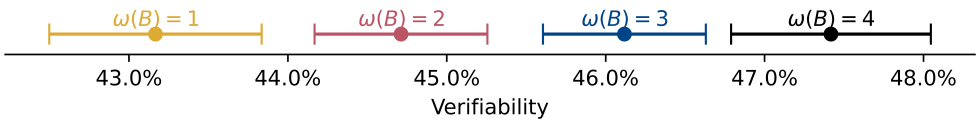


Figure 2.4: **The number of prime divisors of a base $\omega(B)$ determines Verifiability.** For each $o \in [4]$, we sampled 17 bases $B \in \{2, \ldots, 1386\}$ such that $\omega(B) = o$. A Self-Proving transformer was trained via Transcript Learning for twenty epochs on an identical dataset of 1024K samples encoded in base B. For each $\omega(B)$ we depict the mean $\pm$ standard error.

2.3.6 Full experiment details

We provide details of how we implemented the experiments in Section 2.3 and additional figures for each experiment. Code, data and models are available at `https://github.com/orrp/self-proving-models`.

Model architecture. We use Karpathy's *nanoGPT*[11] implementation of GPT. Note that we train the model "from scratch" only on sequences related to the GCD problem, rather than starting from a pretrained checkpoint. We use a 6.3M parameter architecture of 8 layers, 8 attention heads, and 256 embedding dimensions. We optimized hyperparameters via a random hyperparameter search, arriving at learning rate 0.0007, AdamW $\beta_1 = 0.733$ and $\beta_2 = 0.95$, 10% learning rate decay factor, no dropout, gradient clipping at 2.0, no warmup iterations, and 10% weight decay.

Data. We sample integers from the $\log_{10}$-uniform distribution over $\{1, \ldots, 10^4\}$. Models in Table 2.2 and Fig. 2.3 are trained for 100K iterations on a dataset of ≈10M samples. For Figure 2.4 (base ablation) we train for 20K iterations on a dataset of ≈1M samples; this is because this setting required 68 many runs in total, whereas the annotation-cutoff ablation required 18 longer runs.

Compute. All experiments were run on a machine with an NVIDIA A10G GPU, 64GB of RAM, and 32 CPU cores. The longest experiment was the single RLVF run, which took one month and four days. The annotation-cutoff ablation runs took about 75 minutes each. Base of representation ablation runs were shorter at about 15 minutes each. The total running time of the Transcript Learning experiments was approximately 40 hours (excluding time dedicated to a random hyperparameter search), and the RLVF experiment took another month and four days. The overall disk space needed for our models and data is 4GB.

[11] `https://github.com/karpathy/nanoGPT`.

Representing integers. We fully describe how integer sequences are encoded. As a running example, we will use base 210. To encode a sequence of integers, each integer is encoded in base 210, a sign is prepended and a delimiter is appended, with a unique delimiter identifying each component of the sequence. For example, consider the input integers $x_0 = 212$ (which is 12 in base 210) and $x_1 = 159$. Their GCD is $y = 53$, with Bézout coefficients $z_0 = 1$ and $z_1 = -1$. Therefore, the sequence $(212, 159, 53, 1, -1)$ is encoded as

```
+,1,2,x0,+,159,x1,+,53,y,+,1,z0,-,1,z1
```

where commas are added to distinguish between different tokens. Null tokens are appended to pad all sequences in a dataset to the same length. Both the input and the padding components are ignored when computing the loss and updating parameters.

Annotations Annotations are encoded as above, with each component in an intermediate step π_t delimited by a unique token. Since different integer pairs may require a different number of intermediate steps to compute the Bézout coefficients, we chose to pad all annotations to the same length T by the last step π_T in the sequence (which consists of the final Bézout coefficients). This ensures that the final component output by the model in each sequence should be the Bézout coefficient, and allows us to batch model testing (generation and evaluation) resulting in a 1000x speed-up over sequential testing.

As an example, consider the inputs $x_0 = 46$ and $x_1 = 39$. Tracing through the execution of Algorithm 4, we have

x_0	x_1	y	$\vec{s_0}$	$\vec{r_0}$	$\vec{q}$	z_0	z_1
46	39		1	46	1		
			0	39	5		
			1	7	1		
			-5	4	1		
			6	3	3		
		1				-11	13

To encode this as an annotated transcript for the transformer, we must specify a base of representation and an annotation cutoff. Suppose that we wish to encode this instance in base $B = 10$ and cutoff $T = 3$. Then the input with the annotated transcript is encoded as

```
+,4,6,x0,+,3,9,x1,+,1,y,
+,1,z0',+,4,6,z1',+,1,q',
+,0,z0'',+,3,9,z1'',+,5,q'',
+,1,z0''',+,7,z1''',+,1,q''',
-,1,1,z0,+,1,3,z1
```

where commas are used to separate between tokens, and linebreaks are added only for clarity. Notice the three types of tokens: signs, digits, and delimiters. Notice also that the output y is added immediately after the input, followed by the annotated transcript (whose six

tokens comprise the proof itself). Since the Self-Proving model we train has causal attention masking, placing the output y before the proof means that the model "commits" to an output and only then proves it.

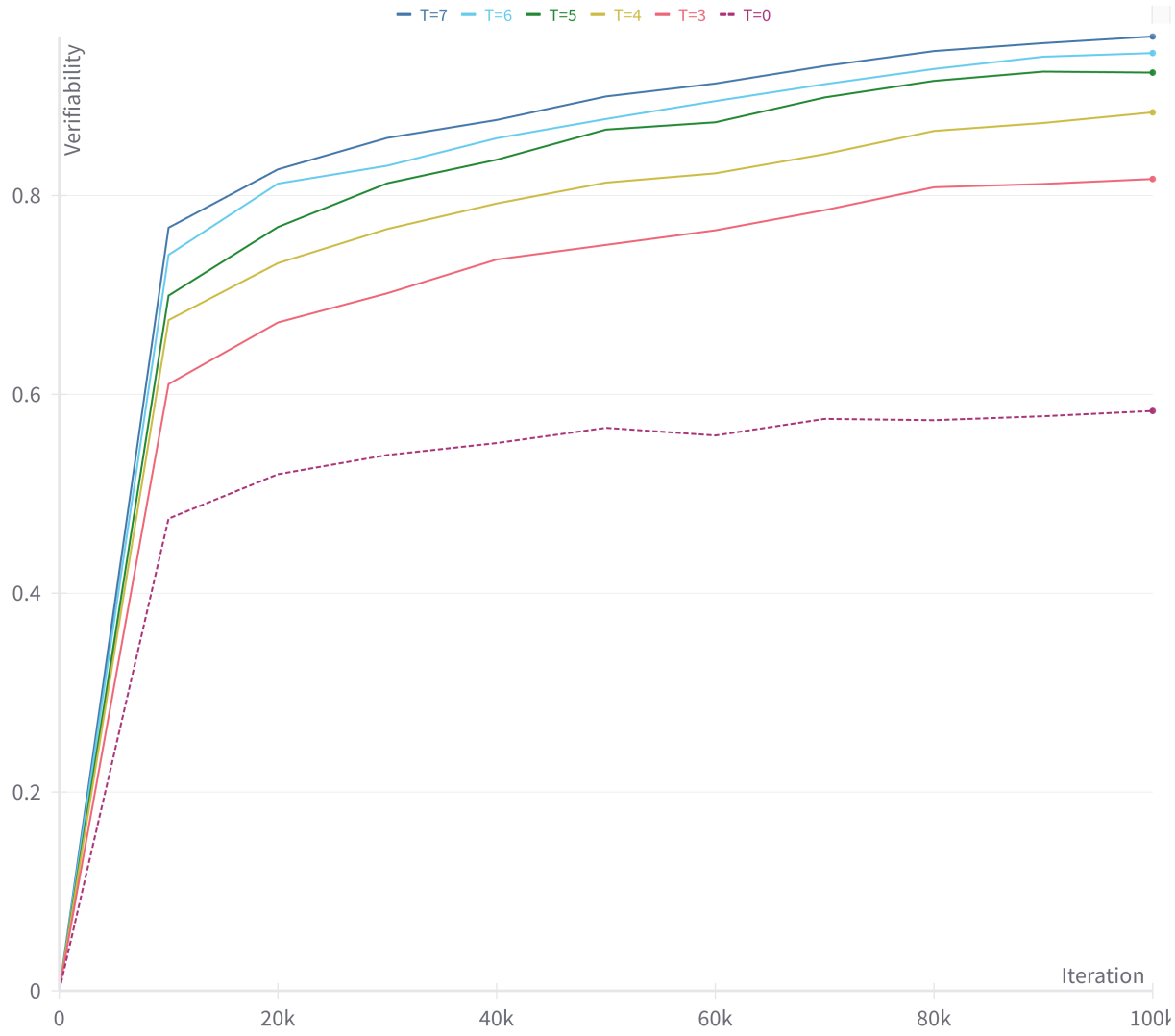


Figure 2.5: **Verifiability as a function of the number of samples** N. Each iteration (X axis) is a batch of 1024 samples from a dataset of ≈10M sequences. Every 10k iterations, Verifiability was evaluated on a held-out dataset of 1k inputs (as described in Section 2.3). T is the number of steps in Annotated Transcript Learning (Figure 2.3), and $T = 0$ is non-annotated Transcript Learning. Each T was run with three seeds, with mean depicted by the curve and standard error by the shaded area.

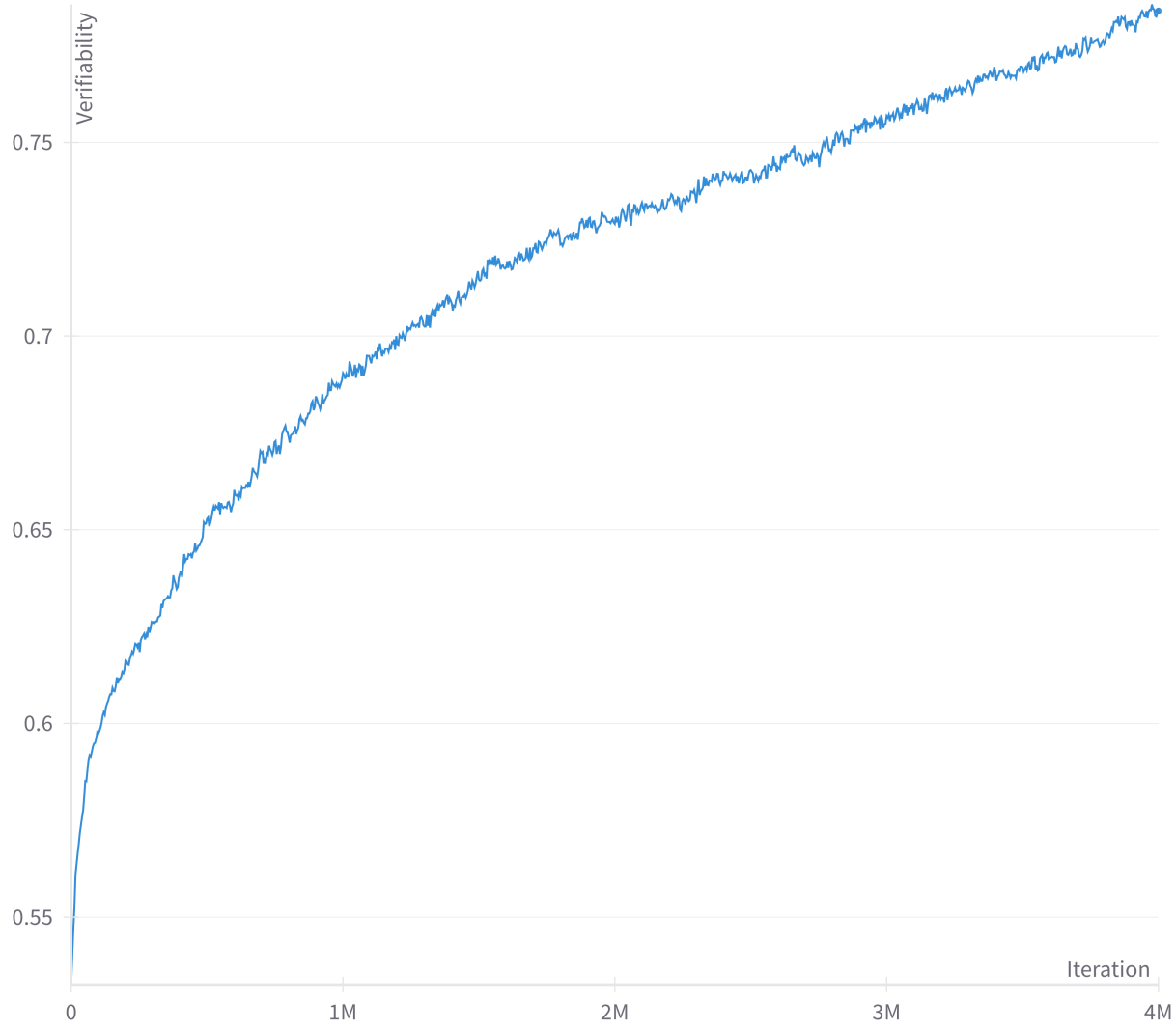


Figure 2.6: **RLVF Verifiability as a function of the number of samples** N. Starting from a base model with Verifiability 48% (obtained via Transcript Learning), in each iteration a batch of 2048 inputs are sampled; the model generates a proof for each; the Verifier is used to check which proofs are accepted; then, the model parameters are updated accordingly (see Algorithm 2). Verifiability was evaluated on a held-out dataset of 1k inputs.

2.3.7 Limitations

Our experiments are focused on a single ground-truth capability, namely, computing the GCD. Yet, the theoretical portion of our work holds for any ground-truth F^* that admits

an Interactive Proof system. Training large Self-Proving models for more complex ground-truths will likely pose additional practical learning challenges. With that said, we stress that generating accepting transcripts for use in Transcript Learning is distinct from these learning challenges. Collecting accepting transcripts is a purely computational task, and can even be done "offline" prior to the model's training.

2.4 Conclusion

Trust between a learned model and its user is fundamental. In recent decades, Interactive Proofs (Goldwasser, Micali, and Rackoff, 1985) have emerged as a general theory of trust established via verification algorithms. This work demonstrates that models can learn to formally prove their answers in an Interactive Proof system. We call models that possess this capability *Self-Proving*.

The definition of Self-Proving models forms a bridge between the rich theory of Interactive Proofs and the contemporary topic of Trustworthy ML. Interactive Proofs offer formal *worst-case soundness guarantees*; thus, users of Self-Proving models can be confident when their models generate correct answers—and detect incorrect answers with high probability.

We demonstrate the theoretical viability of our definition with two generic learning algorithms: Transcript Learning (TL) and Reinforcement Learning from Verifier Feedback (RLVF). The analyses of these algorithms is informed by techniques from theories of learning, RL, and computational complexity. This work can be extended in several directions: finding conditions for the convergence of RLVF, improving sample complexity bounds for TL, or designing altogether different learning algorithms (for example, by taking advantage of properties of the verifier).

To better understand the training dynamics of (Annotated) TL, we train Self-Proving transformers for the Greatest Common Divisor (GCD) problem. We train a small (6.3M parameter) transformer that learns to generate correct answers *and proofs* with high accuracy. Facing forward, we note that Interactive Proofs exist for capabilities far more complex than the GCD (Shamir, 1992); scaling up our experiments is the next step towards bringing Self-Proving models from theory to practice.

Chapter 3

Models That Prove Their Own "Intelligence"

Recent works claim that GPT-4 achieves expert-level performance on complex reasoning tasks (Katz et al., 2023; Lin et al., 2023), with some researchers concluding that it exhibits sparks of intelligence (Bubeck et al., 2023).

But how should intelligence be evaluated? This question dates back to Descartes (1637), formalized by Turing (1950), and continues to be the subject of recent discussion (Chollet 2019; Mitchell and Krakauer 2023; Burnell et al. 2023 *inter alia*). However, none of these attempts prescribe a particular evaluator (e.g., sequence of questions) that guarantees the intelligence of the evaluated model.

This is not a coincidence. We argue that intelligence is in the eye of the evaluator. This maxim is particularly important for the future of natural language processing (NLP): progress cannot be measured by static benchmarks (Raji et al., 2021; Hutchinson et al., 2022; Shirali, Abebe, and Hardt, 2023), with contemporary models surpassing human performance on new evaluations within a few years (Kiela et al., 2021), and benchmarks leaking into training data (Elangovan, He, and Verspoor, 2021).

Instead, we define the notion of *pseudointelligence*. Analogous to pseudorandomness (Blum and Micali, 1984; Yao, 1982), which measures a distribution by its distinguishability from true randomness, pseudointelligence applies to the evaluation of the capabilities of learned models. Importantly, a claim that a model has learned a certain capability is innately entangled with the distinguishing ability of an evaluator.

With the future of NLP in mind, we focus on *learned evaluators*. These evaluators are trained on samples specific to a given capability, much like the models they assess. Notably, emerging evaluation methods, such as model-based evaluation (Perez et al., 2023; Ribeiro et al., 2021) and adversarial evaluation (Jia and Liang, 2017; Nie et al., 2020; Bartolo et al., 2020), can be viewed as specific instances of the framework we propose. Our main takeaways are:

P1: A claim of intelligence must be supplemented by an explicitly-defined *evaluator* and

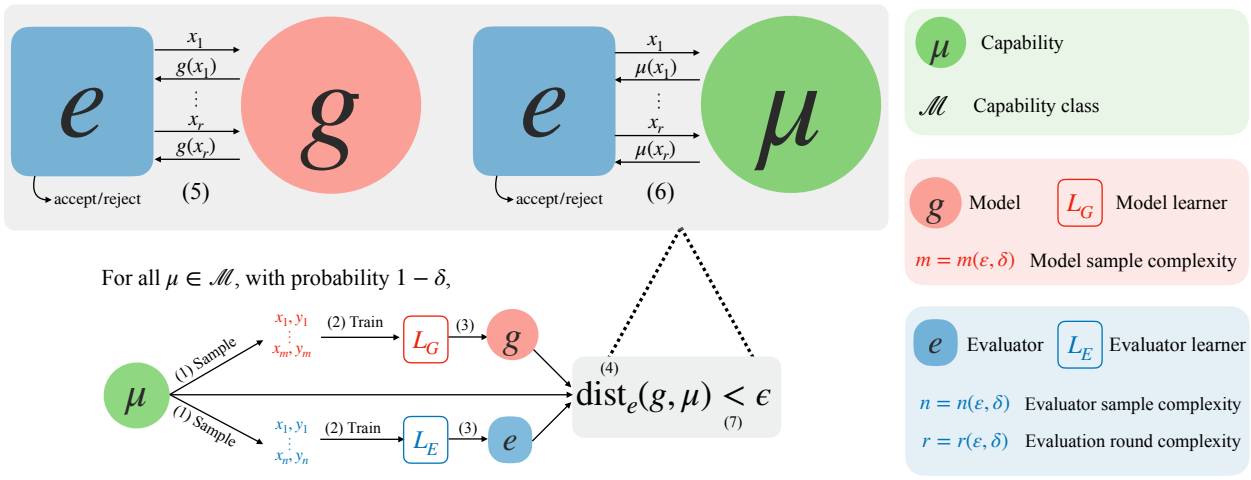


Figure 3.1: **Targeted evaluation of a pseudointelligent model.** For each capability μ, (1) iid samples are drawn and (2) fed to the learners, which (3) output a model and an evaluator. (4) The distinction $\text{dist}_e(g, \mu)$ is computed as the expected difference in evaluator output during a multi-round interaction with (5) the model g versus (6) the ground-truth capability μ. (7) If $\text{dist}_e(g, \mu) < \varepsilon$ with probability[1]greater than $(1 - \delta)$, we say that L_G is pseudointelligent against L_E w.r.t capabilities $\mathcal{M}$. See Definition 3.3 for a formal definition. Note that the targeted evaluator is trained on samples from the capability μ, and adaptively interacts with the model g.

(intelligent) *capabilities* (Section 3.1.1).

P2: Increased resources dedicated to model development should be accompanied by *increased resources dedicated to evaluation.* These include the number of examples of the capability, and the complexity of the space of possible models and evaluators (Section 3.1.2).

P3: *Self-evaluation* cannot support a claim of intelligence if the evaluator is directly derived from the model. It might, however, be useful as means towards a different end (Section 3.1.3).

Besides laying the foundation for theoretical analysis, our framework also provides a *unifying lens* on existing evaluation methods (Section 3.2).

Background: Pseudorandomness and Turing's Imitation Game

First, a brief introduction of pseudorandomness, which forms the conceptual backbone of our framework. See Goldreich (2008b) for an extended introduction.

[1]Over the samples from μ, and any randomness used by the learners L_G, L_E, model g and evaluator e.

Tessa and Obi are playing a game, and would like to decide who gets to go first. They agree to make the decision based on a coin toss: Tessa tosses a coin, and Obi calls *Heads* or *Tails*. If Obi calls the outcome correctly he gets to go first, and otherwise Tessa does. Now consider two cases:

1. Obi is calling the coin based only on the information available to him from eyesight.
2. Obi has access to an array of sensors that measure the initial conditions of Tessa's coin toss, and a powerful computer that can perform complicated calculations in a millisecond.

Tessa would not be happy with a coin toss in the second case, because Obi could call the coin correctly with ease. In other words, the coin toss is no longer "random-enough" due to Obi's increased computational power. More generally, a distribution is *pseudorandom against a particular observer* if she cannot distinguish it from a truly random. Formally,

Definition 3.1. *Fix $\varepsilon \in (0,1)$ and a finite set $\mathcal{X}$. Let $\mathcal{U}_\mathcal{X}$ denote the uniform distribution over $\mathcal{X}$. A distribution $\mathcal{P}$ over $\mathcal{X}$ is ε-pseudorandom against a class of distinguishers D if for every $d \in D$,*

$$\left| \Pr_{x \sim \mathcal{P}} [d(x) \text{ accepts }] - \Pr_{x \sim \mathcal{U}_\mathcal{X}} [d(x) \text{ accepts }] \right| < \varepsilon.$$

One can view Definition 3.1 as consisting of an *ideal* source (uniformly random elements), and a *pseudoideal* approximation to this source (pseudorandom elements). Unlike randomness, intelligence does not have a canonical mathematical operationalization.

Next, we revisit the Imitation Game of Turing (1950). In the Game, also known as the Turing Test, an evaluator converses with either a machine or a human; the machine attempts to convince the evaluator that it is human, while the evaluator aims to distinguish machine from human. If the machine successfully fools the evaluator, Turing argued that it should be considered as exhibiting intelligent behavior. However, while winning the Game (i.e., passing the Test) signifies that the machine is indistinguishable from human by a particular evaluator, it alone does not imply human-level learning or comprehension (independent of an evaluator). Pseudointelligence is defined with this intuition in mind; however, it *explicitly* requires specifying the particular evaluator and (intelligent) capabilities against which the machine is measured.

3.1 Defining Pseudointelligence

Our main message (**P1**) is that claims of intelligence should center the evaluator, and not just the (allegedly) intelligent model. Put differently, a claim that a model is intelligent is actually a claim that it is "intelligent-enough," therefore it is meaningful only with respect to a specific class of evaluators. We provide a complexity-theoretic framework in which evaluators are placed front and center, formalizing Figure 3.1.

3.1.1 Setup

A *model* is a (possibly randomized) mapping $g\colon \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ is a set of *queries* and $\mathcal{Y}$ is a set of responses.

A *capability* is a distribution μ over $\mathcal{X} \times \mathcal{Y}$. For a given query $x \in \mathcal{X}$, we let $\mu(x)$ denote a sample from the conditional distribution on acceptable responses $\mu(\cdot \mid x)$; thus, μ can be thought of as the ground-truth randomized mapping $\mu\colon \mathcal{X} \to \mathcal{Y}$ against which models are evaluated.

In this work, we study the *perceived intelligence* of a model. That is, how well a model appears to posses certain capabilities *as perceived by an evaluator*.[2] We formalize this by considering an evaluator e which is an algorithm that is given black-box access to the model g; for each of $i \in [r]$ rounds, the evaluator queries g on x_i to receive response y_i; finally, the evaluator "accepts" g if it thinks it is the ground-truth capability, and rejects it otherwise. Note that the query x_i may depend on previous responses $y_1, \ldots, y_{i-1}$.

The degree to which an evaluator e is able to distinguish between the model g and a (ground-truth) capability μ is defined next.

Definition 3.2 (Distinction)**.** *Let e be an evaluator, $g\colon \mathcal{X} \to \mathcal{Y}$ be a model and μ be a capability over $\mathcal{X} \times \mathcal{Y}$. For any $\varepsilon \in (0,1)$, we say that e can* ε-distinguish *between g and μ if*

$$\underbrace{\left|\Pr\left[e \text{ accepts } g\right] - \Pr\left[e \text{ accepts } \mu\right]\right|}_{\operatorname{dist}_e(g,\mu)} > \varepsilon.$$

If $\operatorname{dist}_e(g,\mu) \le \varepsilon$ then we say that e cannot ε-distinguish *between g and μ.*

The distinction $\operatorname{dist}_e(g,\mu)$ captures the likelihood that an evaluator distinguishes a *given* model g from the (ground-truth) capability μ. However, intelligence is not the same as possessing a particular capability (Gunderson and Gunderson, 2008). Rather, we view it as an ability to *learn* various capabilities. Thus, we consider a learner L_G that learns a model $g \in G$ from finite samples of μ.

We will say that the learner is pseudointelligent if, with high probability, the evaluator cannot distinguish between the learned model and the capability. Lastly, to allow for *targeted evaluation* of the capability, we consider an evaluator learner L_E that is also given (different) samples from the capability, and outputs an evaluator $e \in E$ targeted at it.

Definition 3.3 (Pseudointelligence)**.** *Fix a query set $\mathcal{X}$, response set $\mathcal{Y}$, and a class of capabilities $\mathcal{M}$. Fix sample complexity functions $m, n\colon (0,1)^2 \to \mathbb{N}$. Given a model class $\mathcal{G} = (G, L_G, m)$ and an evaluator class $\mathcal{E} = (E, L_E, n)$, we say that $\mathcal{G}$ is* pseudointelligent *with respect to $\mathcal{E}$ and capabilities $\mathcal{M}$ if, for any $\varepsilon, \delta \in (0,1)$, whenever L_G (resp. L_E) is given $m := m(\varepsilon,\delta)$ (resp. $n := n(\varepsilon,\delta)$) iid samples from μ, with probability at least $1-\delta$,[3] L_G and*

[2] We prefer *evaluator* over *benchmark* as it emphasizes its role as an active participant in an interaction, rather than a passive dataset.

[3] *Ibid.*, 1.

L_E output model g and evaluator e such that e cannot ε-distinguish between g and μ:

$$\forall \mu \in \mathcal{M} \Pr_{\substack{g \sim L_G \circ \mu^m \\ e \sim L_E \circ \mu^n}} [\text{dist}_e(g, \mu) \leq \varepsilon] \geq 1 - \delta.$$

Note that the number of rounds of interaction between the evaluator e and the model g (denoted $r := r(\varepsilon, \delta)$ in Figure 3.1), also scales with ε and δ. Next, we examine two case-studies to understand the effect of the implicit parameters in Definition 3.3 on the validity of claims of intelligence.

3.1.2 Model resources vs. evaluator resources

Our main message (**P2**) underscores the importance of resources allocated to the evaluator relative to those allocated to the model. There are several axes on which this comparison can be made:

Samples. To evaluate capabilities $\mathcal{M}$ within error δ and distinction ε, the model learner is given $m(\varepsilon, \delta)$ samples and the evaluator learner is given $n(\varepsilon, \delta)$ samples of each capability $\mu \in \mathcal{M}$. How do each of these grow as a function of δ and ε?

Learner expressivity. The model learner L_G outputs a model $g \in G$, and the evaluator learner L_E outputs an evaluator $e \in E$. How expressive is the class of possible models G as compared to the class of possible evaluators E? A naive measure of expressivity compares the number of parameters needed to encode each: $\log |G|$ vs. $\log |E|$. Supervised learning theory has more refined measures that can be applied to infinite spaces and provide tighter bounds (Natarajan, 1989; Daniely and Shalev-Shwartz, 2014). While these measures can be applied to the model class, new measures must be developed to capture evaluator classes.

Learner compute resources. How much computational power is used to train L_G and L_E? Note that learner expressivity is concerned only with the existence of a model $g \in G$ that is indistinguishable by the evaluator, but not with how to find it. This search takes compute resources; the amount of resources available to L_G vs. L_E affects the outcome of the evaluation.

Model and evaluator computational power. Given a query $x \in \mathcal{X}$, how much computational power is needed to compute a response $g(x)$? On the evaluator side, how much power is needed to compute the ith query issued by the evaluator, given the preceding $(i-1)$ queries and responses? Additionally, given a full evaluation $(x_i, y_i)_{i=1}^r$, how much power is needed by the evaluator to decide whether it accepts?

3.1.3 Should a model evaluate itself?

One particularly interesting case is when the model is pitted against itself by playing a dual rule: both model *and evaluator*. Self-evaluation can be used to assist human evaluators (Saunders et al., 2022) or to increase model "honesty" (Kadavath et al., 2022). The validity of self-evaluation for claims of intelligence remains contested (cf. Zhang et al., 2023 and the discussion around it), and is the focus of this case study.

To consider self-evaluation in our framework, we first map models onto evaluators $g \mapsto e_g$.[4] Once such a mapping is fixed, we map a model learner L_G to an evaluator learner L_{E_G} that, given samples $S \sim \mu^n$, computes $g \sim L_G(S)$ and outputs e_g.

Can L_G be pseudointelligent with respect to L_{E_G}? This is akin to asking whether L_G is pseudointelligent *with respect to itself*. This brings us to a crucial detail of our framework: For self-evaluation to fit in our framework, L_{E_G} and L_G should receive *independent* samples from μ. This is in stark contrast to the existing practice of deriving the evaluator directly from the trained model $\hat{g} \mapsto e_{\hat{g}}$ (Kadavath et al., 2022; Saunders et al., 2022; Zhang et al., 2023). Our main message (**P3**) is that this does *not* show that L_G is pseudointelligent—although it may be useful as means towards a different end, as in Kadavath et al. (2022) and Saunders et al. (2022).

3.2 Existing evaluation methods through the lens of Pseudointelligence

Pseudointelligence can serve a *unifying* role by allowing a direct comparison between different evaluation methods. We cast several existing evaluation paradigms into our framework.

Static Datasets. The evaluator memorizes samples drawn from the capability, and queries its black box on a random sample: Given samples $S \sim \mu^n$, L_E outputs an evaluator e_S that draws a sample $(x, y) \sim S$ at random, queries the black box on x, and accepts if and only if the response was y. Clearly, like all inductive inference settings, an evaluator can be fooled by any pseudo-intelligent model that just happens to get the correct labels by learning simple shortcuts.

Adversarial Evaluation (AE). AE requires access to some *auxiliary model* $\hat{g}$ that L_E can use to search for a challenge test set, which can then be used by an evaluator. Concretely, given seed samples S and an auxiliary model $\hat{g}$, L_E filters out all examples where $\hat{g}$ outputs the correct response, thereby creating a challenge test set $\hat{S}$. Such a filtering process can be done in several rounds, where human annotators modify an initial query until $\hat{g}$ makes an

[4]For example, consider the case that g models yes-no questions ($\mathcal{Y} = \{0, 1\}$). Then one can obtain an evaluator e_g from a model g by sampling a query x, querying the black box to receive a response y, and accepting if and only if $g(x) = y$.

error (Bartolo et al., 2022). Intuitively, based on the quality of $\hat{g}$, such filtering can create increasingly hard datasets. Thus, the central *resources* here are the amount of seed samples S and the complexity of the auxiliary model $\hat{g}$.

Model-based Evaluation. These evaluators also use an auxiliary $\hat{g}$, albeit in a non-adversarial way. For instance, Ribeiro et al. (2021) use human-generated templates, filled in by a language model, as queries. Perez et al. (2023) use two auxiliary models: one to generate queries, and the other to find those targeted at a particular capability.

3.3 Conclusion

This brief chapter introduces a principled framework for model evaluation, inspired by the theory of pseudorandomness. Our main message is that claims about model capability must be supplemented with a thorough discussion of the *resources* given to the evaluator, especially in settings where model resources are largely unknown (e.g. OpenAI, 2023). Central to our framework is a model-based evaluator that is targeted at specific capabilities as well as specific models (via multi-round interactions). We hope our framework encourages rigorous analysis of LLM evaluation, and helps unify the study of this increasingly-important topic.

Part II

Translation

Chapter 4

A Theory of Unsupervised Translation

Neural networks are capable of translating between languages—in some cases even between two languages where there is little or no access to parallel translations, in what is known as Unsupervised Machine Translation (UMT). Given this progress, it is intriguing to ask whether machine learning tools can ultimately enable understanding animal communication, particularly that of highly intelligent animals. We propose a theoretical framework for analyzing UMT when no parallel translations are available and when it cannot be assumed that the source and target corpora address related subject domains or posses similar linguistic structure. We exemplify this theory with two stylized models of language, for which our framework provides bounds on necessary sample complexity; the bounds are formally proven and experimentally verified on synthetic data. These bounds show that the error rates are inversely related to the language complexity and amount of common ground. This suggests that unsupervised translation of animal communication may be feasible if the communication system is sufficiently complex.

Recent interest in translating animal communication (Andreas et al., 2022b; Anthes, 2022; Berthet et al., 2022) has been motivated by breakthrough performance of Language Models (LMs). Empirical work has succeeded in unsupervised translation between human-language pairs such as English–French (Lample et al., 2018a; Artetxe, Labaka, and Agirre, 2019) and programming languages such as Python–Java (Rozière et al., 2022). Key to this feasibility seems to be the fact that language statistics, captured by a LM (a probability distribution over text), encapsulate more than just grammar. For example, even though both are grammatically correct, *The calf nursed from its mother* is more than 1,000 times more likely than *The calf nursed from its father*.[1]

Given this remarkable progress, it is natural to ask whether it is possible to collect and analyze animal communication data, aiming towards translating animal communication to a human language description. This is particularly interesting when the source language may be of highly social and intelligent animals, such as whales, and the target language is a human language, such as English.

[1]Probabilities computed using the GPT-3 API `https://openai.com/api/` text-davinci-02 model.

A	B	C
Have you seen any orcas today? I just got back from the reef. $p(A_1) \approx 10^{-18}$ *At the reef, there were a lot of sea turtles.* $p(A) \approx 10^{-22}$	*Have you seen mom? I just returned from the ocean basin.* $p(B_1) \approx 10^{-18}$ *At the reef, there were a lot of sea turtles.* $p(B) \approx 10^{-26}$	*Hat out hat dsjgh!!!* $p(C_1) \approx 10^{-48}$ *bicycle OMG and.* $p(C) \approx 10^{-72}$

Figure 4.1: **LMs identify incoherent text.** The probabilities of three two-paragraph texts computed using the GPT-3 API. The probabilities of just the first paragraphs A_1, B_1, C_1 are also shown. Although $p(A_1) \approx p(B_1)$ and the second paragraphs of A and B are identical, overall $p(A) \gg p(B)$ due to coherence between the paragraphs. C is gibberish.

Challenges. The first and most basic challenge is *understanding the goal*, a question with a rich history of philosophical debate (Wittgenstein, 1953). To define the goal, we consider a hypothetical ground-truth translator. As a thought experiment, consider a "mermaid" fluent in English and the source language (e.g. sperm whale communication). Such a mermaid could translate whale vocalizations that English naturally expresses. An immediate worry arises: what about communications that the source language may have about topics for which English has no specific words? For example, sperm whales have a sonar sense which they use to perform echolocation. In that case, lacking a better alternative, the mermaid may translate such a conversation as *(something about echolocation)*.[2]

Thus, we formally define the goal to be to achieve translations similar to those that would be output by a hypothetical ground-truth translator. While this does not guarantee functional utility, it brings the general task of unsupervised translation and the specific task of understanding animal communication into the familiar territory of supervised translation, where one can use existing error metrics to define (hypothetical) error rates.

The second challenge is that animal communication is unlikely to share much, if any, linguistic structure with human languages. Indeed, our theory will make no assumption on the source language other than that it is presented in a textual format. That said, one of our instantiations of the general theory (the knowledge graph) shows that translation is easier between compositional languages.

The third challenge is *domain gap*, i.e., that ideal translations of animal communications into English would be semantically different from existing English text, and we have no precise prior model of this semantic content. (In contrast, the distribution of English translations of French text would resemble the distribution of English text.) Simply put: whales do not "talk" about smartphones. Instead, we assume the existence of a *broad prior* that models plausible English translations of animal communication. LMs assign likelihood

[2]A better description might be possible. Consider the fact that some people who are congenitally blind comprehend vision-related verbs as accurately as sighted people (Bedny et al., 2019).

to an input text based not only on grammatical correctness, but also on agreement with the training data. In particular, LMs trained on massive and diverse data, including some capturing facts about animals, may be able to reason about the plausibility of a candidate translation. See Figure 4.1 and the discussion in Section 4.7.

Overview of the framework and results

A translator[3] is a function $f\colon \mathcal{X} \to \mathcal{Y}$ that translates source text $x \in \mathcal{X}$ into the target language $f(x) \in \mathcal{Y}$. We focus on the easier-to-analyze case of *lossless* translation, where f is invertible (one-to-one) denoted by $f_\theta : \mathcal{X} \hookrightarrow \mathcal{Y}$. See Section 4.6.1 for an extension to lossy translation.

We will consider a parameterized family of translators $\{f_\theta\colon \mathcal{X} \to \mathcal{Y}\}_{\theta\in\Theta}$, with the goal being to learn the parameters $\theta \in \Theta$ of the most accurate translator. Accuracy (defined shortly) is measured with respect to a hypothetical *ground-truth* translator denoted by $f_\star : \mathcal{X} \to \mathcal{Y}$. We make a *realizability* assumption that the ground-truth translator can be represented in our family, i.e., $\star \in \Theta$.

The source language is defined as a distribution μ over $x \in \mathcal{X}$, where $\mu(x)$ is the likelihood that text x occurs in the source language. The error of a model $\theta \in \Theta$ will be measured in terms of $\mathrm{err}(\theta) := \Pr_{x\sim\mu}[f_\theta(x) \neq f_\star(x)]$, or at times a general bounded loss function $\mathcal{L}(\theta)$. Given μ and $f_\star$, it will be useful to consider the *translated language distribution* τ over $\mathcal{Y}$ by taking $f_\star(x)$ for $x \sim \mu$.

In the case of similar source and target domains, one may assume that the target language distribution ν over $\mathcal{Y}$ is close to τ. This is a common intuition given for the "magic" behind why UMT sometimes works: for complex asymmetric distributions, there may a nearly unique transformation in $\{f_\theta\}_{\theta\in\Theta}$ that maps μ to something close ν (namely $f_\star$ which maps μ to τ). A common approach in UMT is to embed source and target text as high-dimensional vectors and learn a *low-complexity* transformation, e.g., a rotation between these Euclidean spaces. Similarly, translator complexity will also play an important role in our analysis.

Priors. Rather than assuming that the target distribution ν is similar to the translated distribution τ, we will instead assume access to a broad prior ρ over $\mathcal{Y}$ meant to capture how plausible a translation y is, with larger $\rho(y)$ indicating more natural and plausible translation. Section 4.7 discusses one way a prior oracle can be created, starting with an LM $\approx \nu$ learned from many examples in the target domain, and combined with a prompt, in the target language, describing the source domain.

We define the problem of *unsupervised machine translation (with a prior)* to be finding an accurate $\theta \in \Theta$ given m iid unlabeled source texts $x_1, \ldots, x_m \sim \mu$ and oracle access to prior ρ.

[3]In this work, a *translator* refers to the function $f\colon \mathcal{X} \to \mathcal{Y}$ while *translation* refers to an output $y = f(x)$.

MLE. Our focus is on the Maximum-Likelihood Estimator (MLE), which selects model parameters $\theta \in \Theta$ that (approximately) maximize the likelihood of translations $\prod_i \rho\big(f_\theta(x_i)\big)$.

Definition 4.1 (MLE)**.** *Given input a translator family* $\{f_\theta \colon \mathcal{X} \to \mathcal{Y}\}_{\theta\in\Theta}$*, samples* $x_1, \ldots, x_m \in \mathcal{X}$ *and a distribution* ρ *over* $\mathcal{Y}$*, the MLE outputs*

$$MLE^\rho(x_1, x_2, \ldots, x_m) \coloneqq \operatorname*{argmin}_{\theta\in\Theta} \sum_{i=1}^{m} \log \frac{1}{\rho(f_\theta(x_i))}$$

If multiple θ *have equal empirical loss, it breaks ties, say, lexicographically.*

We note that heuristics for MLE have proven extremely successful in training the breakthrough LMs, even though MLE optimization is intractable in the worst case.

Next, we analyze the efficacy of MLE in two complementary models of language: one that is highly structured (requiring compositional language) and one that is completely unstructured. These analyses both make strong assumptions on the target language, but make few assumptions about the source language itself. In both cases, the source distributions are uniform over subsets of $\mathcal{X}$, which (informally) is the "difficult" case for UMT as the learner cannot benefit from similarity in text frequency across languages. Both models are parameterized by the amount of "common ground" between the source language and the prior, and both are randomized. Note that these models are *not* intended to accurately capture natural language. Rather, they illustrate how our theory can be used to study the effect of language similarity and complexity on data requirements for UMT.

Knowledge graph model. Our first model consists of a pair of related *knowledge graphs*, in which edges encode knowledge of binary relations. Each edge yields a text that described the knowledge it encodes. For example, in Figure 4.2 edges encode which animal A eats which other animal B, and text is derived as a simple description A *eats* B.[4]

Formally, there are two Erdős–Rényi random digraphs. The target graph is assumed to contain n nodes, while the source graph has $r \leq n$ nodes corresponding to an (unknown) subset of the n target nodes. The model has two parameters: the average degree d in the (randomly-generated) target language graph, and the agreement $\alpha \in (0, 1]$ between the source and the target graphs. Here $\alpha = 1$ is complete agreement on edges in the subgraph, while $\alpha = 0$ is complete independence. We assume the languages use a *compositional* encoding for edges, meaning that they encode a given edge by encoding both nodes, so we may consider only $|\Theta| = n!/(n - r)!$ translators consistently mapping the r source nodes to the n target nodes, which is many fewer than the number of functions $f : \mathcal{X} \to \mathcal{Y}$ mapping the $|\mathcal{X}| = O(r^2)$ source edges into the $\mathcal{Y} = O(n^2)$ target edges. Human languages as well as the communication systems of several animals are known to be compositional (Zuberbühler, 2020).[5]

[4] In a future work, it would be interesting to consider k-ary relations (hypergraphs) or multiple relations.

[5] A system is *compositional* if the meaning of an expression is determined by the meaning of its parts.

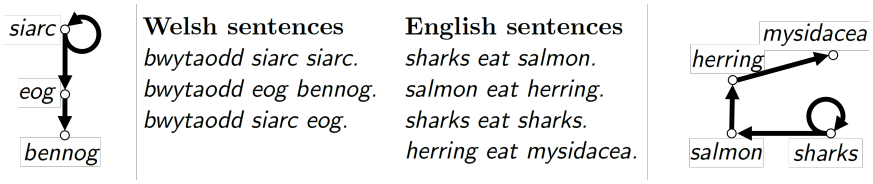


Figure 4.2: **An illustration of the knowledge graph model.** In this example, the Welsh graph is an exact subgraph of the English knowledge graph, but our model allows for differences.

We analyze how the error of the learned translator depends on this "common knowledge" α:

Theorem 4.2 (Theorem 4.18, simplified)**.** *Consider a source language μ and a prior ρ generated by the knowledge graph model over r source nodes, n target nodes, average degree d and agreement parameter α. Then, with at least 99% probability, when given m source sentences $x_1, \ldots, x_m$ and access to a prior ρ, MLE outputs a translator $\hat{\theta}$ with error*

$$\operatorname{err}(\hat{\theta}) \leq O\left(\frac{\log n}{\alpha^2 d} + \frac{1}{\alpha}\sqrt{\frac{r \log n}{m}}\right).$$

The second term decreases to 0 at a $O(m^{-1/2})$ rate, similar to (noisy) generalization bounds (Mohri, Rostamizadeh, and Talwalkar, 2018). Note that the first term does not decrease with the number of samples. The average degree d is a rough model of language complexity capturing per-node knowledge, while the agreement parameter α captures the amount of common ground. Thus, more complex languages can be translated (within a given error rate) with less common ground. Even with $m = \infty$ source data, there could still be errors in the mapping. For instance, there could be multiple triangles in the source and target graphs that lead to ambiguities. However, for complex knowledge relations (degree $d \gg 1/\alpha^2$), there will be few such ambiguities.

Figure 4.2 illustrates an example of four English sentences and three sentences (corresponding to an unknown subset of three of the English sentences) in Welsh. For UMT, one might hypothesize that *bwytaodd* means *eat* because they both appear in every sentence. One might predict that *siarc* means *shark* because the word *siarc* appears twice in a single Welsh sentence and only the word shark appears twice in an English sentence. Next, note that *eog* may mean *salmon* because they are the only other words occurring with *siarc* and *shark*. Similar logic suggests that *bennog* means *herring*. Furthermore, the word order is consistently permuted, with subject-verb-object in English and verb-subject-object in Welsh. This translation is indeed roughly correct. This information is encoded in the directed graphs as shown, where each node corresponds to an animal species and an edge between two nodes is present if the one species eats the other.

"Common nonsense" model. The second model, the *common nonsense* model, assumes no linguistic structure on the source language. Here, we set out to capture the fact that the translated language $\tau = \mu \circ f_\star$ and the prior ρ share some common ground through the fact that the laws of nature may exclude common "nonsense" outside both distributions' support.

Earlier work has justified *alignment* for UMT under the intuition that the target language distribution ν is approximated by a nearly unique simple transformation, e.g., a rotation, of the source distribution μ. However, for a prior ρ, our work suggests that UMT may also be possible if there is nearly a unique simple transformation that maps τ so that it is *contained* in ρ. Figure 4.3 illustrates such a nearly unique rotation—the UMT "puzzle" of finding a transformation f_θ of μ which is contained within ρ is subtly different from finding an alignment.

In the common nonsense model, τ and ρ are uniform over arbitrary sets $T \subseteq P \subseteq \mathcal{Y}$ from which a common $\alpha \in (0, 1/2]$ fraction of text is removed (hence the name "common nonsense"). Specifically, τ and ρ are defined to be uniform over $\tilde{T} = T \setminus S$, $\tilde{P} = P \setminus S$, respectively, for a set S sampled by including each $y \in \mathcal{Y}$ with probability α.

We analyze the error of the learned translator as a function of the amount of common nonsense:

Theorem 4.3 (Theorem 4.13, simplified)**.** *Consider source language μ and a prior ρ generated by the common nonsense model over $|T|$ source texts and common-nonsense parameter α, and a translator family parameterized by $|\Theta|$. Then, with at least 99% probability, when given m source sentences $x_1, \ldots, x_m$ and access to a prior ρ, MLE outputs a translator $\hat{\theta}$ with error*

$$\operatorname{err}\left(\hat{\theta}\right) \coloneqq \Pr_{x \in X}\left[f_{\hat{\theta}}(x) \neq f_\star(x)\right] = O\left(\frac{\ln|\Theta|}{\alpha \min(m, |T|)}\right).$$

Theorem 4.14 gives a nearly matching lower bound. Let us unpack the relevant quantities. First, we think of α as measuring the amount of agreement or common ground required, which might be a small constant. Second, note that $|T|$ is a coarse measure of the complexity of the source language, which requires a total of $\tilde{O}(|T|)$ bits to encode. Thus, the bound suggests that accurate UMT requires the translator to be simple, with a description length that is an α-factor of the language description length, and again α captures the agreement between τ, ρ. Thus, even with limited common ground, one may be able to translate from a source language that is sufficiently complex. Third, for simplicity, we require $\mathcal{X}, \mathcal{Y} \subset \{0,1\}^*$ to be finite sets of binary strings, so WLOG Θ may be also assumed to be finite. Thus, $\log_2|\Theta|$ is the *description length*, a coarse but useful complexity measure that equals the number of bits required to describe any model. (Neural network parameters can be encoded using a constant number of bits per parameter.) Section 4.6.2 discusses how this can be generalized to continuous parameters.

Importantly, we note that (supervised) neural machine translators typically use far fewer parameters than LMs.[6] To see why, consider the example of the nursing calf (page 1) and

[6]For example, a multilingual model achieves state-of-the-art performance using only 5 billion parameters

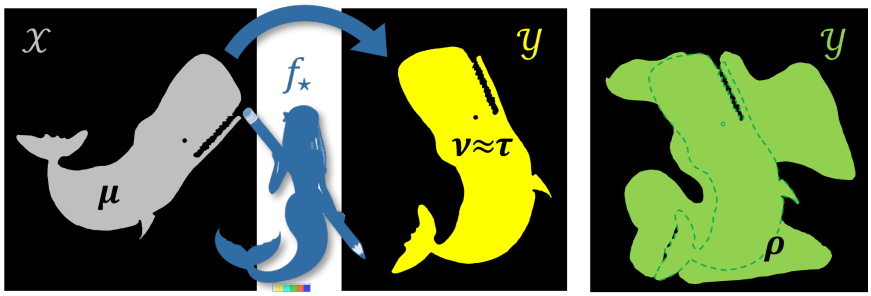


Figure 4.3: The previous intuition behind UMT has the distributions of target language ν (middle) close to ground-truth translations τ, which is assumed to be a low-complexity transformation (in this example a rotation) of the source language μ (left). When source and target are not aligned, restricting to *prior* ρ region (right) allows for translation, as long as there are enough "nonsense" texts (black regions) so that there is a nearly unique rotation of μ that is contained in ρ. For example, both distributions may assign negligible probability to nonsensical texts such as *I died 3 times tomorrow*. (In this toy example, μ is uniform over a two-dimensional shape that happens to look like a whale.)

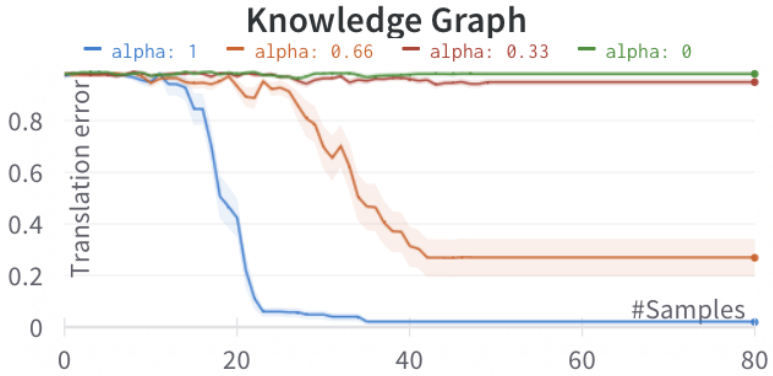
the fact that a translator needs not know that calves nurse from mothers. On the other hand, such knowledge is essential to generate realistic text. Similarly, generating realistic text requires maintaining coherence between paragraphs, while translation can often be done at the paragraph level.

As a warm-up, we include a simplified version of the common nonsense model, called the *tree-based model* (Section 4.3), in which texts are constructed word-by-word based on a tree structure.

Comparison to supervised classification. Consider the dependency on m, the number of training examples. Note that the classic Occam bound $O\left(\frac{1}{m}\log|\Theta|\right)$ is what one gets for noiseless supervised classification, that is, when one is also given labels $y_i = f_\star(x_i)$ at training time, which is similar to Theorem 4.3, and give $\tilde{O}(m^{-1/2})$ bounds for noisy classification as in Theorem 4.2. Furthermore, these bounds apply to translation, which can be viewed as a special case of classification with *many* classes $\mathcal{Y}$. Thus, in both cases, the data dependency on m is quite similar to that of classification.

Experiments. We validate our theorems generating synthetic data from randomly-generated languages according to each model, and evaluating translator error as a function of the number of samples and amount of common ground. The knowledge graph model (Figure 4.4, left) is used to generate a source graph (language) on $r = 9$ nodes to a target graph (language) on $n = 10$ nodes and average degree $d \approx 5$, while varying the agreement parameter α. We

(Tran et al., 2021), compared to 175 billion for GPT-3 (Brown et al., 2020).



r	Accuracy
4	0.10 ± 0.05
7	0.74 ± 0.08
10	1.00 ± 0.00

Figure 4.4: **Knowledge Graph model experiments, each run on twenty seeds with standard errors shown.** Left: error of the top-scoring translator vs. number of source samples m. Right: effect of source language complexity (number of source nodes r) on translator accuracy in the knowledge graph model. We report the accuracy of the top-scoring translator after all source edges were input to the learning algorithm, i.e., as the number of samples $m \to \infty$.

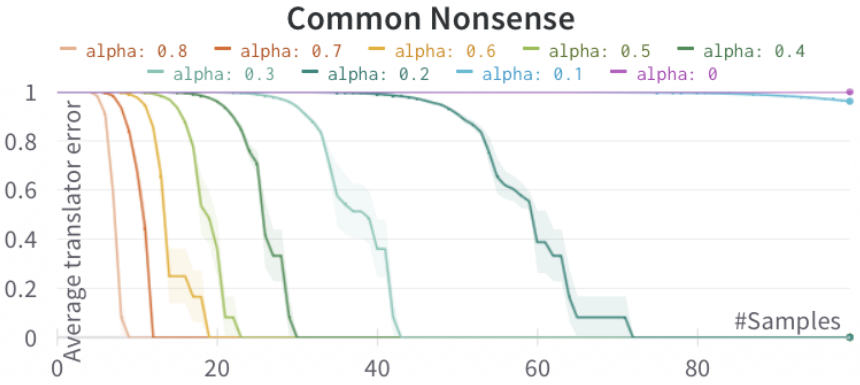


Figure 4.5: **Common Nonsense model.** The X-axis is the number of source samples m, and the Y-axis is the average error among plausible translators (that have not been ruled-out so far). Each experiment was run on five seeds, with standard error depicted by the shaded area.

also vary r (Figure 4.4, right) supporting our main message: more complex languages can be translated more accurately. For the common nonsense model (Figure 4.5) we simulate translation of a source language of size $|T| = 10^5$ while varying the fraction of common nonsense α. Sections 4.4 and 4.5 contain full details of the respective experimental setups.

Contributions. The first contribution of this work is formalizing and analyzing a model of UMT. As an initial work, its value is in the opportunities which it opens for further

work more than the finality and tightness/generality of its bounds. Our model applies even to low-resource source languages with massive domain gap and linguistic distance. We emphasize that this work is only *a first step* in the theoretical analysis of UMT (indeed, there is little theoretical work on machine translation in general). Second, we exhibit two simple complementary models for which we prove that: (a) more complex languages require less common ground, and (b) data requirements may not be significantly greater than those of supervised translation (which tends to use less data than training a large LM). These findings may have implications for the quantity and type of communication data that is collected for deciphering animal communication and for UMT more generally. They also give theoretical evidence that UMT can be successful and worth pursuing, in lieu of parallel (supervised) data, in the case of sufficiently complex languages. All of that said, we note that our sample complexity bounds are information theoretic, that is, they do not account for the computational complexity of optimizing the translator. Finally, animal communication aside, to the best of our knowledge this work is the first theoretical treatment of UMT, and may also shed light on translation between human languages.

4.1 The framework

We use $f\colon \mathcal{X} \hookrightarrow \mathcal{Y}$ to denote a 1–1 function, in which $f(x) \neq f(x')$ for all $x \neq x'$. For $S \subseteq \mathcal{X}$, we write $f(S) \coloneqq \{f(x) \mid x \in S\}$. The indicator $\mathbf{1}_P$ is 1 if the predicate P holds, and 0 otherwise. The uniform distribution over a set S is denoted by $\mathcal{U}(S)$, and $\log = \log_2$ denotes base-2 logarithm.

Language and Prior. A *source language* is a distribution μ over a set of possible texts $\mathcal{X}$. Similarly, a *target language* is a distribution ν over a set of possible texts $\mathcal{Y}$. When clear from context, we associate each language with its corresponding set of possible texts. A *prior* distribution ρ over translations $\mathcal{Y}$ aims to predict the probability of observing each translation. One could naively take $\rho = \nu$, but Section 4.7 describes how better priors can focus on the domain of interest. Intuitively, $\rho(y)$ measures how "plausible" a translation y is. For simplicity, we assume that $\mathcal{X}, \mathcal{Y} \subseteq \{0,1\}^*$ are finite, non-empty sets of binary strings. Section 4.6.2 discusses extensions to infinite sets.

Translators. A *translator* is a mapping $f\colon \mathcal{X} \hookrightarrow \mathcal{Y}$. There is a known set of 1–1 functions $\{f_\theta\colon \mathcal{X} \hookrightarrow \mathcal{Y} \mid \theta \in \Theta\}$ with *parameter* set Θ. Since parameters are assumed to be known and fixed, we will omit them from the theorem statements and algorithm inputs, for ease of presentation. Like $\mathcal{X}, \mathcal{Y}$, the set Θ is assumed to be finite. Section 4.6.1 considers translators that are not 1–1.

Divergence. A translator f_θ and a distribution μ induce a distribution over $y = f_\theta(x)$, which we denote by $f_\theta \circ \mu$. The *divergence* between this distribution and ρ is quantified using

the Kullback–Leibler (KL) divergence,

$$\mathrm{D}(\theta) \coloneqq \mathrm{KL}(f_\theta \circ \mu \parallel \rho) = \mathop{\mathbb{E}}_{x\sim\mu}\left[\log \frac{\mu(x)}{\rho\big(f_\theta(x)\big)}\right] = \sum_x \mu(x) \log \frac{\mu(x)}{\rho\big(f_\theta(x)\big)} \geq 0.$$

Note that since $\mathrm{D}(\theta) = \mathbb{E}_{x\sim\mu}\big[-\frac{1}{m}\sum \log \rho(f_\theta(x_i))\big] - \mathrm{H}(\mu)$, and $\mathrm{H}(\mu)$ is a constant independent of θ, the MLE of Definition 4.1 approximately minimizes divergence.

Ground truth. In order to define semantic loss, we consider a *ground-truth translator* $f_\star$ for some $\star \in \Theta$. We can then define the (ground-truth) *translated language* $\tau = f_\star \circ \mu$ over $\mathcal{Y}$, obtained by taking $f_\star(x)$ for $x \sim \mu$. This is similar to the standard realizability assumption, and some of our bounds resemble Occam bounds with training labels $y_i = f_\star(x_i)$. Of course, the ground-truth translator $\star$ is *not* known to the unsupervised learning algorithm. In our setting, we further require that ground-truth translations never have 0 probability under ρ:

Definition 4.4 (Realizable prior)**.** $\Pr_{x\sim\mu}[\rho(f_\star(x)) = 0] = 0$*, or equivalently* $\mathrm{D}(\star) < \infty$*.*

Semantic loss. The semantic loss of a translator is defined with respect to a *semantic difference* function $\ell\colon \mathcal{Y} \times \mathcal{Y} \to [0, 1]$. This function, unknown to the learner, measures the difference between two texts from the target language $\mathcal{Y}$, with $\ell(y, y) = 0$ for all y. For a given semantic difference ℓ and ground-truth translator $f_\star$, we define the *semantic loss* of a translator f_θ by

$$\mathcal{L}(\theta) \coloneqq \mathop{\mathbb{E}}_{x\sim\mu}\big[\ell(f_\star(x), f_\theta(x))\big].$$

Of particular interest to us is the *semantic error* $\mathrm{err}(\cdot, \cdot)$, obtained when ℓ is taken to be the 0-1 difference $\ell_{01} = (y, y') = 1$ for all $y' \neq y$. Note that since any semantic difference ℓ is upper bounded by 1, the semantic error upper-bounds any other semantic loss $\mathcal{L}$. That is,

$$\mathcal{L}(\theta) \leq \mathrm{err}(\theta) \coloneqq \Pr_{x\sim\mu}[f_\theta(x) \neq f_\star(x)].$$

4.2 A model-free theorem: Translator revisions and plausible ambiguities

Even though $\star$ is unknown, it will be convenient to define, for each $\theta \in \Theta$, a *revision* of $f_\star$ which is the permutation $\pi^\star_\theta\colon \mathcal{Y} \hookrightarrow \mathcal{Y}$ between f_θ and $f_\star$, $\pi^\star_\theta(y) \coloneqq f_\theta(f_\star^{-1}(y))$.[7] We write $\pi_\theta = \pi^\star_\theta$ when $\star$ is clear from context, and $\pi_\star(y) \equiv y$ is the identity revision.

[7]Formally, $f_\theta \circ f_\star^{-1}\colon f_\star(\mathcal{X}) \hookrightarrow \mathcal{Y}$ is only defined on $f_\star(\mathcal{X})$ but can be extended to a full permutation on $\mathcal{Y}$ in many ways. For concreteness, we take the lexicographically smallest extension on $\mathcal{Y} \subseteq \{0, 1\}^*$.

Note that the divergence $\mathrm{D}(\theta)$ and semantic loss $\mathcal{L}(\theta)$ can equivalently be defined using revisions,

$$\mathrm{D}(\theta) \coloneqq \mathrm{KL}(f_\theta \circ \mu \parallel \rho) = \mathrm{KL}(\tau \parallel \pi_\theta^{-1} \circ \rho) = \mathop{\mathbb{E}}_{y\sim\tau}\big[\log \frac{\tau(y)}{\rho(\pi_\theta(y))}\big],$$
$$\mathcal{L}(\theta) \coloneqq \mathop{\mathbb{E}}_{x\sim\mu}\big[\ell(f_\star(x), f_\theta(x))\big] = \mathop{\mathbb{E}}_{y\sim\tau}\big[\ell(y, \pi_\theta(y))\big].$$

To relate divergence and loss, it is helpful to define a notion of *plausible ambiguities* which are θ whose revisions $\pi_\theta(y \sim \tau)$ are not too unlikely under the prior. These may also be of independent interest as they capture certain types of ambiguities that can only be resolved with supervision.

Definition 4.5 (γ-Plausible ambiguities)**.** *For any $\gamma \in [0,1]$, $\star \in \Theta$, and distributions τ, ρ over $\mathcal{Y}$, the set of γ-plausible ambiguities $\mathcal{A}_\gamma = \mathcal{A}_\gamma^{\star,\tau,\rho} \subseteq \Theta$ is:*

$$\mathcal{A}_\gamma \coloneqq \left\{\theta \in \Theta \;\middle|\; \Pr_{y\sim\tau}[\rho(\pi_\theta(y)) = 0] \le \gamma\right\}, \text{ and } \varepsilon_\gamma \coloneqq \max_{\theta\in\mathcal{A}_\gamma} \mathcal{L}(\theta).$$

For example, *left↔right* would constitute a γ-plausible ambiguity if one could swap the two words, making a $\le \gamma$ fraction of the translations have 0 probability. Such a revision would have low loss if such swaps are considered similar in meaning. Condition 4.4 is equivalent to $\star \in \mathcal{A}_0$.

The quantity of interest is ε_γ, the maximum loss of any γ-plausible ambiguity.[8] Next, we prove that the loss of the translator output by the Maximum Likelihood Estimator is not greater than ε_γ given $m \ge \frac{1}{\gamma}\ln\frac{|\Theta|}{\delta}$ examples.

Theorem 4.6. *Let μ, ρ be probability distributions over $\mathcal{X}, \mathcal{Y}$, resp., and $f_\star$ satisfy Condition 4.4: $\Pr_\mu[\rho(f_\star(x)) = 0] = 0$. Fix any $\delta \in (0,1)$, $m \ge 1$, and let $\gamma \ge \frac{1}{m}\ln\frac{|\Theta|}{\delta}$. Then,*

$$\Pr_{x_1,\ldots,x_m\sim\mu}\left[\mathcal{L}(\hat\theta) \le \varepsilon_\gamma\right] > 1-\delta,$$

where $\hat\theta = MLE^\rho(x_1,\ldots,x_m)$ and ε_γ is from Definition 4.5.

Proof of Theorem 4.6. MLE minimizes $\hat v(\theta) \coloneqq \frac{1}{m}\sum_{i\le m}\log\frac{1}{\rho(\pi_\theta(y_i))}$ over $\theta\in\Theta$, where $y_i = f_\theta(x_i)$. By the realizable prior assumption $\hat v(\star) < \infty$. Thus, for the algorithm to fail, there must be a "bad" $\hat\theta \in B \coloneqq \{\theta \in \Theta \mid \mathcal{L}(\theta) > \varepsilon_\gamma\}$ with $\hat v(\theta) \le \hat v(\star) < \infty$. If $\rho(\pi_\theta(y_i)) = 0$ for any i, then $\hat v(\pi_\theta) = \infty$. Note that by Definition 4.5, $B \cap \mathcal{A}_\gamma = \emptyset$, thus for all $\theta \in B$: $\Pr[\rho(\pi_\theta(y)) = 0] > \gamma$ and,

$$\Pr\left[\hat v(\theta) < \infty\right] < (1-\gamma)^m \le e^{-\gamma m} \le \frac{\delta}{|\Theta|}.$$

By the union bound over $\theta \in B$, $\Pr[\exists\theta\in B\ \hat v(\theta) < \infty] < \delta$ since $|B| \le |\Theta|$. □

[8] For intuition, note that ε_γ can be bounded using γ: $\frac{\gamma}{\varepsilon_\gamma} \ge \min_{\theta\in\Theta}\Pr_{y\sim\tau}\big[\rho(\pi_\theta(y)) = 0 \mid \pi_\theta(y) \ne y\big]$.

Since ε_γ is non-decreasing in γ, as the number of examples increases the loss bound ε_γ decreases to approach ε_0. This bound is analogous to the realizable Occam bound of supervised classification, which is provided next for convenience.

4.2.1 Comparison to supervised classification

In this work, for ease of presentation, we have stated that error decreases like $O(\frac{1}{m}\log\frac{|\Theta|}{\delta})$ for noiseless supervised translation. This is based in the classic Occam bound for supervised learning:

Theorem 4.7 (Occam bounds)**.** *Let $\mathcal{X}$, $\mathcal{Y}$, be sets, $\mathcal{D}$ be a joint distribution over $\mathcal{X}\times\mathcal{Y}$, $\ell\colon \mathcal{Y}\times\mathcal{Y}\to[0,1]$ be a loss, and $f_\theta : \mathcal{X}\to\mathcal{Y}$ be a family of functions parameterized by $\theta\in\Theta$, and $\mathcal{L}(\theta) \coloneqq \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(y, f_\theta(x))]$. For any $\delta > 0$,*

$$\Pr_{(x_1,y_1),\ldots,(x_m,y_m)\sim\mathcal{D}^m}\left[\mathcal{L}\big(\hat{\theta}\big) \le \frac{1}{m}\ln\frac{|\Theta|}{\delta}\right] \ge 1-\delta \qquad \text{if } \mathcal{L}(\star) = 0,$$

$$\Pr_{(x_1,y_1),\ldots,(x_m,y_m)\sim\mathcal{D}^m}\left[\mathcal{L}\big(\hat{\theta}\big) \le \mathcal{L}(\star) + \sqrt{\frac{1}{m}\ln\frac{|\Theta|}{\delta}}\right] \ge 1-\delta \qquad \text{if } \mathcal{L}(\star) \neq 0,$$

for $\hat{\theta} \coloneqq \operatorname{argmin}_{\theta\in\Theta}\sum_i \ell(y_i, f_\theta(x_i))$.[9]

Note that supervised translation is simply classification with many classes $\mathcal{Y}$.

4.3 The tree-based model

The first model, which we view as secondary to the other two, is a simpler version of the common nonsense model (Section 4.4). It can be viewed as a "warm-up" leading up to that more general model, and we hope it helps illuminate that model by instantiating the language therein with a simple tree-based syntax.

In the tree-based model, the nodes of a tree are labeled with random words, and plausible texts (according to the prior ρ) correspond to paths from root to leaf. A translated language τ (or equivalently, a source language μ) is then derived from root-to-leaf paths in a random subtree $H \subseteq G$. We prove that, with high probability, a prior ρ and translated language τ sampled by this process satisfy Condition 4.4 and semantic error $\varepsilon_\gamma = O\big(1/\min(m, a^n)\big)$. Therefore, MLE yields a semantically accurate translator for the sampled language with high probability, for sufficiently large n.

The *random tree language* (RT) model is a randomized process for generating a tree-based source language μ, translated language τ, and prior ρ. It is parameterized by a finite

[9] As in MLE, ties can be broken arbitrarily, e.g., lexicographically. Our bounds, like the Occam bound, hold simultaneously for all minimizers. In the realizable case, the Empirical Risk Minimizer $\hat{\theta}$ will have $\sum \ell(y_i, f_\theta(x_i)) = 0$.

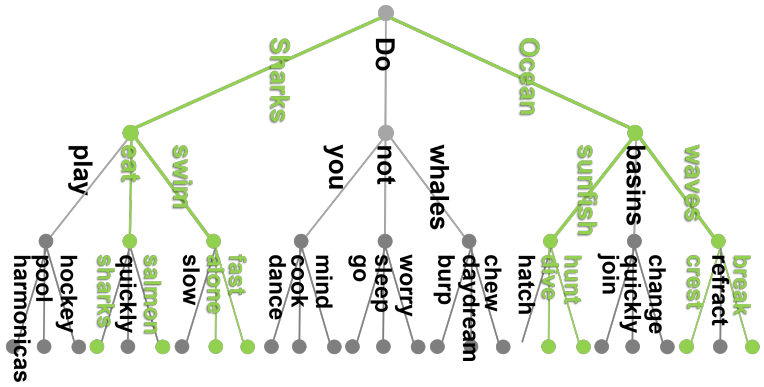


Figure 4.6: An example of a language tree of plausible texts and the subtree of ground-truth translations illustrated in green.

vocabulary set $\mathcal{W}$, depth $n \in \mathbb{N}$, and arities $a, b \in \mathbb{N}$ such that $1 \leq a \leq b \leq |\mathcal{W}|/4$. For simplicity, the possible target texts are taken to be the set of all possible n-grams over $\mathcal{W}$, namely, $\mathcal{Y} \coloneqq \mathcal{W}^n$. Again for simplicity, we also assume $|\mathcal{X}| = |\mathcal{Y}|$ so that each $f_\theta \colon \mathcal{X} \hookrightarrow \mathcal{Y}$ is a bijection. To generate an RT, first a full b-ary, depth $n+1$ tree G is constructed. Labeled edges shall correspond to words, and paths shall correspond to texts (sentences), as follows:

1. Starting from the root node and proceeding in level-order traversal, for each node v: sample b words $w_1, \ldots, w_b \in \mathcal{W}$ uniformly at random and *without* replacement; label each of v's child edges with one of the sampled words.

2. The labels on each path from the root to a leaf $(y_1, \ldots, y_n)$ corresponds to a *plausible text*, giving a set of *plausible texts*

$$P \coloneqq \{y \mid y = (y_1, \ldots, y_n) \text{ labels of a path in } G\}.$$

3. A subtree $H \subseteq G$ is obtained by sampling uniformly at random a out of b of the children at each level of the tree, in level-order traversal. The set of *translated texts* is analogously defined as

$$T \coloneqq \{y \mid y = (y_1, \ldots, y_n) \text{ labels of a path in } H\} \subseteq P.$$

The prior $\rho = \mathcal{U}(P)$ is uniform over plausible texts P, while $\mu =$ is uniform over $f_\star^{-1}(T)$. We let $(\mu, \rho) \sim \mathrm{RT}(\mathcal{W}, n, a, b)$ denote the sampling of a prior and translated language obtained via this randomized process. When the parameters $\mathcal{W}, n, a, b$ are clear from context we simply write RT.

Note that Θ and $f_\star \in \Theta$ may be arbitrary, and by definition of τ, $\mu = \mathcal{U}(f_\star^{-1}(T))$ is uniform over T's preimage. Since we assumed $|\mathcal{X}| = |\mathcal{Y}|$ above, $f_\star$ is invertible.

Next, we will argue that a random tree language sampled in this process satisfies realizability (Condition 4.4) and Definition 4.5 with appropriate choice of parameters. While we make no assumptions on Θ, it is important that the plausible texts P are sampled *after* the possible permutations (ambiguities) Π are specified. Otherwise, if an adversary were able to choose a permutation $\pi\colon \mathcal{Y} \to \mathcal{Y}$ based on P, then they could arbitrarily permute P, resulting in a permutation π with high expected loss but no change in likelihood according to the prior ρ—thereby violating Definition 4.5.

You can think of Theorem 4.8 as pointing to the required text length n and an upper bound on the number of parameters $|\Theta|$ for which Condition 4.4 and Definition 4.5 hold with high probability. As we know from Theorem 4.6, and state clearly next, these in turn imply translatability of a random tree language.

Theorem 4.8 (Translatability in the RT model)**.** *Fix any $m \geq 1$, $\delta \in (0,1)$ vocabulary set $\mathcal{W}$, and tree arities $a \leq b \leq |\mathcal{W}|/4$. Then, for any Θ and any $\star \in \Theta$, with probability at least $1-\delta$ over $(\mu,\rho) \sim \mathrm{RT}(\mathcal{W},n,a,b)$ and iid samples $x_1, x_2, \ldots, x_m \sim \mu$,*

$$\mathrm{err}(\hat{\theta}) \leq 16 \max\left(\frac{1}{m}, \frac{4}{a^n}\right) \cdot \ln \frac{6|\Theta|}{\delta},$$

where $\hat{\theta} = MLE^{\rho}(x_1, x_2, \ldots, x_m)$ and MLE is from Definition 4.1

Note that the probability in the corollary is over both $(\mu,\rho) \sim$ RT and the m iid training samples. Again, note how the first term is similar to the $\frac{1}{m}\log\frac{|\Theta|}{\delta}$ term from realizable supervised learning (see Theorem 4.7), and the second term is additional due to the unsupervised case. The proof, given in Section 4.3, uses Theorem 4.6 and a lemma stating that $\varepsilon_\gamma \leq 16\gamma$ for γ that are not too small. The main challenge is that there are dependencies between the paths, so one cannot directly use standard concentration inequalities.

To better understand Theorem 4.8, let us consider two possible families of translators and suppose we are trying to apply Theorem 4.8 to get bounds on the sample complexity m needed to translate $(\mu,\rho) \sim \mathrm{RT}(\mathcal{W},n,a,b)$ with small constant loss and small constant failure probability.

First, since $|\mathcal{X}| = |\mathcal{Y}| = |\mathcal{W}|^n$, the above bound is meaningless (bigger than 1) if Θ is the family of all translators, as the set of all translators has $\log(|\mathcal{X}|!) = O(|\mathcal{W}|^n \cdot n \log|\mathcal{W}|)$ parameters which is much larger than a^n. In other words, *no free lunch.*

On the other hand, let us consider the family of *word-for-word* translators Θ_{w}, which work by translating each word in a text separately, ignoring the surrounding context. The number of such translators is the same as the number of word-to-word permutations, i.e., $|\Theta_{\mathrm{w}}| = |\mathcal{W}|!$. So Theorem 4.8 gives a sample complexity bound of $m = O(\log \mathcal{W}!) = O(|\mathcal{W}|\log|\mathcal{W}|)$. The number of words in the English language is about $N \approx 10^5$. The quantity a is equal to what is called the perplexity of μ, the effective number of words that may typically appear after a random prefix. For a constant a, the minimum text (or communication) length $n = O(\log N)$ needed is thus logarithmic in the vocabulary size. While word-for-word translators are poor, this analysis still sheds some light on the possible asymptotic behavior (barring computational constraints) of translation, at least in a simplistic model.

Common tree analysis. The tree model helps us demonstrate the generality of the common nonsense model. Instead of the fixed arities $a \le b$ in the tree model, one can consider an *arbitrary* language tree over plausible texts P and an *arbitrary* subset $T \subseteq P$ corresponding to a subtree of ground-truth translations. The common nonsense model implies that if the sets are perturbed by removing, say, a $\alpha = 0.01$ common fraction of translations, then with high probability MLE's error decreases at an $\tilde{O}(1/\min(m, |T|))$ rate. Note that Theorem 4.13 does not directly apply to the random tree LM because in that LM, the choice of which branches to include are not independent due to the fixed a-arity constraint.

4.3.1 Proof of Theorem 4.8

The proof of Theorem 4.8 follows from Lemma 4.9 below. In this section, we will prove this lemma and then derive the theorem from it.

Lemma 4.9 (RT conditions)**.** *Consider any vocabulary set $\mathcal{W}$, tree arities $a \le b \le |\mathcal{W}|/4$, tree depth $n \in \mathbb{N}$. Then, for any $\delta \in (0,1)$ and $\gamma \ge 4a^{-n}\ln(4|\Theta|/\delta)$, with probability $\ge 1-\delta$, (μ, ρ) give,*

$$\Pr_{(\mu,\rho)\sim \mathrm{RT}}[\varepsilon_\gamma \le 16\gamma] \ge 1 - \delta. \tag{4.1}$$

Moreover, any μ, ρ sampled from $\mathrm{RT}(\mathcal{W}, n, a, b)$ satisfy $\Pr_{x\sim\mu}[\rho(f_\star(x)) = 0] = 0$ as needed for the realizability Condition 4.4.

The proof of Lemma 4.9 requires two additional lemmas, stated and proved next. The first is a known variant of the Chernoff bound for so-called 0-negatively correlated random variables.

Lemma 4.10 (Chernoff bound for 0-negatively correlated random variables)**.** *Let $Z_1, \ldots, Z_n$ be* 0-negatively correlated *Boolean random variables, that is, they satisfy*

$$\forall I \subseteq [n] \quad \Pr[\forall i \in I \ Z_i = 0] \le \prod_{i\in I} \Pr[Z_i = 0].$$

Then, letting $Z \coloneqq \sum_{i=1}^n Z_i$ it holds that

$$\Pr\left[Z \le \frac{\mathbb{E}[Z]}{2}\right] \le e^{-\frac{\mathbb{E}[Z]}{8}}$$

Lemma 4.10 follows from (Doerr, 2019, Theorem 1.10.24(a)) with $\delta \coloneqq 1/2$, $a_i \coloneqq 0$, $b_i \coloneqq 1$ and $Y_i \coloneqq X_i$. That theorem, in turn, is simply Theorem 1.10.10 for 0-negatively correlated random variables.

The second lemma used for the proof of Lemma 4.9 is a combinatorial argument that we prove below.

Lemma 4.11. *Given any $\pi : \mathcal{Y} \hookrightarrow \mathcal{Y}$, it is possible to partition $A = \{y \in \mathcal{Y} \mid \pi(y) \neq y\}$ into four disjoint sets $A = A_1 \cup A_2 \cup A_3 \cup A_4$ such that, for each $i = 1, 2, 3, 4$, the following two conditions hold:*

$$\pi(A_i) \cap A_i = \emptyset$$
$$\forall z \in \mathcal{W}^{n-1} \ |\{y \in A_i \mid y \text{ begins with } z\}| \leq \frac{|\mathcal{W}|}{2}$$

Proof. of Lemma 4.11. We will partition A greedily to achieve the two conditions. Begin with four empty sets $A_1 = A_2 = A_3 = A_4 = \emptyset$. For each $y \in A$ in turn, assign it to one of the 4 sets as follows.

1. Let i be the index of the set A_i such that $\pi(y) \in A_i$ has already been assigned to. If $\pi(y))$ has not yet been assigned, let $i = 1$.
2. Similarly, if $\pi(y) \in A_j$ has been assigned, let j be its index otherwise $j = 1$.
3. Let z be the first $n - 1$ elements of x. Let k be the index of the set A_k such that $|\{y \in A_i \mid y \text{ begins with } z\}| \geq |\mathcal{W}|/2$ if there is such a set, otherwise $k = 1$. Note that there can be at most one such k because there are at most $|\mathcal{W}| - 1$ other elements beginning with z that have been assigned already.

Thus $S = \{1, 2, 3, 4\} \setminus \{i, j, k\} \neq \emptyset$ and we can assign x to any (say the minimum) element in that set. By induction, we preserve the two properties stated in the lemma. □

With Lemmas 4.10 and 4.11 in hand, we are ready to prove Lemma 4.9.

Proof. of Lemma 4.9. The realizability Condition 4.4, $\Pr_{x\sim\mu}[\rho(f_\star(x)) = 0] = 0$, follows immediately from the fact that ρ is uniform over P and that τ is supported on $T \subseteq P$. To prove the lemma, it suffices to show that for any $\gamma \geq 4a^{-n} \ln(4|\Theta|/\delta)$,

$$\Pr_{(\mu,\rho)\sim\mathrm{RT}}\left[\exists\theta \in \Theta \quad \mathrm{err}(\theta) > 16\gamma, \ \Pr_{y\sim\tau}[\rho(\pi_\theta(y)) = 0] \leq \gamma\right] \leq \delta.$$

Note that $\mathrm{err}(\theta) = \Pr_{y\sim\tau}[y \neq \pi_\theta(y)]$, and that $\rho(\pi_\theta(y)) = 0$ if and only if $\pi_\theta(y) \notin P$. Therefore, by the union bound over $\theta \in \Theta$, it suffices to show that, for each $\theta \in \Theta$,

$$\Pr_{(\mu,\rho)\sim\mathrm{RT}}\left[\Pr_{y\sim\tau}[y \neq \pi_\theta(y)] > 16\gamma, \ \Pr_{y\sim\tau}[\pi_\theta(y) \notin P] \leq \gamma\right] \leq \frac{\delta}{|\Theta|}. \tag{4.2}$$

We will show that, more generally, for any permutation $\pi : \mathcal{Y} \hookrightarrow \mathcal{Y}$. Equation (4.2) can be restated in terms of *ambiguous texts* $A \coloneqq \{y \in \mathcal{Y} \mid \pi(y) \neq y\}$ and *implausible ambiguities* $B \coloneqq \{y \in \mathcal{Y} \mid \pi(y) \notin P\}$:

$$\Pr_{(\mu,\rho)\sim\mathrm{RT}}[\tau(A) > 16\gamma, \ \tau(B) \leq \gamma] < \frac{\delta}{|\Theta|}.$$

Using Lemma 4.11, partition A into four disjoint sets $A = A_1 \cup A_2 \cup A_3 \cup A_4$ such that for each $i \in [4]$, the following two conditions hold:

$$\pi(A_i) \cap A_i = \emptyset \tag{4.3}$$

$$\forall z \in \mathcal{W}^{n-1} \quad |\{y \in A_i \mid y \text{ begins with } z\}| \leq \frac{|\mathcal{W}|}{2}. \tag{4.4}$$

By the union bound, it suffices to show that,

$$\forall i \in [4] \quad \Pr_{(\mu,\rho)\sim \mathrm{RT}}[\tau(A_i) > 4\gamma,\ \tau(B) \leq \gamma] < \frac{\delta}{4|\Theta|},$$

because $\tau(A) = \sum_{i=1}^{4} \tau(A_i)$, so if $\tau(A) > 16\gamma$ then it must follow that $\tau(A_i) > 16\gamma/4$ for some i. It suffices to show that this holds conditioned on any value of $A_i \cap T$:

$$\forall V \subseteq A_i \quad \Pr_{(\mu,\rho)\sim \mathrm{RT}}[\tau(A_i) > 4\gamma, \tau(B) \leq \gamma \mid A_i \cap T = V] \leq \frac{\delta}{4|\Theta|}. \tag{4.5}$$

Thus, fix any $V \subseteq A_i$. We proceed by analyzing two cases, based on $|V|$ versus $|T| = a^n$.[10]

Case 1: $|V| \leq 4\gamma|T|$. Then Equation (4.5) holds with probability 0, because conditioning on $A_i \cap T = V$, we have

$$\tau(A_i) \coloneqq \frac{|A_i \cap T|}{|T|} = \frac{|V|}{|T|} \leq 4\gamma.$$

Case 2: $|V| > 4\gamma|T|$. For each $v \in V$ we define a Boolean random variable $Z_v = \mathbf{1}_{v \in B}$, i.e.,

$$Z_v = \begin{cases} 1 & v \in B \\ 0 & v \notin B. \end{cases}$$

We claim that the Z_v's satisfy the condition of Lemma 4.10, which follows inductively from the fact any subset of Z_v's being 0 only makes it *less* likely for another Z_v to be 0, because the way that the Z_v's are chosen is such that there are exactly b many 1's among the Z_v's (and other symmetric random variables which we have not named).

Therefore, if we define $Z \coloneqq \sum_{v \in V} Z_v$ and $\zeta \coloneqq \mathbb{E}[Z \mid A_i \cap T = V]$, Lemma 4.10 gives,

$$\Pr\left[Z \leq \frac{\zeta}{2} \;\middle|\; A_i \cap T = V\right] \leq e^{-\zeta/8}. \tag{4.6}$$

We conclude by bounding both sides of Equation (4.6) to obtain Equation (4.5).

[10]Note that while T is randomly sampled, $|T| = a^n$ always, and therefore the case-analysis is valid.

- For the right-hand side of Eq. (4.6), we claim that $\Pr[Z_v = 1 \mid A_i \cap T = V] \geq 1/2$ for each $v \in V$: Fix v and let

$$Q = \{y \in \mathcal{Y} \mid y \text{ and } \pi(v) \text{ agree on the first } n-1 \text{ words}\}.$$

So $|Q| = |\mathcal{W}|$ and $|Q \cap P| = b$. Equation (4.3) implies that $\pi(v) \notin A_i$, and Eq. (4.4) implies that $|Q \setminus A_i| \geq |\mathcal{W}|/2$. Thus, there are $\geq |\mathcal{W}|/2$ elements in $Q \setminus A_i$ that, by symmetry, are as likely to be in $Q \cap P$ as $\pi(v)$ is, and $Q \cap P$ contains exactly $b \leq |\mathcal{W}|/4$ elements, thus the conditional probability that $\pi(v)$ is in $Q \cap P$ (equivalently, $\pi(v) \in P$) is at most:

$$\frac{|\mathcal{W}|/4}{|\mathcal{W}|/2} \leq \frac{1}{2}.$$

Since $\pi(v) \notin P$ is equivalent to $Z_v = 1$, this means that $\Pr[Z_v = 1 \mid A_i \cap T = V] \geq 1/2$, and hence:

$$\zeta = |V| \cdot \mathbb{E}[Z_v \mid A_i \cap T = V] = |V| \cdot \Pr[Z_v = 1 \mid A_i \cap T = V] \geq \frac{|V|}{2}.$$

Because we are in the case that $|V| > 4\gamma|T|$, we have

$$\zeta \geq \frac{|V|}{2} > \frac{1}{2} \cdot 4\gamma \cdot |T| = 2\gamma \cdot a^n. \tag{4.7}$$

Therefore the right-hand side of Eq. (4.6) satisfies

$$e^{-\zeta/8} \leq e^{-\gamma a^n/4} \geq \frac{\delta}{4\,|\Theta|},$$

where the rightmost inequality holds because by our assumption that $\gamma \geq 4a^{-n} \ln(4|\Theta|/\delta)$.

- For the left-hand side of Eq. (4.6), note that

$$Z \coloneqq \sum_{v \in V} Z_v \coloneqq \sum_{v \in V} \mathbf{1}_{v \in B} = |V \cap B|.$$

and therefore, using Eq. (4.7), the left-hand side of Eq. (4.6) satisfies

$$\Pr\left[Z < \frac{\zeta}{2} \;\middle|\; A_i \cap T = V\right] \geq \Pr\left[\frac{|V \cap B|}{|T|} \leq \gamma \;\middle|\; A_i \cap T = V\right].$$

We claim that $V \cap B \subseteq B \cap T$: Due to the conditional event, we have $V \cap B = A_i \cap T \cap B \subseteq A \cap T \cap B$. However, we argue that $A \cap T \cap B \subseteq T \cap B$, i.e., that $T \cap B \subseteq A$. Indeed, by definition, if $y \in T \cap B$ then $y \in T$ but $\pi(y) \notin P$, and since $P \supseteq T$, this implies that $\pi(y) \neq y$; that is, that $y \in A$, as needed.

We have just shown that $V \cap B \subseteq B \cap T$, and therefore

$$\Pr\left[\frac{|V \cap B|}{|T|} \leq \gamma \;\middle|\; A_i \cap T = V\right] \geq \Pr\left[\frac{|B \cap T|}{|T|} \leq \gamma \;\middle|\; A_i \cap T = V\right] = \Pr\left[\tau(B) \leq \gamma \mid A_i \cap T = V\right].$$

This shows that the left-hand side of Eq. (4.6) upper bounds that of Eq. (4.5), and concludes the proof.

□

Lastly, we prove Theorem 4.8 using Lemma 4.9

Proof. of Theorem 4.8. Let γ be,

$$\gamma := \max\left(\frac{1}{m}, \frac{4}{a^n}\right) \ln\frac{6|\Theta|}{\delta}.$$

Lemma 4.9 below shows that with probability $\geq 1 - 2\delta/3$ over μ, ρ, we have that $\varepsilon_\gamma \leq 16\gamma$. Theorem 4.6 implies that with probability $\geq 1 - \delta/3$ over $x_1, \ldots, x_m$, we have $\mathrm{err}(\hat{\theta}) \leq \varepsilon_\gamma$. Thus, by the union bound, with probability $\geq 1 - \delta$ we have

$$\mathrm{err}(\hat{\theta}) \leq \varepsilon_\gamma \leq 16\gamma = 16 \max\left(\frac{1}{m}, \frac{4}{a^n}\right) \ln\frac{6|\Theta|}{\delta}.$$

□

4.4 The common nonsense model

We next perform a "smoothed analysis" of arbitrary Language Models (LMs) μ, ρ that are uniform over sets that share a small amount of randomness, i.e., a small common random set has been removed from both. This shared randomness captures the fact that some texts are implausible in both languages and that this set has some complex structure determined by the laws of nature, which we model as random.

The α-common-nonsense distribution is a meta-distribution over pairs (μ, ρ) which themselves are uniform distributions over perturbed versions of P, T. This is inspired by Smoothed Analysis (Spielman and Teng, 2009). Recall that $\mathcal{U}(S)$ denotes the uniform distribution over the set S.

Definition 4.12 (Common nonsense)**.** *The α-common-nonsense distribution $\mathcal{D}_\alpha^{P,T}$ with respect to nonempty sets $T \subseteq P \subseteq \mathcal{Y}$ is the distribution over $\big(\rho = \mathcal{U}(P \cap S), \tau = \mathcal{U}(T \cap S)\big)$ where $S \subseteq \mathcal{Y}$ is formed by removing each $y \in \mathcal{Y}$ with probability α, independently.*[11]

[11] Again, in the exponentially unlikely event that either $P \cap S$ or $T \cap S$ is empty, we define both τ, ρ to be the singleton distribution concentrated on the lexicographically smallest element of $\mathcal{Y}$, so MLE outputs a 0-error translator.

To make this concrete in terms of a distribution μ on $\mathcal{X}$, for any ground-truth translator $f_\star : \mathcal{X} \hookrightarrow \mathcal{Y}$, we similarly define a distribution $\mathcal{D}^{P,T}_{\alpha,\star}$ over (μ, ρ) where $\mu \coloneqq \mathcal{U}(f_\star^{-1}(T \cap S))$ is the uniform distribution over the subset of $\mathcal{X}$ that translates into τ. We now state the formal version of Theorem 4.3.

Theorem 4.13 (Translatability in the CN model)**.** *Let $\{f_\theta : \mathcal{X} \hookrightarrow \mathcal{Y} \mid \theta \in \Theta\}$ a family of translators, $\star \in \Theta$, $\alpha, \delta \in (0, 1/2]$, $T \subseteq P \subseteq \mathcal{Y}$, and $m \geq 1$. Then with probability $\geq 1 - \delta$, MLE run on ρ and $m \geq 1$ iid samples from μ outputs $\hat{\theta}$ with,*

$$\operatorname{err}(\hat{\theta}) \leq \frac{6}{\alpha} \max\left(\frac{1}{m}, \frac{16}{|T|}\right) \cdot \ln \frac{6|\Theta|}{\delta}.$$

Note that the probability is over both (μ, ρ) drawn from $\mathcal{D}^{P,T}_{\alpha,\star}$, and the m iid samples from μ. More simply, with probability ≥ 0.99,

$$\operatorname{err}(\hat{\theta}) = O\left(\frac{\log|\Theta|}{\alpha \min(m, |T|)}\right).$$

When the amount of shared randomness α is a constant, then this decreases asymptotically like the bound of supervised translation (Theorem 4.7) up until a constant, similar to Theorem 4.18. For very large m, each extra bit describing the translator (increase by 1 in $\log|\Theta|$) amounts to a constant number of mistranslated x's out of all $\mathcal{X}$. The proof is deferred to Section 4.4.

We also prove the following lower-bound that is off by a constant factor of the upper bound.

Theorem 4.14 (CN lower-bound)**.** *There exists constants $c_1, c_2 \geq 1$ such that: for any set $T \subseteq \mathcal{Y}$, for any $m \geq 1$, any $\alpha \in (0, 1/2]$, and any Θ with $c_1 \leq \log|\Theta| \leq \alpha \min(m, |T|)$, there exists Θ' of size $|\Theta'| \leq |\Theta|$ such that, for any $P \supseteq T$ and any algorithm $A^\rho : \mathcal{X}^m \to \Theta'$, with probability ≥ 0.99 over $\star \sim \mathcal{U}(\Theta')$ and (μ, ρ) drawn from $\mathcal{D}^{P,T}_{\alpha,\star}$ and $x_1, \ldots, x_m \sim \mu$,*

$$\operatorname{err}\big(\hat{\theta}\big) \geq \frac{\log|\Theta|}{c_2 \alpha \min(m, |T|)},$$

where $\hat{\theta} = A^\rho(x_1, x_2, \ldots, x_m)$.

The only purpose of Θ in the above theorem is to upper-bound the description length of translators, as we replace it with an entirely different (possibly *smaller*) translator family Θ' that still has the lower bound using $\log|\Theta| \geq \log|\Theta'|$. Since $\mathcal{U}(\Theta')$ is the uniform distribution over Θ', the ground-truth classifier is uniformly random from Θ'. A requirement of the form $\log|\Theta| = O\big(\alpha \min(m, |T|)\big)$ is inherent as otherwise one would have an impossible right-hand side error lower-bound greater than 1, though the constants could be improved.

The proof of this theorem is given in Section 4.4.2, and creates a model with $O(\log n)$ independent "ambiguities" that cannot be resolved, with high probability over $S, x_1, x_2, \ldots, x_m$.

4.4.1 Proving Theorem 4.13

First, to convey intuition, we think of the case of say $\delta = 99\%$, and we omit constants in the following discussion. Recall that Theorem 4.6 asserts that with high probability, the learned translator $\hat{\theta}$ has error at most ε_γ for $\gamma = \log|\Theta|/\min(m, |T|) \geq \log|\Theta|/m$. As such, the main technical challenge in this proof is to show that, with high probability over the generated source language μ and the prior ρ, it holds that

$$\varepsilon_\gamma \lesssim \frac{\gamma}{\alpha}.$$

By definition of ε_γ, we ought to show that w.h.p over μ and ρ, any $\theta \in \Theta$ with large semantic error must have many translations $f_\theta(x)$ deemed implausible by ρ. Slightly more formally: Since any $y \in \mathcal{Y}$ is implausible ($\rho(y) = 0$) only if $y \notin S$, a union bound over $\theta \in \Theta$ means that is suffices to show that

$$\Pr_{S,\mu,\rho}\left[\Pr_{x\sim\mu}[f_\theta(x) \neq f_\star(x)] \lesssim \frac{\gamma}{\alpha},\ \Pr_{x\sim\mu}[f_\theta(x) \notin S] \gtrsim \gamma\right] \lesssim \exp(-\gamma|T|)) \leq \frac{1}{|\Theta|},$$

where the right inequality is by choice of $\gamma \coloneqq \log|\Theta|/\min(m, |T|)$. The above inequality "looks like" it could be proven by a Chernoff bound, but a closer look reveals a subtle flaw with this argument.

To use a Chernoff bound, we'd first want to fix (i.e., condition on) each $\mathrm{supp}(\mu) \coloneqq f_\star^{-1}(S \cap T)$ and then use Chernoff over the conditional random variables $\mathbf{1}_{f_\theta(x)\notin S}$ for each $x \in \mathrm{supp}(\mu)$ such that $f_\theta(x) \neq f_\star(x)$. Unfortunately, these conditional random variables are *not* independent. To see this, consider the case that $f_\theta(x) = f_\star(x')$ for two different $x \neq x'$. Then, since we are considering $x' \in \mathrm{supp}(\mu)$, we have $f_\theta(x) = f_\star(x') \in S$ with probability 1.

To avoid this dependency, we prove a combinatorial lemma showing that it is possible to partition the set

$$A \coloneqq \{x \in X \mid f_\theta(x) \neq f_\star(x)\}$$

into three parts $A = A_1 \cup A_2 \cup A_3$ such that $f_\theta(A_i) \cap f_\star(A_i) = \emptyset$ for each $i \in [3]$. This resolves the dependency issue demonstrated above. We then proceed by applying a Chernoff bound separately for each A_i, which suffices since a union bound (over $i \in [3]$) loses only a constant factor in the upper-bound.

The full proof of Theorem 4.13 follows from the following main lemma. We first prove this lemma, and then show how the theorem follows from it.

Lemma 4.15. *Let $\alpha, \delta \in (0,1)$ and $T \subseteq P \subseteq \mathcal{Y}$. Then, for any $\gamma \geq \frac{8}{(1-\alpha)\cdot|T|}\ln\frac{4|\Theta|}{\delta}$:*

$$\Pr_{(\mu,\rho)\sim\mathcal{D}_\alpha^{P,T}}\left[\varepsilon_\gamma \leq \frac{6\gamma}{\alpha}\right] \geq 1 - \delta.$$

The proof of Lemma 4.15 relies on a simple combinatorial proposition. This proposition is a special case of Lemma 4.11, but since it is much simpler we give a self-contained proof.

Proposition 4.16. *For any finite set $\mathcal{Y}$ and any $\pi : \mathcal{Y} \hookrightarrow \mathcal{Y}$, it is possible to partition $\{y \in \mathcal{Y} | \pi(y) \neq y\} = A_1 \cup A_2 \cup A_2$ into three sets such that, $\pi(A_i) \cap A_i = \emptyset$ for $i = 1, 2, 3$.*

Proof. Let $S \coloneqq \{y \in \mathcal{Y} \mid \pi(y) \neq y\}$. We proceed iteratively, dividing each (non-trivial) cycle of π separately into the three A_i's: Fix a cycle $\{s_1, s_2, \ldots, s_n\}$ such that $\pi(s_i) = s_{i+1}$ and $\pi(s_n) = s_1$. If n is even we can just partition it into two sets: put the s_i for even i's into A_1 and odd i's into A_2. If n is odd, we can do the same except put the last element s_n into A_3. □

We can now prove Lemma 4.15.

Proof. of Lemma 4.15. Note that the lemma holds trivially for any $\gamma > 1/6$ because we always have $\varepsilon_\gamma \leq 1$. Assume that $\gamma \leq 1/6$. Let $a \coloneqq \frac{6}{\alpha}\gamma$. The probabilities in this proof are over the choice of $S \subseteq \mathcal{Y}$. It suffices to show that,

$$\Pr_S \left[\exists \theta \in \Theta \;\; \Pr_\tau[\pi_\theta(y) \neq y] > a \;\wedge\; \Pr_\tau[\pi_\theta(y) \notin S] \leq \gamma \right] \leq \delta, \tag{4.8}$$

because $\rho(\pi_\theta(y)) = 0$ whenever $\pi_\theta(y) \notin S$, thus $\mathcal{A}_\gamma = \{\theta \in \Theta \mid \Pr_\tau[\pi_\theta(y) \notin S] \leq \gamma\}$. The set S determines the perturbed sets $\tilde{T} \coloneqq T \cap S$ and $\tilde{P} \coloneqq P \cap S$ and the distributions $\tau = \mathcal{U}(\tilde{T})$ and $\rho = \mathcal{U}(\tilde{P})$.

Define $\beta \coloneqq 1 - \alpha$. Since $\mathbb{E}\big[|\tilde{T}|\big] = \beta|T|$, a multiplicative Chernoff bound[12] gives

$$\Pr_S \left[|\tilde{T}| \leq \frac{\beta}{2}|T| \right] \leq \exp\left(-\frac{\beta}{8}|T| \right) \leq \exp\left(-\frac{\beta}{8} \cdot \frac{8}{\beta\gamma} \ln \frac{4|\Theta|}{\delta} \right) < \frac{\delta}{4|\Theta|} \leq \frac{\delta}{4}.$$

Thus, to show Eq. (4.8), it suffices to show that

$$\Pr \left[|\tilde{T}| \geq \frac{\beta}{2}|T| \;\wedge\; \exists \theta \in \Theta \;\; \Pr_\tau[\pi_\theta(y) \neq y] > a \;\wedge\; \Pr_\tau[\pi_\theta(y) \notin S] \leq \gamma \right] \leq \frac{3\delta}{4}. \tag{4.9}$$

By the union bound, to show Eq. (4.9) it thus suffices to show that for any $\pi : \mathcal{Y} \hookrightarrow \mathcal{Y}$,

$$\Pr \left[|\tilde{T}| \geq \frac{\beta}{2}|T| \;\wedge\; \Pr_\tau[\pi(y) \neq y] > a \;\wedge\; \Pr_\tau[\pi(y) \notin S] \leq \gamma \right] \leq \frac{3\delta}{4|\Theta|}. \tag{4.10}$$

Fix any $\pi : \mathcal{Y} \hookrightarrow \mathcal{Y}$. By Proposition 4.16, we can partition $\{y \in \mathcal{Y} \mid \pi(y) \neq y\} = A_1 \cup A_2 \cup A_3$ such that $\pi(A_i) \cap A_i = \emptyset$, and hence $\Pr_\tau[\pi(y) \neq y] = \sum_i \tau(A_i)$. So if $\Pr_\tau[\pi(y) \neq y] > a$, then $\tau(A_i) > a/3$ for some i. Therefore, it suffices to show that

$$\Pr \left[|\tilde{T}| \geq \frac{\beta}{2}|T| \;\wedge\; \exists i \in [3] \; \tau(A_i) > \frac{a}{3} \;\wedge\; \Pr_\tau[\pi(y) \notin S] \leq \gamma \right] \leq \frac{3\delta}{4|\Theta|}.$$

[12] Specifically, that the probability that a sum of binary random variables is less than half its mean $\beta|T|$ is at most $\exp(-\beta|T|/8)$.

With a union bound over $i \in [3]$, it suffices to show that for each $i \in [3]$

$$\Pr\left[|\tilde{T}| \geq \frac{\beta}{2}|T| \ \wedge \ \tau(A_i) > \frac{a}{3} \ \wedge \ \Pr_{\tau}[\pi(y) \notin S] \leq \gamma\right] \leq \frac{\delta}{4|\Theta|}. \tag{4.11}$$

Thus, now in addition to fixing π, we fix $i \leq 3$, thus fixing A_i. To continue, imagine we are picking S by first selecting $A_i \cap S$, and subsequently selecting $S \setminus A_i$. We will show that Equation (4.11) holds when conditioning on each possible value for the first selection, that is, each possible $A_i \cap S$. Formally, we fix $V \subseteq A_i$ and condition on $A_i \cap S = V$, claiming that

$$\Pr\left[|\tilde{T}| \geq \frac{\beta}{2}|T| \ \wedge \ \tau(A_i) > \frac{a}{3} \ \wedge \ \Pr_{\tau}[\pi(y) \notin S] \leq \gamma \middle| V = A_i \cap S\right] \leq \frac{\delta}{4|\Theta|}. \tag{4.12}$$

First, observe that

$$\tau(A_i) = \frac{|A_i \cap \tilde{T}|}{|\tilde{T}|} = \frac{|A_i \cap S|}{|\tilde{T}|} = \frac{|V|}{|\tilde{T}|},$$

therefore if $|V| < \beta a|T|/6$ then Equation (4.12) holds with probability 0 (due to the first two events in the conjunction). Thus, we can assume that $|V| \geq \beta a|T|/6$.

Note that $\tau(V) = \tau(A_i)$, and that $\tau(A_i) \geq a/3$ implies

$$\Pr_{y\sim\tau}[\pi(y) \notin S] \geq \Pr_{y\in V}[\pi(y) \notin S] \cdot \tau(V) \geq \Pr_{y\in V}[\pi(y) \notin S] \cdot \frac{a}{3},$$

therefore the left-hand side of Equation (4.12) is upper-bounded by

$$\Pr\left[\Pr_{y\in V}[\pi(y) \notin S] \leq \frac{3\cdot\gamma}{a} \ \middle| \ V = A_i \cap S\right] = \Pr\left[\Pr_{y\in V}[\pi(y) \notin S] \leq \frac{a}{2} \ \middle| \ V = A_i \cap S\right]. \tag{4.13}$$

We conclude the proof by upper-bounding Equation (4.13) with a Chernoff bound. Consider the random variables $Z_y = \mathbf{1}_{\pi(y)\notin S}$ for $y \in V$. These random variables are independent by definition of α-common-nonsense. Furthermore, due to the fact that $\pi(A_i) \cap A_i = \emptyset$, they remain independent even when conditioning on the event $A_i \cap S = V$. By linearity of expectation,

$$\mathbb{E}\left[\sum_{y\in V} Z_y \ \middle| \ V = A_i \cap S\right] = \alpha|V|.$$

Using the same Chernoff bound as above, we have

$$\Pr\left[\frac{\sum_{y\in V} Z_y}{|V|} \leq \frac{\alpha}{2} \ \middle| \ V = A_i \cap S\right] \leq \exp(-\alpha|V|/8)$$

Noting that $\sum Z_y/|V| = \Pr_{y\in V}[\pi(y) \notin S]$, we conclude that that Equation (4.13) is upper-bounded by

$$\exp\left(-\frac{\alpha|V|}{8}\right) \leq \exp\left(-\frac{\alpha\beta a|T|}{48}\right) \leq \frac{\delta}{4|\Theta|}.$$

This proves the inequality in Equation (4.12), thereby concluding the proof.

□

Finally, with the main technical lemma in hand, we prove Theorem 4.13.

Proof. of Theorem 4.13. The proof is very similar to that in Section 4.3. First, the realizability Condition 4.4, $\Pr_{x\sim\mu}[\rho(f_\star(x)) = 0] = 0$, follows immediately from the fact that μ, ρ are uniform distributions with $f_\star(\text{supp}(\mu)) \subseteq \text{supp}(\rho)$ which follows from the fact that $T \subseteq P$ and the definitions of μ, ρ.

Let $\beta \coloneqq 1 - \alpha$ and define γ by,

$$\gamma \coloneqq \max\left(\frac{1}{m}, \frac{8}{\beta|T|}\right) \ln\frac{6|\Theta|}{\delta}.$$

Lemma 4.15 below shows that with probability $\geq 1-2\delta/3$ over μ, ρ, we have that $\varepsilon_\gamma \leq 6\gamma/\alpha$. Theorem 4.6 implies that with probability $\geq 1 - \delta/3$ over $x_1, \ldots, x_m$, we have $\text{err}(\hat{\theta}) \leq \varepsilon_\gamma$. Thus, by the union bound, with probability $\geq 1 - \delta$ we have

$$\text{err}(\hat{\theta}) \leq \varepsilon_\gamma \leq \frac{6\gamma}{\alpha} = \frac{6}{\alpha}\max\left(\frac{1}{m}, \frac{8}{\beta|T|}\right) \ln\frac{6|\Theta|}{\delta}.$$

□

4.4.2 Proving Theorem 4.14

The proof of the lower bound works by creating $\log n$ candidate "plausible ambiguities" and arguing that a constant fraction of them survive the random removal of elements.

Proof of Theorem 4.14. The constants c_1, c_2 will be determined through this proof to be large enough to satisfy multiple conditions defined below. No effort has been made to minimize the constants in this proof.

Let $n \coloneqq |\Theta|$.

We will lay out two $a \times b$ grids $X \subseteq \mathcal{X}$ and $Y \subseteq \mathcal{Y}$, for:

$$a \coloneqq \lfloor \log n \rfloor, b \coloneqq \left\lfloor \frac{1}{\alpha}\max\left(1, \frac{|T|}{10^5 m}\right)\right\rfloor.$$

For integer t, denote $[t] \coloneqq \{1, 2, \ldots, t\}$. For $i \in [a], j \in [b]$, choose distinct elements $x_{ij} \in \mathcal{X}$ and $y_{ij} \in T$. To ensure this is even possible, we must make sure $ab \leq |T|$, which holds because we assumed $\log n \leq \alpha \min(m, |T|)$ thus,

$$ab \leq \alpha \min(m, |T|)\frac{1}{\alpha}\max\left(1, \frac{|T|}{m}\right) = \min(m, |T|) \cdot \max\left(\frac{1}{|T|}, \frac{1}{m}\right)|T| = |T|.$$

Let $X \coloneqq \{x_{ij} \mid i \in [a], j \in [b]\}$ and $Y \coloneqq \{y_{ij} \mid i \in [a], j \in [b]\}$. Let $h : \mathcal{X} \setminus X \hookrightarrow \mathcal{Y} \setminus Y$ be a fixed 1–1 mapping, say the lexicographically smallest. The parametrized translator family is defined by,

$$\Theta' = \Theta_{m,n,Y} \coloneqq \{-1, 1\}^a, \quad f_\theta(x_{ij}) \coloneqq \begin{cases} y_{ij}, & \theta_i = 1 \\ y_{i(j+1 \bmod b)}, & \theta_i = -1 \end{cases}, \quad \forall x \notin X \; f_\theta(x) = h(x).$$

Clearly $|\Theta'| = 2^a \leq n = |\Theta|$ as needed. Let $\mathbf{x} = (x_1, x_2, \ldots, x_m)$. It suffices to show:

$$\Pr_{S,\mathbf{x}}\left[\mathrm{err}\big(\hat{\theta}\big) \geq \frac{\log n}{c_3 \alpha \min(10^5 m, |T|)}\right] \geq 0.99,$$

for some constant c_3 sufficiently large, because we can set $c_2 = 10^5 c_3$. The above equation is equivalent to the following two cases based on m:

$$\text{Case 1. } 10^5 m > |T| \implies \Pr_{S,\mathbf{x}}\left[\mathrm{err}\big(\hat{\theta}\big) \geq \frac{\log n}{c_3 \alpha |T|}\right] \geq 0.99 \tag{4.14}$$

$$\text{Case 2. } 10^5 m \leq |T| \implies \Pr_{S,\mathbf{x}}\left[\mathrm{err}\big(\hat{\theta}\big) \geq \frac{\log n}{c_3 \alpha 10^5 m}\right] \geq 0.99 \tag{4.15}$$

In both cases, it will be convenient to notice that, for any $z \geq 2$, $\lfloor z \rfloor \geq z - 1 \geq \frac{z}{2}$. Since $b \geq 2$ (because $\alpha \leq 1/2$), we therefore have

$$\frac{1}{2\alpha} \max\left(1, \frac{|T|}{10^5 m}\right) \leq b \leq \frac{1}{\alpha} \max\left(1, \frac{|T|}{10^5 m}\right) \tag{4.16}$$

Case 1: $10^5 m > T$. In this case $\frac{1}{2\alpha} \leq b \leq \frac{1}{\alpha}$ by Equation (4.16). We will show Equation (4.14). Now, consider the "full rows":

$$C(S) \coloneqq \{i \in [a] \mid \forall j \in [b]\ y_{ij} \in S\}.$$

These rows will be useful to consider because nothing has been removed from the entire row, no information about θ_i has been revealed and (on average) one cannot achieve error $< 1/2$ on these examples, because one cannot distinguish between the two permutations on this row.

Note that the membership of different $i, i' \in C(S)$ is independent since S is chosen independently, and by definition of C and S:

$$\mathbb{E}[|C(S)|] = (1-\alpha)^b a \geq (1-\alpha)^{1/\alpha} a \geq \frac{a}{4},$$

since $(1-\alpha)^{1/\alpha}$ is decreasing in α and $\alpha \leq 1/2$. Thus, by multiplicative Chernoff bounds (specifically, $\Pr[Z \leq \mathbb{E}[Z]/2] \leq e^{-\mathbb{E}[Z]/8}$),

$$\Pr_S\left[|C(S)| \leq \frac{a}{8}\right] \leq e^{-a/32} \leq e^{-\lfloor c_1 \rfloor/32} \leq 0.001, \tag{4.17}$$

for sufficiently large c_1. Thus, $\Pr_S[|C(S)| > a/8] \geq 0.999$. Let $C'(S, \mathbf{x}) \subseteq C(S)$ be those i which $\hat{\theta}_i \neq \star_i$,

$$C'(S, \mathbf{x}) \coloneqq \{i \in C(S) \mid \hat{\theta}_i \neq \star_i\}.$$

Clearly, for any algorithm and any $C(S)$, $\mathbb{E}_{\mathbf{x}}[|C'(S,\mathbf{x})| \mid S] = |C(S)|/2$ because no information whatsoever has been revealed about θ_i for any $i \in C$. Thus, by the same Chernoff bound, we have:

$$\Pr_{S,\mathbf{x}}\left[|C'(S,\mathbf{x})| \le \frac{1}{4}|C(S)| \;\middle|\; |C(S)| > \frac{a}{8}\right] \le e^{-\frac{a}{16}\cdot\frac{1}{8}} \le 0.001,$$

for sufficiently large c_1, because $a \ge c_1$. By the union bound over this and Equation (4.17),

$$\Pr_{S,\mathbf{x}}\left[|C'(S,\mathbf{x})| \ge \frac{a}{32}\right] \ge 0.998.$$

Since each row $i \in C'$ incurs b errors on examples x, one for each j because $f_\star(x_{ij}) \in S$:

$$\operatorname{err}\big(\hat{\theta}\big) \ge \frac{b \cdot \big|C'(S,\mathbf{x})\big|}{|T|}.$$

Thus,

$$\Pr_{S,\mathbf{x}}\left[\operatorname{err}\big(\hat{\theta}\big) \ge \frac{ba}{32\alpha|T|}\right] \ge 0.998.$$

Now, $a \ge \frac{1}{2}\log n$ for sufficiently large c_1 and as mentioned $b \ge \frac{1}{2\alpha}$. Thus,

$$\Pr_{S,\mathbf{x}}\left[\operatorname{err}\big(\hat{\theta}\big) \ge \frac{\log n}{128\alpha|T|}\right] \ge 0.998.$$

This establishes Equation (4.14) as long as $c_3 \ge 128$.

It remains to prove Equation (4.15).

Case 2: $10^5 m \le T$. In this case $\frac{|T|}{2\alpha 10^5 m} \le b \le \frac{|T|}{\alpha 10^5 m}$ by Equation (4.16). Next, consider the set of rows with at least $1/2$ of the elements in S:

$$D(S) \coloneqq \left\{ i \in [a] \;\middle|\; \big|\{j \in [b] \mid y_{ij} \in S\}\big| \ge \frac{b}{2} \right\}.$$

Intuitively, any row $i \in D(S)$ is "dangerous" in the sense that if $\hat{\theta}_i \ne \star_i$, then it causes errors on $b/2$ different x's in the support of μ, i.e., for which $f_\star(x_{ij}) \in S$. Observe that $\mathbb{E}[|D(S)|] \ge a/2$ since each size $s = \big|\{j \in [b] \mid y_{ij} \in S\}\big| \ge b/2$ is at least as likely as the size $b - s$, since $\alpha \le 1/2$. And also, membership of $i, i' \in D(S)$ since S is independent. Thus, by the same Chernoff bound as above, for sufficiently large c_1,

$$\Pr_{S}\left[|D(S)| \le \frac{a}{4}\right] \le e^{-a/16} \le e^{-\lfloor c_1 \rfloor/16} \le 0.001. \tag{4.18}$$

Let $-\theta \coloneqq (-\theta_1, -\theta_2, \ldots, -\theta_a)$. This makes it convenient to define the *giveaways* $G(S) \subseteq X$ to be,

$$G(S) \coloneqq \{x_{ij} \mid i \in [a], j \in [b], f_\star(x_{ij}) \in S, f_{-\star}(x_{ij}) \notin S\}.$$

These are the points x_{ij} which we might observe $f_\star(x_{ij})$ which would imply that $\theta_i = \star_i$ (and not its negative). Also let,

$$\hat{G}(S, \mathbf{x}) = \{x_1, x_2, \ldots, x_m\} \cap G(S).$$

(Note that if for a give row i, we do not have any $x_{ij} \in \hat{G}(S, \mathbf{x})$, then we have no information about θ_i. As a preview to what is to come, we now argue that with high probability $|\hat{G}(S, \mathbf{x})| < a/8$ which will mean that, if $|D(S)| > a/4$, then we have no information about θ_i for at least $a/8$ of the rows $i \in D(S)$.)

For any fixed i, j, observe that $\Pr[x_{ij} \in G(S)] = \alpha(1-\alpha)$ so $\mathbb{E}[|G(S)|] = \alpha(1-\alpha)ab \le \alpha ab$. By the Chernoff bound that $\Pr[Z \ge 2\,\mathbb{E}[Z]] \le e^{-\mathbb{E}[Z]/3}$,

$$\Pr_S[|G(S)| \ge 2\alpha ab] \le e^{-\alpha ab/3} \le e^{-a/3} \le 0.001,$$

for sufficiently large c_1. (We have used the fact that the above probability is smaller than if $E[|G(S)|]$ were actually αab.)

Also, $\mathbb{E}_S[|T \cap S|] \ge |T|/2$ since $\alpha \le 1/2$. So, by the Chernoff bound $\Pr[Z \le \mathbb{E}[Z]/2] \le e^{-\mathbb{E}[Z]/8}$,

$$\Pr_S\left[|T \cap S| \le \frac{|T|}{4}\right] \le e^{-|T|/16} \le 0.001,$$

for sufficiently large c_1 since $|T| \ge \log n \ge c_1$.

Thus, by the union bound:

$$\Pr_S\left[|T \cap S| \le \frac{|T|}{4} \vee |G(S)| \ge 2\alpha ab\right] \le 0.002. \tag{4.19}$$

Also,

$$\mathop{\mathbb{E}}_{\mathbf{x}}\left[|\hat{G}(S, \mathbf{x})| \mid S\right] \le m\frac{|G(S)|}{|T \cap S|}.$$

Thus, using Markov's inequality in the second line below,

$$\mathop{\mathbb{E}}_{\mathbf{x},S}\left[|\hat{G}(S, \mathbf{x})| \;\middle|\; |T \cap S| > \frac{|T|}{4}, |G(S)| \le 2\alpha ab\right] \le m\frac{2\alpha ab}{|T|/4} = \frac{8\alpha abm}{|T|},$$

$$\Pr_{\mathbf{x},S}\left[|\hat{G}(S, \mathbf{x})| > \frac{8000\alpha abm}{|T|} \;\middle|\; |T \cap S| > \frac{|T|}{4}, |G(S)| \le 2\alpha ab\right] \le 0.001.$$

By the union bound over the above and Equation (4.19), since $\Pr[E] \le \Pr[E|F] + \Pr[\neg F]$

$$\Pr_{\mathbf{x},S}\left[|\hat{G}(S, \mathbf{x})| > \frac{8000\alpha abm}{|T|}\right] \le 0.001 + 0.002.$$

Finally, since

$$\begin{aligned} b &\leq \frac{|T|}{\alpha 10^5 m} \\ \frac{8000\alpha abm}{|T|} &\leq 0.08 \cdot a \leq \frac{a}{8} \\ \Pr_{\mathbf{x},S}\left[|\hat{G}(S,\mathbf{x})| > \frac{a}{8}\right] &\leq 0.003 \end{aligned}$$

By the union bound with Equation (4.18),

$$\Pr_{\mathbf{x},S}\left[|D(S)| \leq \frac{a}{4} \vee |\hat{G}(S,\mathbf{x})| > \frac{a}{8}\right] \leq 0.004$$

Let

$$F(S,\mathbf{x}) \coloneqq \{i \in D(S) \mid \forall j \in [b]\ x_{ij} \notin \hat{G}(S)(S,\mathbf{x})\} \subseteq D.$$

Clearly $|F(S,\mathbf{x})| \geq |D(S)| - |\hat{G}(S,\mathbf{x})|$ because each $x \in \hat{G}(S,\mathbf{x})$ can remove at most one row from $D(S)$. Thus,

$$\Pr_{\mathbf{x},S}\left[|F(S,\mathbf{x})| \leq \frac{a}{8}\right] \leq 0.004. \tag{4.20}$$

$F(S,\mathbf{x})$ will function exactly like C in the analysis above of Equation (4.14). We repeat this analysis for completeness, replacing C by F. Let $F'(S,\mathbf{x}) \subseteq F(S,\mathbf{x})$ be those i which $\hat{\theta}_i \neq \star_i$,

$$F'(S,\mathbf{x}) \coloneqq \{i \in F(S,\mathbf{x}) \mid \hat{\theta}_i \neq \star_i\}.$$

For any algorithm and any $F(S,\mathbf{x})$, $\mathbb{E}_{\mathbf{x}}[|F'(S,\mathbf{x})| \mid S] = |F(S,\mathbf{x})|/2$ because no information whatsoever has been revealed about θ_i for any $i \in F$. Thus, by the same Chernoff bound, we have:

$$\Pr_{S,\mathbf{x}}\left[|F'(S,\mathbf{x})| \leq \frac{1}{4}|F(S,\mathbf{x})| \;\middle|\; |F(S,\mathbf{x})| > \frac{a}{8}\right] \leq e^{-\frac{a}{16}\cdot\frac{1}{8}} \leq 0.001,$$

for sufficiently large c_1. By the union bound over this and Equation (4.20),

$$\Pr_{S,\mathbf{x}}\left[|F'(S,\mathbf{x})| \geq \frac{a}{32}\right] \geq 0.995.$$

Since each row $i \in F'$ incurs $\geq b/2$ errors on examples x by definition of F' and D, since $F' \subseteq D$ and thus at least $b/2$ errors on j for which $f_\star(x_{ij}) \in S$. Thus,

$$\operatorname{err}\left(\hat{\theta}\right) \geq \frac{b \cdot \left|F'(S,\mathbf{x})\right|}{2|T|}.$$

Thus,

$$\Pr_{S,\mathbf{x}}\left[\operatorname{err}\left(\hat{\theta}\right) \geq \frac{ba}{64\alpha|T|}\right] \geq 0.995.$$

Name	Symbol	Value
Number of source sentences	$\lvert T\rvert$	10^5
Number of target sentences	$\lvert P\rvert$	10^6
Number of training data	m	$1, \ldots, 100$
Number of validation data		1000
Fraction of common nonsense	α	$0, 0.1, \ldots, 0.8$

Figure 4.7: **Parameters for experiments in the common nonsense model (Figure 4.5).** The experiments were run in parallel on an AWS r6i.4xlarge for a total of four CPU-hours.

Now, $a \geq \frac{1}{2}\log n$ for sufficiently large c_1 and we also have $b \geq \frac{|T|}{2\alpha 10^5 m}$ by Equation (4.16) since $\frac{|T|}{\alpha 10^5 m} \geq 2$ since $\alpha \leq 1/2$ and we are in the case where $10^5 m \leq |T|$. Thus,

$$\Pr_{S,\mathbf{x}}\left[\mathrm{err}\big(\hat{\theta}\big) \geq \frac{\log n}{256\alpha 10^5 m}\right] \geq 0.995.$$

This establishes Equation (4.15) for $c_3 \geq 256 \times 10^5$. □

4.4.3 Experiments

Since in the common nonsense model the structure of sentences is arbitrary, we represent sentences by integer IDs, $[10^5] = 1, 2, \ldots, 10^5$ and $[10^6]$ for the target language. We generate a prior ρ from the common nonsense model by taking the target sentence ids $[10^6]$ and labeling a random α-fraction of them as nonsense; the remaining sentences are called sensical S. Given a ground-truth translator $f_\star \colon [10^5] \to [10^6]$, the source language then distributes uniformly over the back-translation of sensical sentences, $f_\star^{-1}(S)$.

The translator family $\{f_\theta | \theta \in \Theta\}$ is taken to be a set of 10^5 random one-to-one translators, of which one is secretely chosen to be ground-truth $f_\star$. We then train an MLE algorithm on random samples from the source language: Each sample $x \sim \mu$ rules-out a subset of translators, namely, all $\theta \in \Theta$ such that $f_\theta(x) \notin S$, i.e., is nonsensical.

Figure 4.5 shows that as the number of samples increases, the average error over the plausible translators (that have not been ruled-out) decreases. To show how language complexity / common ground affect translatability, we ablate the parameter α which determines the fraction of common nonsense. Our experiments validate the intuition that increased common nonsense results in lower translation error. The error of a translator is computed as the fraction of disagreements with the ground-truth on a hold-out validation set of size 1000. The values with which the model is instantiated are detailed in Figure 4.7.

4.5 The knowledge-graph model

In this section, we define a model in which each text represents an edge between a pair of nodes in a knowledge graph. Both languages have knowledge graphs, with the source language weakly agreeing with an unknown subgraph of the target language.

We fix $\mathcal{X} = X \times X = X^2$ and $\mathcal{Y} = Y \times Y = Y^2$ with $r \coloneqq |X|$ and $n \coloneqq |Y|$. The set of translators considered is all mappings from the r source nodes to the n target nodes, namely $\Theta_{XY} = \{\theta : X \hookrightarrow Y\}$ and $f_\theta\big((u,v)\big) \coloneqq \big(\theta(u), \theta(v)\big)$. The random knowledge graph is parametrized by the number of source nodes r, target node set $\mathcal{Y}$, an edge density parameter $p \in (0,1)$ representing the expected fraction of edges present in each graph, and an agreement parameter $\alpha \in (0,1]$ representing the correlation between these edges. In particular, $\alpha = 1$ corresponds to the case where both graphs agree on all edges, and $\alpha = 0$ corresponds to the case where edges in the graphs are completely independent. These parameters are unknown to the learner, who only knows X and Y (and thus $\mathcal{X} = X^2, \mathcal{Y} = Y^2$).

Definition 4.17 (Random knowledge graph)**.** *For a natural number $r \le |Y|$, the* $\mathrm{KG} = \mathrm{KG}(Y, r, p, \alpha)$ *model determines a distribution over sets $T, P \subseteq \mathcal{Y}$ (which determine distributions ρ and μ). The sets T and P are sampled as follows:*

1. *Set $P \subseteq \mathcal{Y}$ is chosen by including each edge $y \in \mathcal{Y}$ with probability p, independently.*
2. *Set $S \subseteq Y$ of size $|S| = r$ is chosen uniformly at random.*
3. *Set $T \subseteq S^2$ is chosen as follows. For each edge $y \in S^2$, independently,*
   a) *With probability α, $y \in T$ if and only if $y \in P$.*
   b) *With probability $1 - \alpha$, toss another p-biased coin and add y to T if it lands on "heads"; that is, $y \in T$ with probability p, independently.*

It is easy to see that $T \subseteq S^2$ and $P \subseteq Y^2$ marginally represent the edges of Erdős–Rényi random graphs $G_{r,p}$ and $G_{n,p}$, respectively. Moreover, the event that $y \in T$ is positively correlated with $y \in P$: for each $y \in S^2$, since with probability $\alpha > 0$ they are identical and otherwise they are independent. Formally, the equations below describe the probability of $y \in T$ for each $y \in S^2$ after we fix S and choosing $T \subseteq S^2$. Letting $q \coloneqq (1-p)$, for each $y \in S^2$:

$$\Pr[y \in T] = \Pr[y \in P] = p \tag{4.21}$$

$$\Pr[y \in T \setminus P] = \Pr[y \in P \setminus T] = (1-\alpha)pq \tag{4.22}$$

$$\Pr[y \notin P \mid y \in T] = \Pr[y \notin T \mid y \in P] = \frac{(1-\alpha)pq}{p} = (1-\alpha)q \tag{4.23}$$

The last equality, shows that the probability of excluding a random $y \in T$ from P is smaller than the probability of excluding a random "incorrect translation" $y' \neq y$, $\Pr[y' \notin P] = q > (1-\alpha)q$.

We now describe how ρ, τ, are determined from T, P and how $\mu, \star$ may be chosen to complete the model description. The ground-truth target translated distribution $\tau \coloneqq \mathcal{U}(T)$ is uniform over T. The prior ρ is uniform over P, and then "smoothed" over the rest of the domain $\mathcal{Y}$. Formally,

$$\rho(y) \coloneqq \begin{cases} \frac{1}{2} \cdot \left(\frac{1}{|P|} + \frac{1}{|\mathcal{Y}|} \right) & \text{if } y \in P \\ \frac{1}{2|\mathcal{Y}|} & \text{if } y \notin P. \end{cases}$$

The ground-truth translator $\star \in \Theta$ is obtained by sampling a uniformly random $\star : X \hookrightarrow S$. Lastly, we take $\mu = \mathcal{U}(f_\star^{-1}(T))$, which agrees with the definition of τ.[13]

Next, we state the main theorem for this model, formalizing Theorem 4.2 from the introduction.

Theorem 4.18 (Translatability in the KG model)**.** *Fix any $m \geq 1$, $\emptyset \neq S \subseteq Y, \delta, \alpha, p \in (0,1)$, and let $r \coloneqq |S|, n \coloneqq |Y|, q = 1 - p$. Then, with probability $\geq 1 - \delta$ over T, P from* $\mathrm{KG}(S, Y, p, \alpha)$*,*

$$\mathrm{err}\left(\hat{\theta}\right) \leq \max\left(\frac{64}{\alpha^2 p q^2 r^2} \ln \frac{6n^r}{\delta}, \frac{2}{\alpha q} \sqrt{\frac{2}{m} \ln \frac{6n^r}{\delta}} \right),$$

where $\hat{\theta} = MLE^\rho(x_1, x_2, \ldots, x_m)$ is from Definition 4.1. Simply, for $p < 0.99$, with probability ≥ 0.99,

$$\mathrm{err}(\theta) = O\left(\frac{\log n}{\alpha^2 p r} + \frac{1}{\alpha} \sqrt{\frac{r \log n}{m}} \right).$$

4.5.1 Proving Theorem 4.18

Our goal in this section is to prove Theorem 4.18. The proof is based on the following main lemma. We first state and prove this lemma, and then derive the theorem from it. Recall that the sets $T, P \subseteq \mathcal{Y} = Y \times Y = Y^2$ represent the edges of the two knowledge graphs.

Lemma 4.19. *Fix $\emptyset \neq S \subseteq Y$, $\pi : Y^2 \hookrightarrow Y^2$, $\delta, p, \alpha \in (0,1)$, $q \coloneqq 1 - p$, and*

$$\varepsilon \geq 32 \cdot \frac{\ln(1/\delta)}{p \alpha^2 q^2 |S|^2}.$$

[13] Formally, the KG model outputs T, P which may not determine S if some nodes have zero edges. In that case, we choose $\star$ randomly among θ such that $f_\theta(\mathcal{X}) \supseteq T$. In the exponentially unlikely event that either S or T is empty, we define both τ, ρ to be the singleton distribution concentrated on (y, y) for the lexicographically smallest $y \in Y$ and μ to concentrated on (x, x) for $x = f_\star^{-1}(y)$. It is not difficult to see that MLE selects a translator with 0 error.

For any $(T, P) \sim \mathrm{KG}(S, Y, p, \alpha)$ *chosen from the random knowledge graph distribution, we define*

$$A \coloneqq \{y \in T \mid \pi(y) \neq y\}$$
$$B \coloneqq \{y \in A \mid \pi(y) \notin P\}$$
$$C \coloneqq \{y \in A \mid y \notin P\}$$

Then,

$$\Pr_{T,P}\left[|A| \geq \varepsilon|T| \ \wedge \ |B| - |C| \leq \frac{\alpha q}{2}|A|\right] \leq 5\delta.$$

Proof. Let $A' \coloneqq \{y \in S^2 \mid \pi(y) \neq y\}$, so $A \subseteq A'$. If π is the identity then the lemma holds trivially, therefore we can assume that $A' \neq \emptyset$. If $\varepsilon > 1$ then the lemma holds trivially as well, because $A \subseteq T$ and therefore $|A| \leq |T|$, so we assume $\varepsilon \in (0, 1]$.

For each $y \in A'$, Equations (4.21) and (4.22) under Definition 4.17 imply that $\Pr[y \in C] = \Pr[y \in T \setminus P] = (1-\alpha)pq$ and $\Pr[y \in A] = \Pr[y \in T] = p$, thus Bayes rule gives

$$\forall y \in A' \quad \Pr[y \in C \mid y \in A] = \frac{(1-\alpha)pq}{p} = (1-\alpha)q.$$

Now suppose we fix $V \subseteq A'$ and condition on $A \coloneqq A' \cap T = V$. Note that the event $y \in C$ is independent for different $y \in V$, therefore for any $y \in V$ it holds that

$$\Pr_{T,P}[y \in C \mid A = V] = \Pr_{T,P}[y \in C \mid y \in A] = (1-\alpha)q.$$

Therefore, $\mathbb{E}[|C| \mid A = V] = (1-\alpha)q|A|$, and so a Chernoff bound gives

$$\forall V \subseteq A' \quad \Pr_{T,P}\left[|C| \leq (1-\alpha)q|A| + \sqrt{\frac{1}{2}|A| \ln \frac{1}{\delta}} \ \middle| \ A = V\right] \geq 1 - \delta.$$

(Normally, Chernoff bounds would give the tight inequality that $\Pr[|C| < \ldots] \geq 1 - \delta$, but the $\leq$ in the above is necessary for the case in which $A = \emptyset$ in which case Chernoff bounds do not apply because it would be over $|A| = 0$ coin flips.) Since this holds for every V, we have:

$$\Pr_{T,P}\left[|C| \leq (1-\alpha)q|A| + \sqrt{\frac{1}{2}|A| \ln \frac{1}{\delta}}\right] \geq 1 - \delta. \tag{4.24}$$

By Proposition 4.16, we can partition A' into three **disjoint** sets,

$$A' = A'_1 \cup A'_2 \cup A'_3 \text{ such that } \pi(A'_i) \cap A'_i = \emptyset.$$

As above, we are going to condition on the value of $A_i \coloneqq A'_i \cap T$. Also, define,

$$B_i \coloneqq \{y \in A_i \mid \pi(y) \notin P\} = B \cap A_i.$$

Now, fix any $i \in [3]$ and any set $V \subseteq A'_i$. We now claim that for all $i \in [3]$, $V \in A'_i$, and $y \in V$

$$\Pr_{T,P}[y \in B_i \mid A_i = V] = \Pr_{T,P}[\pi(y) \notin P \mid y \in T] = q.$$

The rightmost equality follows from the fact that $\pi(y) \neq y$ so $\pi(y) \notin P$ is independent of $y \in T$. The leftmost equality follows similarly: Since $\pi(A'_i) \cap A'_i = \emptyset$, the event $A_i = V$ is independent of $\pi(y) \notin P$. Thus, again by Chernoff bounds we have

$$\forall i \in [3] \;\; \forall V \subseteq A'_i \;\; \Pr_{T,P}\left[|B_i| \geq q|A_i| - \sqrt{\frac{1}{2}|A_i| \ln \frac{1}{\delta}} \;\middle|\; A_i = V\right] \geq 1 - \delta.$$

Since this holds for all V, it holds unconditionally, and by the union bound it follows that

$$\Pr_{T,P}\left[\forall i \in [3] \;\; |B_i| \geq q|A_i| - \sqrt{\frac{1}{2}|A_i| \ln \frac{1}{\delta}}\right] \geq 1 - 3\delta. \tag{4.25}$$

Now, since the sets B_i partition B and A_i partition A, we have $|B| = \sum_i |B_i|$, $|A| = \sum_i |A_i|$, and also $\sum_{i=1}^3 \sqrt{|A_i|} \leq \sqrt{3|A|}$ by Cauchy–Schwartz. Thus, summing the three equations in Equation (4.25) probability implies

$$\Pr_{T,P}\left[|B| \geq q|A| - \sqrt{\frac{3}{2}|A| \ln \frac{1}{\delta}}\right] \geq 1 - 3\delta.$$

Combining with Equation (4.24) gives, by the union bound,

$$\Pr_{T,P}\left[|B| - |C| \geq q|A| - \sqrt{\frac{3}{2}|A| \ln \frac{1}{\delta}} - (1-\alpha)q|A| - \sqrt{\frac{1}{2}|A| \ln \frac{1}{\delta}}\right] \geq 1 - 4\delta.$$

Since $\sqrt{3/2} + \sqrt{1/2} \leq 2$, this implies:

$$\Pr_{T,P}\left[|B| - |C| \geq \alpha q|A| - 2\sqrt{|A| \ln \frac{1}{\delta}}\right] \geq 1 - 4\delta.$$

Or equivalently,

$$\Pr_{T,P}\left[|B| - |C| < \alpha q|A| - 2\sqrt{|A| \ln \frac{1}{\delta}}\right] \leq 4\delta.$$

Since adding additional restrictions can only reduce a probability, we have:

$$\Pr_{T,P}\left[\frac{p|S|^2}{2} \leq |T| \leq \frac{|A|}{\varepsilon} \wedge |B| - |C| < \alpha q|A| - 2\sqrt{|A| \ln \frac{1}{\delta}}\right] \leq 4\delta.$$

But if $\frac{p|S|^2}{2} \leq |T| \leq \frac{|A|}{\varepsilon}$ then $2|A| \geq \varepsilon p|S|^2$ and then, since $\varepsilon \geq \frac{32}{\alpha^2 q^2 p|S|^2} \ln \frac{1}{\delta}$:

$$2\sqrt{|A| \ln \frac{1}{\delta}} \leq 2\sqrt{\frac{2|A|}{\varepsilon p|S|^2} \cdot |A| \ln \frac{1}{\delta}} \leq 2|A| \sqrt{\frac{2}{p|S|^2 \frac{32}{\alpha^2 q^2 p|S|^2} \ln \frac{1}{\delta}} \ln \frac{1}{\delta}} = \frac{\alpha q}{2}|A|.$$

Thus,

$$\Pr_{T,P}\left[\frac{p|S|^2}{2} \leq |T| \leq \frac{|A|}{\varepsilon} \wedge |B| - |C| < \frac{\alpha q}{2}|A|\right] \leq 4\delta.$$

Since, in general, for any two events X and Y it holds that $\Pr[Y] \leq \Pr[X, Y] + \Pr[\overline{X}]$, we have

$$\begin{aligned}\Pr_{T,P}\left[|T| \leq \frac{|A|}{\varepsilon} \wedge |B| - |C| < \frac{\alpha q}{2}|A|\right] &\leq \Pr_{T,P}\left[\frac{p|S|^2}{2} \leq |T| \leq \frac{|A|}{\varepsilon} \wedge |B| - |C| < \frac{\alpha q}{2}|A|\right] + \\ &\quad + \Pr_{T,P}\left[\frac{p|S|^2}{2} > |T|\right] \\ &\leq 4\delta + \Pr_{T,P}\left[\frac{p|S|^2}{2} > |T|\right] \\ &\leq 4\delta + \delta,\end{aligned}$$

which is equivalent to the statement in the lemma. To see the last step above, note that $\mathbb{E}[|T|] = p|S|^2$ and thus by multiplicative Chernoff bounds,

$$\Pr\left[|T| < \frac{p|S|^2}{2}\right] \leq \exp\left(-\frac{p|S|^2}{8}\right) \leq \exp\left(-\frac{p|S|^2}{8} \cdot \frac{1}{\varepsilon} \cdot \frac{32}{\alpha^2 q^2 p|S|^2} \ln \frac{1}{\delta}\right) = \delta^{\frac{4}{\varepsilon \alpha^2 q^2}} \leq \delta.$$

In the last step we have utilized the fact that $\alpha, q, \delta \in (0, 1]$, and the fact (observed in the first paragraph of this proof) that we may assume that $\varepsilon \in (0, 1]$ else the lemma holds trivially. □

Using the above lemma, we now prove our main theorem regarding knowledge graphs.

Proof. of Theorem 4.18. Let $q \coloneqq 1 - p$ and,

$$\varepsilon \coloneqq \max\left(\frac{64}{\alpha^2 p q^2 |S|^2} \ln \frac{6n^{|S|}}{\delta}, \frac{2}{\alpha q}\sqrt{\frac{2}{m} \ln \frac{6n^{|S|}}{\delta}}\right).$$

For any $\theta \in \Theta$ define,

$$\begin{aligned}A_\theta &\coloneqq \{y \in T \mid \pi_\theta(y) \neq y\} \\ B_\theta &\coloneqq \{y \in A_\theta \mid \pi_\theta(y) \notin P\} \\ C_\theta &\coloneqq \{y \in A_\theta \mid y \notin P\}\end{aligned}$$

Note that since $\text{err}(\theta) = |A_\theta|/|T|$, our goal is to show that, with probability $\geq 1-\delta$, we will not output any θ with $|A_\theta| \geq \varepsilon|T|$.

Recall that $|\Theta| \leq n^{|S|}$. By Lemma 4.19 substituting $\delta' = \frac{1}{6n^{|S|}}\delta$ and the union bound over $\theta \in \Theta$ which is of size $|\Theta| \leq n^{|S|}$,

$$\Pr_{T,P}\left[\exists\theta \in \Theta \;\; |A_\theta| \geq \varepsilon|T| \wedge |B_\theta| - |C_\theta| \leq \frac{\alpha q}{2}|A_\theta|\right] \leq \frac{5\delta}{6}.$$

Using $\text{err}(\theta) = |A_\theta|/|T|$, this implies,

$$\begin{aligned}\Pr_{T,P}\left[\exists\theta \in \Theta \quad \text{err}(\theta) \geq \varepsilon \wedge |B_\theta| - |C_\theta| \leq \frac{\alpha q\varepsilon|T|}{2}\right] &\leq \frac{5\delta}{6}\\ \Pr_{T,P}\left[\exists\theta \in \Theta \quad \text{err}(\theta) \geq \varepsilon \wedge \frac{|B_\theta|}{|T|} - \frac{|C_\theta|}{|T|} \leq \frac{\alpha q\varepsilon}{2}\right] &\leq \frac{5\delta}{6} \qquad (4.26)\end{aligned}$$

Finally, define the empirical "errors" for any θ to be,

$$\hat{e}_\theta = \frac{1}{m}\{i \mid f_\theta(x_i) \notin P\}.$$

It is not difficult to see that the algorithm outputs a θ with minimal $\hat{e}_\theta$, and thus it will not output any θ with $\hat{e}_\theta - \hat{e}_\star > 0$. Now, it is also not difficult to see that $\hat{e}_\theta - \hat{e}_\star$ is the mean of m random variables in $\{-1, 0, 1\}$ and

$$\mathbb{E}[\hat{e}_\theta - \hat{e}_\star] = \Pr_{y\sim\tau}[\pi_\theta(y) \notin P] - \Pr_{y\sim\tau}[y \notin P] = \frac{|B_\theta|}{|T|} - \frac{|C_\theta|}{|T|}.$$

The last step above follows because $\pi_\star$ is the identity, and because if $y = \pi_\theta(y)$ then $y \in P \iff \pi_\theta(y) \in P$. (Formally, one may define $E_\theta \coloneqq \{y \in T \mid \pi_\theta(y) \notin P\}$ and observe that $B_\theta \subseteq E_\theta$, $C_\theta \subseteq E_\star$ and $E_\theta \setminus B_\theta = E_\star \setminus C_\theta$). Thus, by Chernoff bounds,

$$\forall\theta \in \Theta \quad \Pr_{x_1,\ldots,x_m}\left[\hat{e}_\theta - \hat{e}_\star \leq \frac{|B_\theta|}{|T|} - \frac{|C_\theta|}{|T|} + \sqrt{\frac{2}{m}\ln\frac{6|\Theta|}{\delta}}\right] \leq \frac{\delta}{6|\Theta|}.$$

By the union bound over $\theta \in \Theta$,

$$\Pr_{x_1,\ldots,x_m}\left[\exists\theta \in \Theta \;\; \hat{e}_\theta - \hat{e}_\star \leq \frac{|B_\theta|}{|T|} - \frac{|C_\theta|}{|T|} + \sqrt{\frac{2}{m}\ln\frac{6|\Theta|}{\delta}}\right] \leq \frac{\delta}{6}.$$

Combining with Equation (4.26) gives,

$$\Pr_{T,P}\left[\exists\theta \in \Theta \quad \text{err}(\theta) \geq \varepsilon \wedge \hat{e}_\theta - \hat{e}_\star \leq \frac{\alpha q\varepsilon}{2} - \sqrt{\frac{2}{m}\ln\frac{6|\Theta|}{\delta}}\right] \leq \frac{5\delta}{6} + \frac{\delta}{6} = \delta.$$

Since, for our choice of $\varepsilon \geq \frac{2}{\alpha q}\sqrt{\frac{2}{m}\ln\frac{6|\Theta|}{\delta}}$,

$$\Pr_{T,P}\left[\exists\theta\in\Theta \quad \mathrm{err}(\theta)\geq\varepsilon\wedge\hat{e}_\theta-\hat{e}_\star\leq 0\right]\leq\delta.$$

Put another way,

$$\Pr_{T,P}\left[\forall\theta\in\Theta \quad \mathrm{err}(\theta)\leq\varepsilon\vee\hat{e}_\theta-\hat{e}_\star>0\right]\geq 1-\delta.$$

We claim that we are done: Observe that if MLE outputs some $\theta\neq\star$ then, $\hat{b}_\theta\leq\hat{b}_\star$. To see this, recall the definition of the prior ρ,

$$\rho(f_\theta(x)) \coloneqq \begin{cases}\frac{1}{2}\cdot\left(\frac{1}{|P|}+\frac{1}{|\mathcal{Y}|}\right) & \text{if } f_\theta(x)\in P\\ \frac{1}{2|\mathcal{Y}|} & \text{if } f_\theta(x)\notin P.\end{cases}$$

and therefore the objective function minimized by MLE, namely, $\frac{1}{m}\sum_{i=1}^m -\log(\rho(f_\theta(x_i)))$, is strictly monotonic in $\hat{b}_\theta$:

$$\begin{aligned}\frac{1}{m}\sum_{i=1}^m -\log(\rho(f_\theta(x_i))) &= \hat{b}_\theta\cdot\log\frac{2}{1/|\mathcal{Y}|}+(1-\hat{b}_\theta)\log\frac{2}{1/|P|+1/|\mathcal{Y}|}\\ &= \log\frac{2}{1/|P|+1/|\mathcal{Y}|}+\hat{b}_\theta\cdot\log\frac{1/|P|+1/|\mathcal{Y}|}{1/|\mathcal{Y}|}\end{aligned}$$

so the θ output by MLE necessarily minimizes b_θ.

Finally, for the simplification in the theorem, note that for $p<0.99$, $1/q<100$ is at most a constant and note that a maximum is never more than a sum. □

It is interesting to note that it is possible to prove the same theorem using a generalization of Plausible Ambiguities, though we use the shorter proof above here because it is somewhat more involved. This generalization may be useful for other priors of full support. Many LMs, in practice, assign non-zero probability to every string due to softmax distributions or a process called "smoothing." A full-support prior ρ has full support, then $\mathcal{A}_\gamma=\Theta$ and so the parameter ε_γ becomes too large to be meaningful even for $\gamma=0$. To address this, we refine our definition of plausible ambiguities as follows. For generality, we state them in terms of arbitrary loss $\mathcal{L}$, though we only use them for the semantic error $\mathcal{L}=\mathrm{err}$.

Definition 4.20 ((γ,κ)-plausible ambiguities)**.** *For any $\gamma,\kappa\in[0,1]$, the set of (γ,κ)-plausible ambiguities is:*

$$\mathcal{A}_{\gamma,\kappa}\coloneqq\left\{\theta\in\Theta \;\middle|\; \Pr_{y\sim\tau}[\rho(\pi^\star_\theta(y))\leq\kappa]\leq\gamma\right\},\ \textit{and}\ \varepsilon_{\gamma,\kappa}\coloneqq\max_{\theta\in\mathcal{A}_\gamma}\mathcal{L}(\theta).$$

Furthermore, $\mathcal{A}_\gamma=\mathcal{A}_{\gamma,0}$ and $\varepsilon_\gamma=\varepsilon_{\gamma,0}$.

Name	Symbol	Value
Number of source nodes	r	$1, 4, 7, 10$
Number of target nodes	n	10
Number of training data	m	$1, 2, \ldots$ up to all edges
Edge density (probability of including an edge)	p	0.5
Agreement parameter	α	$0, 0.33, 0.66, 1$

Figure 4.8: **Parameters for experiments in the knowledge graph model (Figure 4.4).** For ablations on r we take $\alpha = 0.5$, and for ablations on α we take $r = 9$. The experiments were run in parallel on an AWS r6i.4xlarge for a total of two and a half CPU-hours.

4.5.2 Experiments

In the knowledge graph model, text describes relations between nodes in a directed graph. Due to computational constraints, we consider ten nodes, each corresponding to a different word in the target language. To generate edges corresponding to the target language P, two nodes are connected with a directed edge independently, with probability 0.5. We then consider source languages with $r \leq 10$ words. Given a ground-truth translator $f_\star \colon [r] \to [10]$, the source language graph T is obtained by choosing a random subset of nodes S of size r, taking the pre-image of graph induced on S under $f_\star$, and (3) adding noise by redrawing each edge with probability $1 - \alpha$ for a fixed agreement coefficient $\alpha \in (0, 1)$.

The prior ρ is derived from the edges of P, and the source language μ is derived from the (noisy) permuted subgraph T. We consider the translator family $\{f_\theta | \theta \in \Theta\}$ of all node-to-node (word-to-word) injective translators, of which one is secretly chosen to be ground-truth. Similarly to the previous setting, we train an MLE algorithm on randomly chosen edges from T, which correspond to sentences in the source language. For each sampled edge (x_1, x_2), we increase the "score" of each translator that agrees with the edge, that is, that $(f_\theta(x_1), f_\theta(x_2))$ is an edge in the graph P.

To show how common ground affects translatability, we ablate the parameter α determines the fraction of edges on which the source language graph T and the target language graph P agree. Figure 4.4 validates the intuition that increased agreement results in lower translation error, and that as the number of samples increases, the error of the top-scoring translator decreases.

To show how language complexity affects translatability, we ablate r, which is the size of the subgraph corresponding to the source language. Figure 4.4 (right) validates the intuition that a larger subgraph results in lower translation error.

The error of a translator is computed as the fraction of edges whose labels are different than the ground-truth. The values with which the model is instantiated are detailed in Figure 4.8.

4.6 Generalizing the framework

4.6.1 Lossy Translation

Many things can be included in textual transcriptions suitable for translation. For instance, one can distinguish the speakers, e.g., "*Whale 1: ... Whale 2: ... Whale 1: ...*" if the source data is annotated with speaker identifiers. Some aspects of the way one is speaking can be transcribed, e.g., "*Whale 1 (fast tempo clicking): ... Whale 2 (slow, loud clicking): ...*" It may be possible to encode these textually in x.

However, if x is encoded in a more flexible format, such as arbitrary binary files, one can include as much raw information as possible, including the source audio recording, to provide the translator with as much context as possible. In that case, lossless translation will no longer be possible, because one cannot compute the raw x from a textual translation.

Given the possible benefits of such annotations, we propose an extension of our theory to the lossy setting. A natural generalization of the maximum-likelihood approach is as follows:

$$\min_{\theta\in\Theta} \frac{1}{m}\sum_{i=1}^{m} -\log \rho(f_\theta(x_i)) - \frac{1}{\lambda}\log \phi_\theta(x_i \mid y = f_\theta(x_i)).$$

Here $\phi_\theta \colon \mathcal{X}\times\mathcal{Y} \to [0,1]$ is a probabilistic inverse ("randomized back translation") of f whose parameters are also encoded in θ. Note that the family $\{(f_\theta, \phi_\theta) \mid \theta \in \Theta\}$ must satisfy that for all $y \in \mathcal{Y}$, $\sum_{x\in f^{-1}(y)} \phi_\theta(x \mid y) = 1$, though it is no longer required that f_θ be 1–1.

As λ decreases to 0, the optimal solution would assign infinite loss to any $f_\theta \colon \mathcal{X} \hookrightarrow \mathcal{Y}$ and ϕ_θ that are not perfect inverses, where there is some x (with positive probability under μ) such that $\phi_\theta(x \mid y = f_\theta(x)) < 1$. Thus, for sufficiently small λ, the algorithm is exactly the minimum cross-entropy (maximum likelihood) algorithm of Definition 4.1. For arbitrarily large λ, the algorithm will collapse to always outputting the most likely $y \in \mathcal{Y}$ under ρ. For example, everything could be translated to *Hello* regardless of its contents.

For intermediate values of λ, the chosen translator trades off naturalness in the form of $\rho(f_\theta(x))$ versus information loss which is inversely related to $\phi_\theta(x_i \mid y = f_\theta(x_i))$. This trade-off makes it challenging to define and analyze the success of a lossy unsupervised translation algorithm. Nonetheless, the algorithm is intuitive.

4.6.2 Infinite parameter sets Θ

In some cases, one can learn with many fewer examples, which is important when parameters are real-valued and $|\Theta| = \infty$ or when the model is over-parameterized. However, one can analyze even these cases in our model using the following "trick." Suppose one has a supervised translation algorithm SUPER that takes m labeled examples (x_i, y_i) as input and outputs $\theta \in \Theta$. A simple observation in these cases is that one could use an initial set of m unlabeled examples, $x_1, \ldots, x_m$, to define a subset of translators:

$$\overline{\Theta} \coloneqq \left\{ \textsc{Super}\big((x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\big) \mid y_1, y_2, \ldots, y_m \in \mathcal{Y} \right\}.$$

Sentence	Probability
I just ate a giant squid.	1.5×10^{-9}
I just ate a giant cheeseburger.	$\mathbf{1.9 \times 10^{-8}}$
A sperm whale said: I just ate a giant squid.	$\mathbf{6.8 \times 10^{-14}}$
A sperm whale said: I just ate a giant cheeseburger.	1.2×10^{-17}

Figure 4.9: Without using a prompt, the sentence *I just ate a giant cheeseburger* is more likely, but using the prompt *A sperm whale said:*, the sentence *I just ate a giant squid* is much more likely. Probabilities are from the GPT-3 API.

That is, we restrict attention to the set of possible translators that we could output for any given ground-truth translations, then it is not difficult to see $\log|\overline{\Theta}| \leq m \log|\mathcal{Y}|$. If we assume that one of these is accurate and natural, then restricting the attention of MLE to this set will suffice, and Theorem 4.6 means that the number of examples required is $O(\log|\overline{\Theta}|) = O(m \log|\mathcal{Y}|)$ which is a linear blowup in the number of examples m used for supervised translation. To make this formal, one would start from only assuming that one of the translators had negligible error—this is left to future work.

4.7 Where might we find a good prior?

The most direct way to improve a prior is to train (or fine-tune) the LM on a dataset that includes a large number of articles relevant to the source language, e.g., volumes of oceanic research, for whales. In addition, training on a wide variety of sources including multiple languages, diverse sources, and encoding systems may be helpful. It is possible that a system that has how to transfer knowledge between hundreds of languages and even programming languages, may have a better prior.

Another general strategy for creating a prior is to use prompting: Given a *prompt* string s, one can define $\rho(y) \propto \nu(s\,y)$, that is the prior distribution of text that that LM generates conditioned on the text beginning with s. Figure 4.9 illustrates some toy examples of how even a simple prompt like *A sperm whale said:* can help focus on translations that are more likely for a sperm whale to say, and eliminate irrelevant translations.

Background prompts. There is a natural and potentially powerful idea that an unsupervised translator, in addition to outputting translations, would automatically generate a *background prompt* that increases the intelligibility of many translations.[14] Suppose, for example, across numerous communications, the unsupervised translator determines that sperm

[14]In general, the problem of AI-based prompt generation has recently attracted attention, e.g., Shin et al., 2020.

whales measure time in "number of naps" and that typical nap duration varies by age. Then, rather than having to repeatedly explain this in each translation, it can be explained once in a background prompt that is automatically inserted before each translation. For example, following the prompt

$s =$ *Sperm whales measure time in numbers of naps, but each whale's typical nap duration depends on their age. So if a whale's nap duration is 9 minutes, then 4 naps is about 36 minutes (though whales tend to exaggerate times). A sperm whale said:*

would make translations like *Wow, I just dove for 6 naps* or *How many naps ago was that?* both more likely and more understandable.

4.8 Conclusion

We have given a framework for unsupervised translation and instantiated it in two stylized models. Roughly speaking, in both models, the error rate is inversely related to the amount of samples, common ground, and the language complexity. The first two relations are intuitive, while the last is perhaps more surprising. All error bounds were *information-theoretic*, meaning that they guarantee a learnable accurate translator, but learning this translator might be computationally intensive.

In both models, the translators are *restricted*. In the knowledge graph, the translators must operate node-by-node following an assumed compositional language structure.[15] In the common nonsense model, the restriction is based on the translator description bit length $\log|\Theta|$. To illustrate how such restrictions can be helpful, consider *block-by-block* translators which operate on limited contexts (e.g., by paragraph). Consider again the hypothetical example of Figure 4.1. Suppose the three texts are outputs of three translators $\Theta = \{A, B, C\}$. Let us suppose that translator A always produces accurate and natural translations, and further that all translators work paragraph-by-paragraph, as modern translation algorithms operate within some limited context window. In fact, one can imagine the translators of different paragraphs as a set of *isolated adversaries* where each adversary is trying to mistranslate a paragraph, knowing the ground-truth translation of their paragraph, while attempting to maintain the plausibility of the entire translation. If only the first-paragraph adversary mistranslates *reef* to *ocean basin*, then the translation lacks coherence and is unlikely. If the adversaries are in cahoots and coordinate to all translate *reef* to *ocean basin*, they would generate: *Have you seen mom? I just returned from the ocean basin. At the basin, there were a lot of sea turtles.* which has low probability $\approx 10^{-25}$, presumably because encoded in GPT-3's training data is the knowledge that there are no turtles deep in the ocean near the basin. While the adversary could also decide to change the word *turtle* to something else when it

[15] That is, we assume that each translator has a latent map from nodes in the source graph into nodes in the target graph, and edges are mapped from the source to target graphs in the natural way. The study of compositional communication systems, among humans and animals, has played a central role in linguistics (Zuberbühler, 2020).

appears near *basin*, eventually it would get caught in its "web of deceit." The intuition is that, across sufficiently many translations, the prior will not "rule out" the ground-truth translations while very incorrect translators will be ruled out.

Judging success. Our analysis sheds some light on whether it is even possible to tell if translation without parallel data (UMT) is successful. A positive sign would be if millions of translations are fluent English accounts that are consistent over time across translations. In principle, however, this is what LM likelihood should measure (excluding consistencies across translations which sufficiently powerful LMs may be able to measure better than humans). We also considered a statistical distance (KL divergence) between the translations $f_{\hat{\theta}}(x)$ for $x \sim \mu$ and the prior $y \sim \rho$, and μ could be estimated given enough samples. If this distance is close to zero, then one can have predictive accuracy regardless of whether the translations are correct. This raises a related philosophical quandary: a situation in which two beings are communicating via an erroneous translator, but both judge the conversation to be natural.

Future Work. Our initial exploration leaves plenty of room for future work. In particular, we propose the following possible directions:

1. In our lossless models, the target language subsumes the source language in the sense that everything that is representable in the source language can also be represented in the target language. It would be interesting to extend our work to the partially-overlapping case.

2. The language distribution in our models are all uniform. It would be interesting to examine non-uniform distributions such as Zipfian or power-law distributions.

3. As stated earlier, our analysis is purely information-theoretic and leaves the question of the *efficiency* of UMT open.

4. A good starting point for the efficiency question would be to design efficient UMT algorithms for one of the randomized models of language presented in this chapter.

Chapter 5

Towards A Translative Model of Sperm Whale Vocalizations

Understanding the communication of sperm whales (*Physeter macrocephalus*) is among the most fascinating questions in animal behavioral studies.

Sperm whales communicate using *codas*—short sequences of clicks that vary in number, rhythm, and tempo (Watkins and Schevill, 1977; Weilgart and Whitehead, 1993; Sharma et al., 2024a). They live in stable, female-led social units that form larger vocal clans based on *dialect* (Rendell and Whitehead, 2003). That is, the dialect of a social unit determines its clan, with social units associating exclusively with other units from their clan (Gero et al., 2016). Furthermore, dialects are believed to be learned socially rather than inherited genetically (Cantor and Whitehead, 2015; Rendell et al., 2012).

The complexity of these learned vocal patterns has motivated new computational approaches to understanding codas and their functionality. Leitao et al. (2024) modeled codas as (variable-length) Markov chains, revealing new patterns of inter-clan social learning. Beguš et al. (2023) study vowel-like spectral properties of codas, which were initially suggested by interpreting the codebook of a Generative Adversarial Network (GAN). Most recently, Sharma et al. (2024b) train a transformer on click timings (inter-click intervals), which is able to predict codas in an exchange based on long-term dependencies, as well as future diving behavior. These studies collectively highlight how machine learning—particularly transformer architectures—can decode patterns imperceptible to traditional methods.

Transformers (Vaswani et al., 2017) originated in natural language translation, where they revolutionized the field by enabling high-quality, context-aware machine translation. Whereas transformers have since become ubiquitous across machine learning (e.g. Chen et al. 2021; Khan et al. 2022; Moussad, Roche, and Bhattacharya 2023), in this work we propose again to use transformers towards translation—of animal communication.

While transformers have been used in settings where parallel data is nonexistent (Conneau and Lample, 2019) and for translation from audio (Kano, Sakti, and Nakamura, 2021), applying these advances to animal communication presents deep challenges. Even merely defining the problem has been the subject of studies spanning theoretical computer sci-

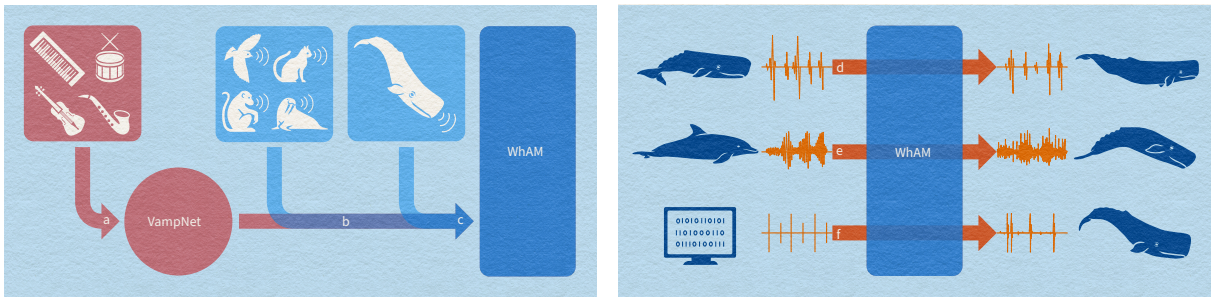


Figure 5.1: Left: WhAM is trained by finetuning VampNet (García et al., 2023), an audio-to-audio transformer pretrained on a large music dataset (a). Namely, we perform **domain adaptation** (b) on animal vocalizations followed by **species-specific finetuning** (c) on a novel sperm whale coda dataset. Right: WhAM synthesizes **context-aware variations** (d) of input codas and **acoustically translates** (e) natural and (f) artificial audio into coda-like audio. Illustration ©Alex Boersma.

ence (Goldwasser et al., 2023), biology (Yovel and Rechavi, 2023; Amphaeris et al., 2023), linguistics (Berwick and Chomsky, 2016; Amphaeris, Shannon, and Tenbrink, 2022), and philosophy (Suzuki, Wheatcroft, and Griesser, 2020; Hobaiter, Graham, and Byrne, 2022).

Existing approaches to modeling sperm whale codas have made significant advances in our understanding of sperm whale codas. Bermant et al. (2019) developed effective methods for coda detection and classification, while generative models based on GANs (Beguš, Leban, and Gero, 2023; Kopets et al., 2024) have shown the potential for synthesizing coda-like audio. The aforementioned timing-based analyses of Leitao et al. (2024) and Sharma et al. (2024b) have yielded new insight into the social and behavioral aspects of sperm whale communication.

Our work will address challenges left open by these works: While GAN-based models can generate coda-like audio (Beguš, Leban, and Gero, 2023; Kopets et al., 2024), they cannot easily condition on a given context. Timing-based approaches (Leitao et al., 2024; Sharma et al., 2024b) capture important temporal patterns but may miss features only present in the raw audio, such as the recently discovered vowels (Beguš et al., 2023). Moreover, current methods train separate models for classification (Bermant et al., 2019) and generation, despite the intuition that a model capable of realistic generation should also learn representations useful for classification. Lastly, none of these tackled the issue of *translating across acoustic domains*.

To address these challenges, we introduce the Whale Acoustics Model (WhAM, Figure 5.1), a new approach to modeling sperm whale codas that unifies three capabilities:

- **Acoustic translation:**[1] WhAM can translate an audio prompt (e.g. other animal

[1]We emphasize that translation is in the acoustic sense; semantic translation remains a distinct and more

vocalizations or even noise) into the acoustic style of sperm whale codas, acting as a form of cross-domain style transfer.

- **Generation:** WhAM can generate novel "pseudocodas" that are perceptually similar to real codas, as evaluated by expert listeners.
- **Classification:** WhAM's learned representations are useful for a range of classification tasks, including rhythm type (Sharma et al., 2024a), social unit classification (Best, 1979; Christal and Whitehead, 2001; Gero, Whitehead, and Rendell, 2016a), and the recently discovered vowel-like features of Beguš et al. (2023)—despite being trained primarily for generation.

Contributions. This chapter presents the first unified model of sperm whale codas capable of acoustic translation, generation, and classification. Notably, WhAM demonstrates that meaningful bioacoustic features emerge from purely generative training, aligning with recent work on self-supervised (non-generative) modeling of animal vocalizations (Hagiwara, 2023).

WhAM serves as a proof of concept, applying advances in neural audio modeling to bioacoustics in a novel and unifying way. To facilitate further research, we will release the model and its training and evaluation code. Remarkably, WhAM achieves strong results after just five days of training on a single GPU. While the dataset is small compared to those used for large audio models (Borsos et al., 2023; Agostinelli et al., 2023), our results suggest that scaling up could yield even greater improvements.

Finally, WhAM was developed in close collaboration with marine biologists and underwater acousticians with domain expertise in sperm whale vocalizations. The model was evaluated through perceptual studies conducted by an interdisciplinary team of specialists. To our knowledge, this is the first work to evaluate the perception of experts on synthetically generated codas, igniting a crucial discourse for validating the utility of generative models in bioacoustics research.

5.1 Sperm whale vocalizations

Sperm whales have evolved remarkable acoustic capabilities. Figure 5.2 illustrates the key anatomical and acoustic aspects of these capabilities, which form the basis for their complex communication system.

Sperm whales live in a multileveled social structure with female lines living together in 'units' with stable membership (Whitehead, 2003). Early acoustic research proposed that codas might serve as individual signatures (Watkins and Schevill, 1977), but subsequent studies instead suggested that different coda types may have distinct functions (Antunes et al., 2011), and that variation of coda usage among units suggested a function in unit-level social identity (Moore, Watkins, and Tyack, 1993; Weilgart and Whitehead, 1993; Weilgart

ambitious goal.

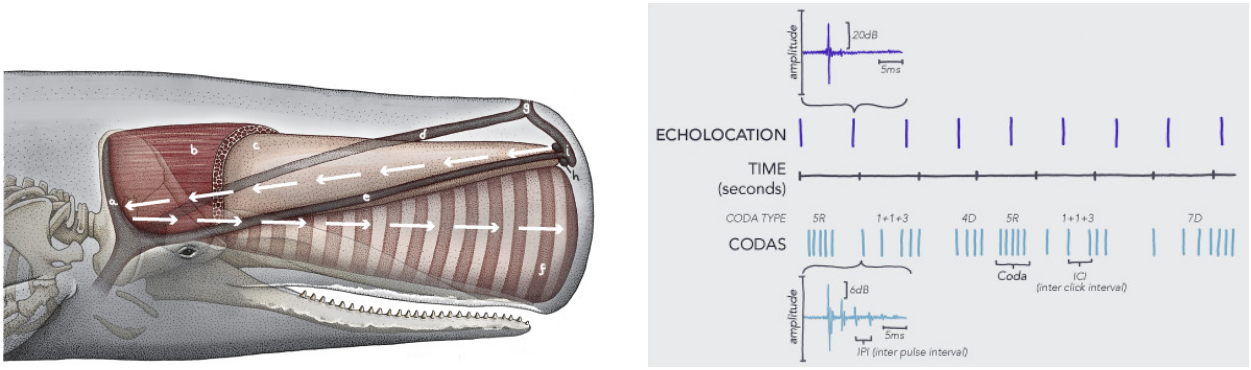


Figure 5.2: **Left:** The sperm whale head contains the spermaceti organ (c), a cavity filled with almost 2kL of wax-like liquid, and the junk compartment (f), comprising a series of wafer-like bodies believed to act as acoustic lenses. The spermaceti organ and junk act as two connected tubes, forming a bent, conical horn of about 10m in length and 0.8m aperture in large mature males. The sound emitted by the phonic lips (i) in the front of the head is focused by traveling through the bent horn, producing a flat wavefront at the exit surface. **Right:** Typical temporal structure of sperm whale echolocation and coda clicks. Echolocation signals are produced with consistent inter-click intervals (of approximately 0.4s) while coda clicks are arranged in stereotypical sequences called "codas" lasting less than 2s. Codas are characterized by the different number of constituent clicks and the intervals between them (called inter-click intervals). Codas are typically produced in multi-party exchanges that can last from about 10s to over half an hour. Each click, in turn, presents itself as a sequence of equally spaced pulses, with inter-pulse interval of an order of 3–4ms in an adult female, which is the result of the sound reflecting within the spermaceti organ. Figures and captions reproduced with permission from Andreas et al. (2022a).

and Whitehead, 1997). Even when living in the same waters, whales from different social units will only associate with units which share a similar repertoire of codas. This social segregation based on acoustic similarity was used to delineate the highest level of social organization which structures their populations, the vocal clan; and that codas function as symbolic markers of these cultural groups (Rendell and Whitehead, 2003; Gero et al., 2016; Hersh et al., 2022). Importantly, there is good evidence that these distinct dialects of codas, with variation in number of clicks, as well as rhythm and tempo, are the product of social learning, and not genetically inherited (Cantor and Whitehead, 2015; Rendell et al., 2012).

5.2 Training the Whale Acoustics Model

5.2.1 Masked Acoustic Token Modeling with VampNet

VampNet (García et al., 2023) is an audio-to-audio generative model, pretrained on 797k music tracks from thousands of artists. VampNet consists of three neural models: a tokenizer, a coarse-token model, and a coarse-to-fine model. For simplicity of presentation we will avoid the distinction between coarse and fine tokens, instead decomposing VampNet into an *Acoustic Tokenizer* and a *Masked Acoustic Token Model*. The reader is referred to García et al. (2023) for full details of the model, and Section 5.6.2 for a specification of hyperparameters used in training WhAM.

Acoustic Tokenizer. The tokenizer takes as input an N_{sec}-second audio snippet sampled at N_{sam} Hz, and outputs a sequence of ℓ discrete tokens from a finite vocabulary Σ. A jointly-trained detokenizer will convert token sequences back into audio:

$$T\colon \mathsf{R}^{N_{\text{sec}} \times N_{\text{sam}}} \to \Sigma^{\ell}$$
$$T^{-1}\colon \Sigma^{\ell} \to \mathsf{R}^{N_{\text{sec}} \times N_{\text{sam}}}.$$

VampNet uses a residual vector quantization approach known as the Descript Audio Codec (DAC, Kumar et al. 2023). At a high level, audio is tokenized in a temporal and hierarchical fashion, such that each interval of samples is replaced with a "stack" of tokens; this means that neighboring stacks of tokens correspond to contiguous intervals of samples in the audio. For example, the first five token stacks $(\sigma_1, \ldots, \sigma_5)$ could correspond to the first 0.5 seconds of audio.

Masked Acoustic Token Model (MATM). A bidirectional transformer M is trained to perform the cloze task on acoustic token sequences. That is, each audio snippet in the pretraining dataset is tokenized, and then a bidirectional transformer is trained to predict a random subset of masked tokens.

$$M\colon (\Sigma \cup \{[\texttt{MASK}]\})^{\ell} \to \Sigma^{\ell}$$

A pretrained MATM can be finetuned in various ways. Following García et al. (2023), we finetune using Low Rank Adaptation (LoRA, Hu et al. 2022).

Generation. After training a tokenizer T, detokenizer T^{-1} and a (possibly finetuned) MATM M, VampNet can be used to generate variations of given "prompt" audio snippets. This is done in the natural way, by randomly masking the tokenized audio; importantly, the masking scheme used in generation time does *not* need to be uniformly random. For example, the scheme can leave (classically-detected) *beats* unmasked, so as to preserve the rhythm of the prompt. Rather than generating all masked tokens simultaneously (e.g. as in BERT, Devlin et al. 2019), VampNet uses *iterative parallel decoding* (Chang et al., 2022) wherein tokens are gradually "unmasked" in a sequence of forward passes through the model.

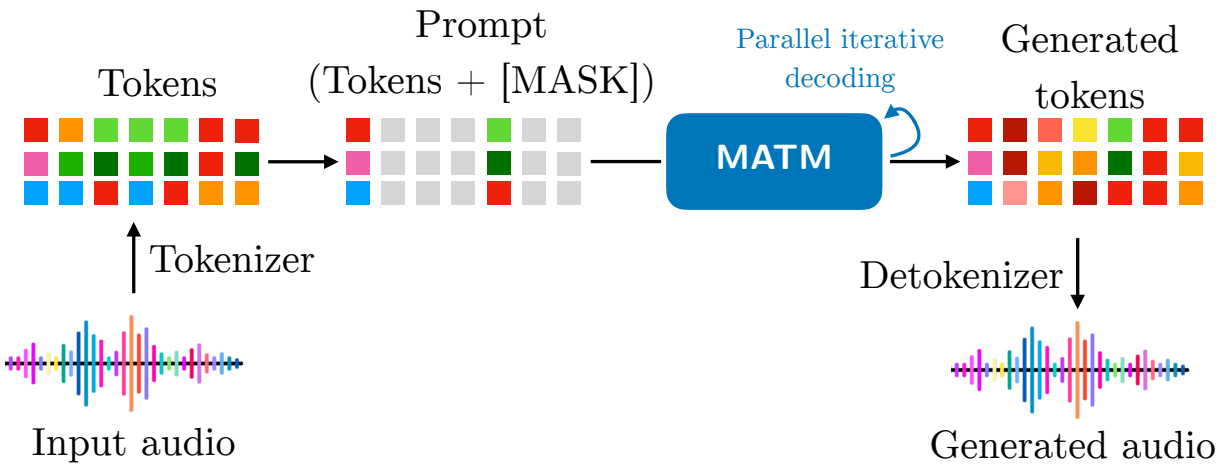


Figure 5.3: **Overview of VampNet's generation pipeline.** Input audio is first converted into a grid of tokens by the Tokenizer. These tokens are then partially masked to create a prompt. The Masked Acoustic Token Model (MATM) uses parallel iterative decoding to generate new tokens, which are finally converted back into audio by the Detokenizer. The colored squares represent acoustic tokens, with grey squares indicating masked positions.

5.2.2 Data

WhAM is trained by finetuning VampNet (Section 5.2.1) on various datasets.

FSD. The Freesound Dataset (Font, Roma, and Serra, 2013) consists of 50k human-labeled recordings. We used recordings with the *animal* tag, which totaled 7h45m of audio.

AudioSet. A dataset of two million human-labeled audio clips taken from YouTube (Gemmeke et al., 2017a). Of these, we used audio with the *animal* tag, totaling at about 5 hours.

WMMS. The Watkins Marine Mammal Sound Database (Sayigh et al., 2016) totaling 4h8m. It includes audio collected over seven decades in at least 67 sites around the world. Sperm whales are among the 51 species recorded.

DSWP. A dataset of 2507 annotated codas (1h26m) collected over thirteen years in a 2000km^2 area off the coast of Dominica. It consists of codas recorded using far-field boat-based hydrophones and noninvasive animal-borne tags.

CETICETI. A growing dataset of sperm whale vocalizations consisting of 7653 annotated codas (4h33m) at time of model training.

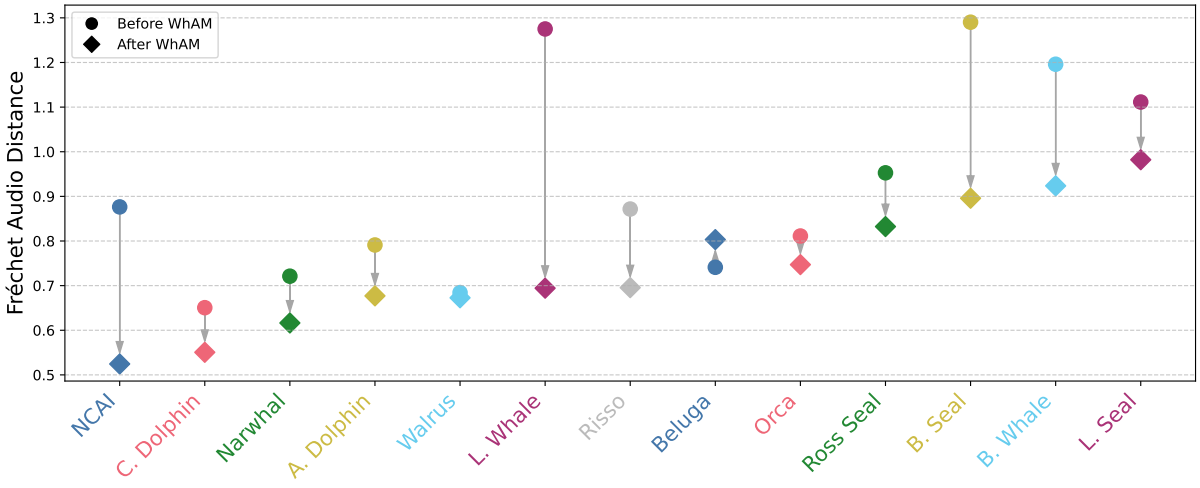


Figure 5.4: Fréchet Audio Distance between natural sperm whale codas and various audio sources, before and after translation through WhAM. Lower FAD indicates greater acoustic similarity to natural codas. Full names of animals along with the number of samples from each can be found in Table 5.3

The training of WhAM is split into two phases: (1) *Domain adaptation*, in which the base VampNet is finetuned on FSD+AudioSet+WMMS for 500k iterations; (2) *species-specific finetuning*, in which domain-adapted VampNet is finetuned on DSWP+CETI for an additional 500k iterations. Both phases follow the same (LoRA) finetuning procedure, but we find this split to be conceptually useful. Additional details are deferred to Section 5.6.1

5.3 Experimental results

We evaluate WhAM through three complementary analyses. First, we assess the quality of WhAM's synthetic codas through quantitative metrics, specifically the Fréchet Audio Distance (FAD, Kilgour et al. 2019) between generated and natural codas. Second, we conduct a perceptual study with expert marine biologists to evaluate how well our synthetic codas preserve the characteristic features of natural sperm whale vocalizations. Finally, we evaluate WhAM's learned representations on downstream classification tasks to investigate whether our model captures meaningful acoustic features of sperm whale communication.

5.3.1 Fréchet Distance of Audio Translation

A key aspect of WhAM is its ability to "translate" audio inputs into the acoustic style of sperm whale codas. To evaluate this capability quantitatively, we measure the Fréchet Audio Distance (FAD, Kilgour et al. 2019) between natural and WhAM-generated synthetic codas.

FAD measures the similarity between two audio datasets by comparing embeddings of the audio signals; lower FAD indicates greater acoustic similarity between the datasets.

While FAD can use embeddings from various models, we chose the Contrastive Language-Audio Pretraining (CLAP, Wu et al. 2023) based on a principled calibration experiment that compared the sensitivity of different embeddings to the rhythmic patterns crucial to coda structure (Section 5.4.1). We evaluate WhAM's translation ability using audio prompts from three domains:

1. *Natural codas*: Recordings of codas produced by sperm whales (see Section 5.2.2). These natural codas serve as a control group, as the FAD of a dataset to itself is zero. When passing natural codas through WhAM, we expect a increase in FAD due to the distortion introduced by the translation process. Indeed, the FAD between the original and WhAM-processed natural codas increased from 0 to 0.288.

2. *Animal sounds*: Vocalizations from 12 species of marine mammals. Figure 5.4 shows that WhAM consistently reduces the acoustic distance to natural codas, effectively translating these diverse inputs into the acoustic style of sperm whale codas. Among the 12 mammals tested, the Beluga vocalization samples were the only audio source for which WhAM failed to produce a better output, possibly due to the subjectively louder noise profile present in the Beluga audio compared to the remainder of the WMMS dataset (Sayigh et al., 2016).

3. *Non-Coda Acoustic Impulses (NCAI)*: Artificial samples generated by initializing an array of zeros and randomly selecting points to assign a peak amplitude of 1. WhAM reduced the FAD of these samples from 0.88 to 0.52, demonstrating its ability to translate even highly abstract acoustic patterns into coda-like structures.

The FAD results across these three domains demonstrate WhAM's effectiveness in capturing and translating the essential temporal patterns of coda structures. The slight increase in FAD for natural codas serves as a baseline, quantifying the distortion introduced by the translation process itself. The consistent FAD reduction for animal sounds and NCAI samples showcases WhAM's ability to project diverse acoustic inputs onto the manifold of sperm whale vocalizations.

5.3.2 Expert Perceptual Study

To evaluate the perceptual quality of WhAM's synthetic codas, we conducted a comprehensive study with domain experts to assess how well our generated outputs match natural sperm whale vocalizations. This study aimed to measure both audio-only and spectrogram-based discrimination performance, while also gathering qualitative insights about specific acoustic features that distinguish synthetic from natural codas. Additional details are deferred to Section 5.6.5.

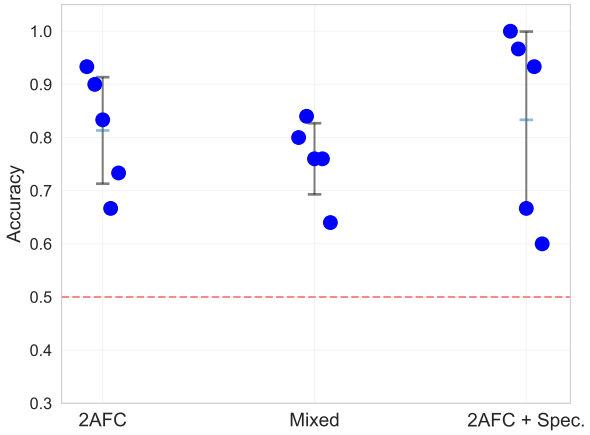


Figure 5.5: Expert performance on audio-only 2AFC (Task 1), mixed classification (Task 2), and spectrogram-assisted 2AFC (Task 3). Error bars show standard deviation across experts. While all tasks elicited above-chance performance (dashed line), spectrogram analysis showed the greatest variability between experts ($\sigma = 0.17$). Task 1 and 3 had 30 items each, Task 2 had 25.

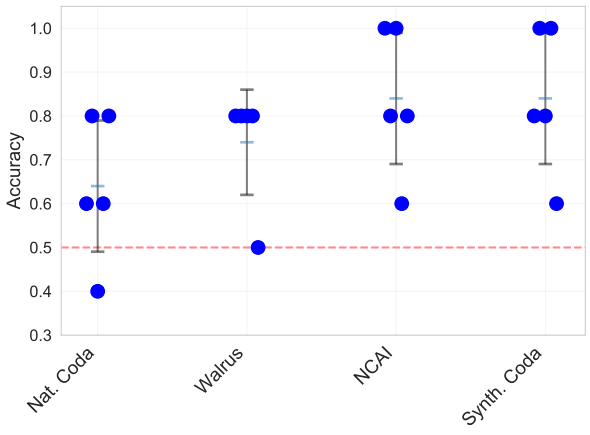


Figure 5.6: Domain-specific accuracy in mixed classification (Task 2). Error bars show standard deviation across experts. Natural codas (left) were misclassified as synthetic 36% of the time. The remaining columns depict performance on synthetic codas generated by WhAM from walrus vocalizations, non-coda acoustic impulses, and codas (respectively). There were five synthetic codas from each domain, plus ten natural codas for a total of 25 items.

Expert backgrounds. Five academic experts participated in the perceptual study. Three identified as marine biologists, and two as underwater acoustics specialists. They listed between 3 and 20 years of experience working with coda audio, specifically field recordings, manual detection and classification, and the development of automatic detection systems. All experts had experience analyzing coda audio and spectrograms, which are the two media through which the experiment was carried out.

Experiment design. We designed a four-task study to be completed sequentially by each expert:

1. **Audio-only two-alternative forced choice (2AFC):** Experts compared pairs of codas (one natural, one synthetic) in audio-only conditions, and were asked to identify the synthetic coda. Synthetic codas were generated by WhAM using the paired natural coda as input.

2. **Mixed Collection Classification:** Experts classified clips as natural or synthetic. Clips were either natural codas, or synthetic codas generated from three different source

domains: natural codas, non-coda acoustic impulses (NCAI),[2] or walrus vocalizations (Sayigh et al., 2016). This task measured false positives (natural misclassified) and false negatives (synthetic undetected).

3. **Spectrogram-assisted 2AFC:** Experts repeated the first task while visualizing audio with software of their choice. The experts were given the exact same samples as in the first task, ensuring direct comparability between audio-only and spectrogram-aided performance. This task mirrored real-world analysis workflows while quantifying the perceptual "advantage" of multimodal inspection.

4. **Qualitative assessment:** Experts were given five representative samples of synthetic codas. They were then asked questions about how well synthetic codas captured / missed characteristics of natural codas, whether any non-natural patterns appeared in synthetic codas, and which features did they use to distinguish between codas in each of the previous tasks.

Fleiss's κ quantified inter-expert agreement (Fleiss, 1971), and accuracy was calculated relative to ground-truth labels. Task order was chosen towards minimizing bias (audio-first to avoid visual priming), with background information collected in a final section.

Quantitative analysis

Experts achieved 81% accuracy ($\kappa = 0.41$), in audio-only 2AFC (Task 1), rising marginally to 83% ($\kappa = 0.41$) with spectrograms visualized (Task 3). This 2% improvement suggests WhAM's synthetic codas lack glaring spectro-temporal artifacts detectable by trained analysts. As expected, accuracy with spectrograms was generally better per-expert, with one expert's performance dramatically increasing from 66% to 93% (another expert even achieved a perfect score). Surprisingly, one expert's performance *decreased* from 83% in Task 1 to 66% in Task 3; comments in the qualitative section did not suggest an explanation.

Performance varied substantially across tasks and among experts (Figure 5.5). The most experienced expert ranked highest in both 2AFC tasks, but not in mixed classification. These variations reflect diverging expert strategies—some focused on inter-click patterns, others on spectral properties: "rhythm" to quote one expert, versus "DC offsets" and "inter-pulse structures" (Møhl et al., 2003) to quote others.

Misclassification rates in Task 2 (Figure 5.6) revealed WhAM's efficacy in acoustic translation: on average, experts correctly flagged walrus-to-coda audio only 75% of the time—less than NCAI or coda-to-coda outputs of WhAM. For one expert, walrus-to-coda audio was detected only 50% of the time (random chance).

Fleiss's κ values (0.41–0.44) indicated moderate agreement across tasks, with experts showing greatest consensus on mixed classification ($\kappa = 0.44$). Performance on spectrogram-aided 2AFC performance was the most diverse—one expert achieved perfect performance while another approached chance (60%).

[2] i.e., an artificial sequence of clicks

Qualitative feedback

Synthetic codas successfully replicated key acoustic features of natural codas. Most experts noted preservation of *rhythm*, referred to as inter-click intervals (ICI); that is, clicks occur "at the right time" in synthetic codas. Additionally, one expert answered that "spectral components" were overall preserved in synthetic codas.

That said, experts identified missing components which can be partitioned into three categories:

- *Within a single click:* Some clicks "came on and disappeared too strongly," had "varying amplitude [within a single coda]," and "inverted peaks." On a spectral level, an expert answered that clicks were too "broadband" compared to natural clicks which have a low-frequency bias.
- *Rhythmic/temporal:* One expert noted that the timing of clicks fit echolocation moreso than codas.[3]
- *Recording-level anomalies:* One expert noted a "DC offset" which they described as the unrealistic background noise on synthetic codas. Similarly, another noted that background noise in synthetic codas oscillated too much.

5.3.3 Utility of embeddings for downstream tasks

We test whether WhAM's internal representations capture meaningful features of sperm whale vocalizations through three downstream classification tasks. For each task, we train a small (two-layer) classifier head that takes coda embeddings as input. We compare WhAM to naive random-embedding and majority-class baselines, as well as AVES (Hagiwara, 2023), a self-supervised model achieving state-of-the-art performance on bioacoustic classification tasks. Full details of the experimental setup are deferred to Section 5.6.6.

The downstream tasks are:

1. *Coda detection*: Given a snippet of audio, determine whether it contains a coda. The classifier is trained on positive (coda) and negative (no coda) snippets, with negative examples drawn from the same recording conditions to ensure the model learns coda features rather than recording artifacts.
2. *Rhythm type*: Given a snippet of audio, classify its temporal pattern. Rhythm of inter-click intervals serves as a key axis for classification of sperm whale codas in cetacean research (Schulz et al., 2011; Bermant et al., 2019; Sharma et al., 2024a).
3. *Social unit classification*: The lowest level of sperm whale social structure are called social units (SU) and have stable, matrilineally-related membership of females and their

[3]Echolocation clicks have consistent inter-click intervals, whereas codas have irregular rhythmic patterns. See Section 5.1.

Table 5.1: **Classification accuracies (%) of different audio embeddings.** For AVES and WhAM, the classifier head is trained with different random seeds, with mean±stderr reported. Random baseline uses randomly initialized AVES (training only the classifier); Majority predicts most common class.

Task	AVES	WhAM	Baseline	
			Rand.	Maj.
Detection	**92.8** ± 0.2	91.3 ± 0.1	60.9	60.9
Rhythm	**90.4**±1.6	84.4±1.6	66.3	60.9
Social Unit	**92.0**±5.6	70.5±0.7	42.5	35.1
Vowel	**91.8**±2.5	79.6±2.9	66.3	66.3

young (Christal, Whitehead, and Lettevall, 1998). Importantly, all SUs in DSWP+CETI belong to the same vocal clan and thus share a common repertoire of coda types, making this more of a speaker identification task than dialect classification.[4]

4. *Vowel type*: Given a coda recording, classify the recently discovered vowel-like features of Beguš et al. (2023).

Table 5.1 shows classification accuracies for each task. While AVES consistently outperforms WhAM, this is expected as AVES is a non-generative model specifically designed for bioacoustic classification tasks, serving more as a performance ceiling than a baseline. Notably, WhAM's representations are useful despite being trained only for generation, outperforming both naive baselines. This suggests that meaningful acoustic features emerge naturally from training for coda generation, even without explicit supervision for these tasks.

We conducted an ablation study to assess how fine-tuning affects embedding quality by evaluating different WhAM variants with specific components removed (detailed in Section 5.4.2). The results reveal that fine-tuning did not significantly alter WhAM's downstream utility compared to base VampNet embeddings, despite WhAM's specialization on whale codas. However, as shown in Section 5.4.3, species-specific fine-tuning was essential for enabling WhAM's core capability of translating audio into sperm whale vocalization acoustics.

[4]By analogy to human language, consider the task of classifying speakers by city of origin. It would be significantly easier to distinguish between speakers from cities that use different dialects; indeed, this is *not* the case in our data.

5.4 Supplementary experiments

5.4.1 FAD Embedding Selection

The Fr'echet Audio Distance (FAD) measures similarity between audio datasets using embeddings to map the audio into a feature space. The choice of embedding is crucial, as different embeddings capture different aspects of the signal. For analyzing sperm whale codas, we sought an embedding that prioritizes the temporal patterns critical to coda structure over background noise. This appendix describes the calibration experiment we conducted to select the most suitable embedding for our FAD analysis.

Let the coda recordings in DSWP+CETI be denoted by $\{x_1, ..., x_n\}$, we:

1. Created denoised versions $\{\hat{x}_1, ..., \hat{x}_n\}$ as detailed in Section 5.6.1
2. Isolated the removed noise components $\{x_1 - \hat{x}_1, ..., x_n - \hat{x}_n\}$
3. For each candidate embedding f_i, compared:
   - d_1^i = FAD score between codas and their denoised versions:
   - d_2^i = FAD score between codas and their noise components:

We evaluated four common audio embeddings VGGish (Gemmeke et al., 2017b; Hershey et al., 2017), Encodec-embd (Défossez et al., 2023), LAION CLAP Music, and LAION CLAP Audio (Wu* et al., 2023; Chen et al., 2022) using the Fréchet Audio Distance implementation of Gui et al. (2024). The ratio d_2^i/d_1^i indicates how much more weight embedding i gives to background noise versus temporal structure. A larger ratio indicates stronger emphasis on temporal patterns and better suitability for the quantitative assessment of audio translation experiment. Table 5.2 shows these ratios for each embedding.

Table 5.2: Comparison of Audio Embeddings for Temporal Structure Sensitivity.

Embedding	d_1 (Coda vs. Denoised)	d_2 (Coda vs. Noise)	d_2/d_1 Ratio
VGGISH	2.0844	1.5027	0.7209
Encodec-embd	25.9716	3.156	0.1215
LAION CLAP Music	0.1483	0.1080	0.7282
LAION CLAP Audio	0.1144	0.1098	**0.9597**

Based on these results, we selected LAION CLAP Audio for our main FAD experiments, as it showed the strongest preference for temporal structure.

5.4.2 Downstream Task Ablation Study

To evaluate the contributions of different components in WhAM, we conduct an ablation study by progressively removing elements and assessing performance across the same set of downstream tasks. The results are presented in Figure 5.7.

No finetuning. We test the effect of skipping domain-adaptation (step (b) in Figure 5.1), or skipping finetuning of VampNet altogether (steps b,c) in Figure 5.1). For all tasks except Social Unit classification, removing species-specific finetuning or domain adaptation does not have a significant impact on the accuracy. This indicates that the inclusion of these steps in WhAM does not significantly degrade the performance on most downstream tasks.

Tokenizer-only. We falsify the hypothesis that the neural audio codec is sufficient for capturing semantic properties in the audio by testing downstream classification directly on the acoustic tokens (Figure 5.3), without embedding them through the MATM. This causes a statistically significant performance drop, particularly in Social Unit classification (-10.9 points, from 70.5% $\pm$ 0.7% to 59.6% $\pm$ 2.0%)

5.4.3 Fréchet Ablation Study

To complement the ablation study of Section 5.4.2, the experiments detailed in Section 5.3.1 were repeated twice with marine mammal sounds. First using the model without **Species-Specific Fine-Tuning** (SSFT, step (c) in Figure 5.1), and then with the **Tokenizer-only** model (as in Section 5.4.2). These results (Figure 5.8) show that, as expected, fine-tuning WhAM on sperm whale data results in outputs that are more similar to sperm whale vocalizations.

5.5 Limitations and future work

The most immediate technical limitation concerns the audio codec architecture. Our current implementation only finetunes the MATM while keeping the codec fixed (see Section 5.2.1). This design choice, while computationally efficient, may limit the model's ability to capture nuanced acoustic features specific to sperm whale vocalizations. For instance, the recently discovered vowel-like features in the 3.7–5.7kHz band (Beguš et al., 2023) may be inadequately represented by a codec primarily trained on human music. Future work could explore either finetuning the entire codec or developing specialized codecs for bioacoustic signals.

Expert feedback (Section 5.3.2) highlighted specific limitations in click generation: unnatural onset and decay patterns, inconsistent background noise, and click properties more reminiscent of echolocation than communication codas. These limitations might be addressed through architectural modifications, such as incorporating adversarial components (Beguš,

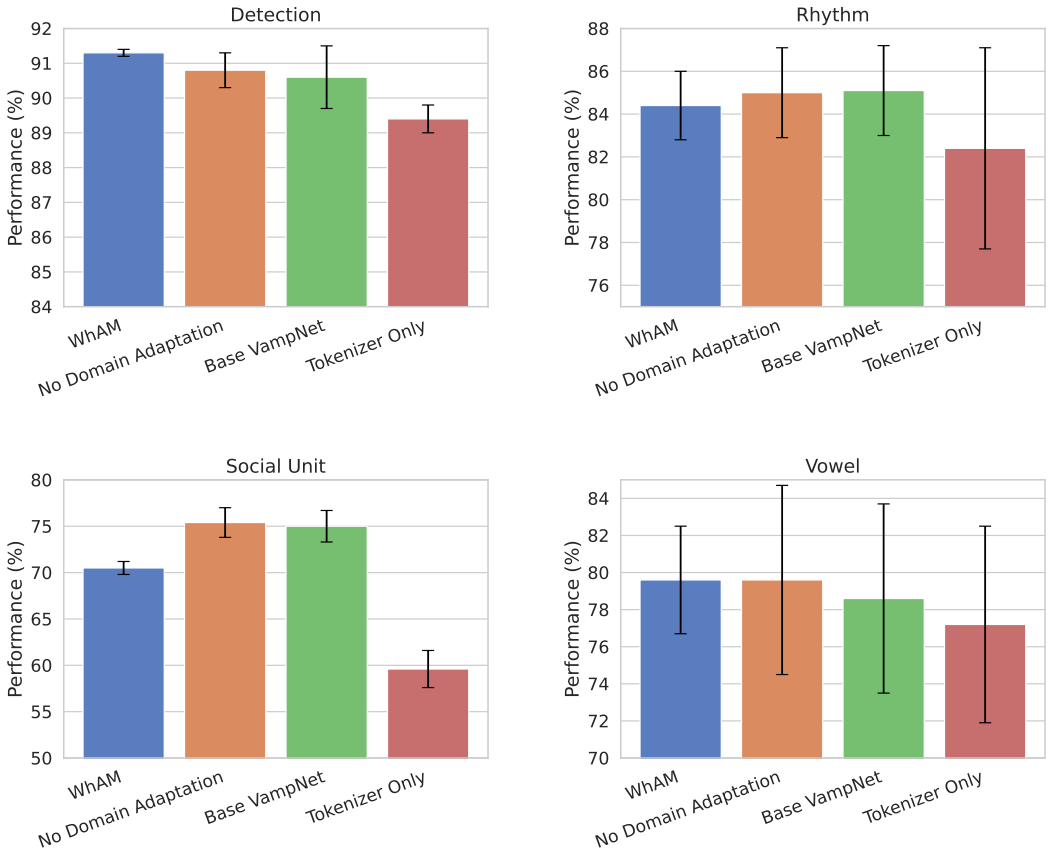


Figure 5.7: Accuracy scores downstream tasks ablation study.

Leban, and Gero, 2023) or introducing specialized modules that leverage domain knowledge about sperm whale click structure. Notably, the observation about echolocation-like properties led to an unexpected finding in our dataset preparation: the presence of echolocation sequences in datasets intended for communication codas. This discovery highlights a broader challenge in bioacoustics research—the difficulty of building clean, well-labeled datasets at scale. Future work should focus on developing robust methods for distinguishing between different types of vocalizations, perhaps by leveraging existing automated detection systems (Bermant et al., 2019).

These data quality challenges underscore the importance of thorough evaluation protocols. Expanding the expert panel would provide more robust perceptual assessments, though we acknowledge the practical challenges in recruiting specialists in sperm whale vocalizations. Additionally, developing more principled evaluation methods—and meta-evaluating these—would help establish standardized benchmarks for bioacoustic generation tasks.

While our results demonstrate impressive performance with relatively small datasets—orders of magnitude smaller than typical in modern acoustic model training—scaling up the

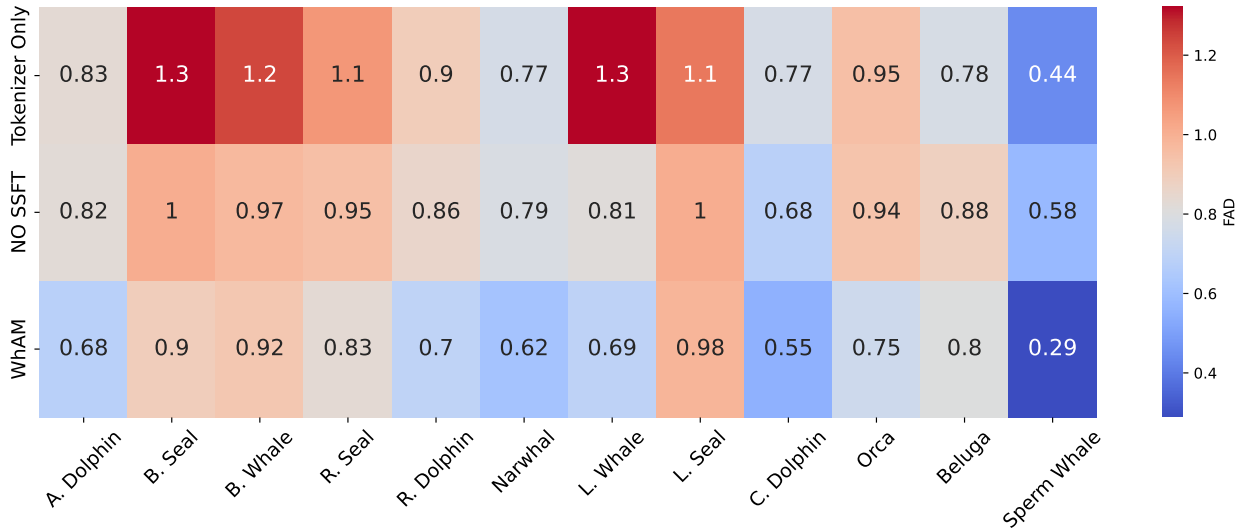


Figure 5.8: Ablation Study FAD Results.

training data could yield substantial improvements. This would require significant effort in aggregating and preprocessing additional sperm whale datasets, as our experience with DSWP+CETI highlighted the technical challenges involved in preparing bioacoustic data for machine learning pipelines.

Looking beyond technical improvements, future work could explore unsupervised learning approaches to uncover new coda features, following the success of similar approaches in bioacoustics (Beguš et al., 2023). This could lead to discoveries about sperm whale communication that complement traditional analytical methods while providing new directions for improving generative models of animal vocalizations.

Our methodological framework—from the two-phase training approach to the expert evaluation protocol—could be adapted for studying other animal communication systems. Our experience suggests that success will require careful attention to species-specific acoustic features and close collaboration with domain experts who can identify subtle but important characteristics of vocalizations.

The gap between generating vocalizations and understanding their meaning remains vast. While WhAM represents the first attempt at acoustic translation in the context of sperm whale communication, future work should explore ways to bridge this semantic gap while maintaining minimal assumptions about the underlying communication system.

5.6 Here be dolphins: full details of the model and experimental setup

5.6.1 Data

FSD. The FSD50k dataset includes 3,159 audio recordings labeled with the "animal" tag, amounting to a total duration of 7 hours and 45 minutes. Noisy segments were retained to preserve real-world variability in training data.

AudioSet. The AudioSet dataset was used to supplement training with additional animal vocalizations. It contains 5h8m hours of audio.

WMMS. The Watkins Marine Mammal Sound Database consists of raw, unlabeled audio recordings. The dataset contains a total of 4 hours and 8 minutes of audio. Each recording was segmented into 10-second snippets for training. No additional denoising was applied. The dataset contained vocalizations from the following mammals (names as listed on the WMMS website):

Atlantic Spotted Dolphin	Bearded Seal	Beluga (White Whale)
Bottlenose Dolphin	Boutu (Amazon River Dolphin)	Bowhead Whale
Clymene Dolphin	Commerson's Dolphin	Common Dolphin
Dall's Porpoise	Dusky Dolphin	False Killer Whale
Fin, Finback Whale	Finless Porpoise	Fraser's Dolphin
Grampus (Risso's Dolphin)	Gray Seal	Gray Whale
Harbor Porpoise	Harbour Seal	Harp Seal
Heaviside's Dolphin	Hooded Seal	Humpback Whale
Irrawaddy Dolphin	Juan Fernandez Fur Seal	Killer Whale
Leopard Seal	Long-Beaked (Pacific) Common Dolphin	Long-Finned Pilot Whale
Melon-Headed Whale	Minke Whale	Narwhal
New Zealand Fur Seal	Northern Right Whale	Pantropical Spotted Dolphin
Ribbon Seal	Ringed Seal	Ross Seal
Rough-Toothed Dolphin	Sea Otter	Short-Finned (Pacific) Pilot Whale
Southern Right Whale	Sperm Whale	Spinner Dolphin

Spotted Seal	Steller Sea Lion	Striped Dolphin
Tucuxi Dolphin	Walrus	Weddell Seal
West Indian Manatee	White-beaked Dolphin	White-sided Dolphin

DSWP. The dataset consists of codas collected between 2005–2018 in a 2000km^2 area off the coast of Dominica. Codas were recorded using various recording systems including far-field boat-based hydrophones and animal-borne tags. Recording setups were as follows:

2005: A Fostex VF-160 multitrack recorder (44.1kHz sampling rate) and a custom built towed hydrophone (Benthos AQ-4 elements, frequency response: 0.1–30kHz) with a filter box with high-pass filters up to 1 kHz resulting in a recording chain with a flat frequency response across a minimum of 2–20kHz.

2006: No recordings during this short season.

2007,2009,2011: A Zoom H4 portable field recorder (48kHz sampling rate) and a Cetacean Research Technology C55 hydrophone (frequency response: 0.02–44kHz) and no filters.

2008,2010,2012,2015: A custom-built towed hydrophone (Benthos AQ-4 elements, frequency response: 0.1–30kHz) with a filter box with high-pass filters up to 1 kHz resulting in a recording chain with a flat frequency response across a minimum of 2–20 kHz. This was connected to a computer based recording system as a part of the International Fund for Animal Welfare's (IFAW) LOGGER software package (48kHz sampling rate) or PAMGUARD (minimum 48 kHz sampling rate). In addition, recordings were also made through the deployment of animal-borne sound and movement tags (DTag generation 3, Johnson and Tyack 2003).

CETI. All systems were sampling above 96kHz with a 16bit resolution with a minimum flat (±2dB) frequency response within 1–45kHz.

The DSWP and CETI dataset contain background noise such as water sounds. To improve model performance, we denoise datasets before training on the model. A noise profile of each recording in the frequency domain was generated by sampling sections which did not contain codas. Then, we perform spectral subtraction to remove noise in the frequency domain, and transform back to the time domain of the audio signal.

All audio samples were downsampled to 16 kHz and normalized to have zero mean and unit variance when passed into VampNet.

5.6.2 Model Training

The model training procedure consisted of two phases: domain adaptation and species-specific fine-tuning.

Table 5.3: Quantitative Assessment Data Summary.

Full Name	Shortened Name	Num. Samples
Atlantic Dolphin	A. Dolphin	58
Bearded Seal	B. Seal	37
Bowhead Whale	B. Whale	60
Beluga Whale, White Whale	Beluga	50
Walrus	Walrus	38
Clymene Dolphin	C. Dolphin	63
Narwhal	Narwhal	50
Leopard Seal	L. Seal	10
Long-finned Whale	L. Whale	10
Killer Whale (Orca)	Orca	35
Ross Seal	Ross Seal	50
Risso's Dolphin	Risso	67

Acoustic Tokenizer Settings. Discrete token vocabulary size (Σ) = 1024. Frequency of Input Audio N_{sam} = 16kHz. Tokenizer input length $N_{\text{sec}} = 10$.

Domain Adaptation. In the first phase, the model was pretrained on a mixture of general animal vocalizations, including data from FSD and AudioSet. This step aimed to establish a broad understanding of bioacoustic patterns. The model was trained for 500,000 iterations using the AdamW optimizer with a learning rate of 0.0001. A batch size of 6 was used, and gradient clipping was applied to stabilize training. The model took 123 hours to train using an NVIDIA A10 GPU.

Species-Specific Fine-Tuning. Following domain adaptation, the model was fine-tuned on whale-specific data from DSWP+CETI to adapt its representations to sperm whale vocalizations. The fine-tuning process used the same optimizer and learning rate as the pretraining phase and a batch size of 6. Training continued for another 500,000 iterations. This took 39 hours to run using an NVIDIA A10 GPU.

5.6.3 Generating data for Sections 5.3.1 and 5.3.2

Three different input sources were used to generate samples for both the **Quantitative Assessment of Audio Translation** and the **Expert Perceptual Evaluation**. The prompt settings for each input type are summarized in Table 5.4

Table 5.4: Prompt settings for each input type.

Input	Periodic Prompt	Onset Mask Width	Num. of Steps	Typical Mass	Sample Cutoff
Codas	12	21	50	0.102	0.17
NCAI	12	21	50	0.102	0.17
A. Dolphin	16	5	74	0.15	0.39
B. Seal	7	1	70	0.15	0.44
B. Whale	7	1	70	0.15	0.44
Beluga	13	13	85	0.15	0.39
Walrus	18	1	107	0.15	0.33
C.Dolphine	12	14	72	0.15	0.25
Narwhal	6	4	39	0.15	0.21
L. Seal	6	4	46	0.15	0.39
L. Whale	15	19	57	0.15	0.42
Orca	13	2	46	0.15	0.39
Ross Seal	18	3	66	0.15	0.49
Risso	13	13	85	0.15	0.39

Watkins Marine Mammals. Eleven species were selected from the "Best of Watkins Marine Mammals" dataset. Due to variations in vocalization characteristics and recording conditions, prompt settings were manually optimized for each species. These species and prompt settings can be found in Table 5.4.

Non-Coda Acoustic Impulses (NCAI). Five NCAI sequences were generated. Each snippet was initialized as a zero-filled array at a 44.1 kHz sample rate. Clicks were simulated by selecting random indices and setting them to a peak amplitude of 1. To ensure realistic timing and rhythm, real coda sequences were prepended to each generated sample before synthesis. These prepended codas were then removed after generation.

5.6.4 Quantitative Assessment of Audio Translation

For each input type listed in Table 5.4, samples were generated using the specified prompt configurations. The **Fréchet Audio Distance (FAD)** was computed between the generated samples and real codas using LAION CLAP Audio embeddings.

5.6.5 Expert Perceptual Evaluation

Five domain experts in sperm whale bioacoustics participated in the evaluation. Given the highly specialized nature of sperm whale vocalization analysis, the pool of qualified experts

with years of direct experience analyzing and annotating these vocalizations is notably small. All participants were recruited from an established research collaboration studying cetacean communication, and each had at least three years of experience working with sperm whale codas.

The evaluation was conducted via Google Form. The form began with the following introduction:

Welcome

Thank you for participating in this study. Your expertise in analyzing sperm whale vocalizations is invaluable for evaluating our model.
The study consists of four parts, to be completed in order. A final section includes three short questions about your background.
Technical Setup

- Download and extract the `listener_evaluation.zip` file from a provided link
- Use headphones for all listening tasks
- Complete the experiment in a quiet environment
- You can take breaks between sections as needed

If you encounter any technical difficulties or have questions about the procedure, please contact [omitted].

Participant Identification
Name (used for tracking responses only): _____

Audio-Only Two-Alternative Forced Choice (2AFC)

Listeners were presented with 30 pairs of codas. Each pair contained an original, denoised coda and a model-generated counterpart. Participants were asked to identify which sample was the original and which was generated.

Task Instructions

In this section, you will listen to pairs of codas. For each pair, one is a natural recording and one is synthetic. Please indicate which one you believe is synthetic.
The audio files are located in the ***section1/*** folder. Each pair consists of two files:

- *1a.wav* + *1b.wav*
- *2a.wav* + *2b.wav*
- etc.

Please listen to each file **at most three times**. Base your decision only on the provided audio. Do not visualize the audio.

Mixed Two-Alternative Forced Choice (2AFC)

Listeners were presented with 25 individual samples: 10 real codas, 5 generated from real codas, 5 generated from walrus vocalizations, and 5 generated from Non-coda Acoustic Impulses (NCAI). Each listener classified each sample as either real or generated.

Task Instructions

In this section, you will listen to individual codas and classify each as either natural or synthetic.
The audio files are located in the ***section2*** folder:

- *1.wav*
- *2.wav*
- etc.

Please listen to each file at most three times. **Base your decision only on the provided audio. Do not visualize the audio.**

Visualized Two-Alternative Forced Choice (2AFC)

This experiment was identical to the **Audio-Only 2AFC** condition, except participants were allowed to inspect the spectrograms of each recording using their preferred software before making their decision. Marine biologists preferred Adobe Auditions, while underwater acoustics experts used Matlab.

Task Instructions

Once again, you will listen to pairs of codas (a.wav and b.wav). For each pair, one is a natural recording and one is synthetic. Please indicate which one you believe is synthetic.
The audio files are located in the ***section3/*** folder. Each pair consists of two files:

- *1a.wav* + *1b.wav*
- *2a.wav* + *2b.wav*
- etc.

Please listen to each file at most three times. **You may now visualize the audio using any software you are familiar with.**
What software will you use to visualize the audio? _____

Qualitative Assessment

Task Instructions

For this final section, please first listen to the reference synthetic codas provided in the section4 folder. These examples were chosen to represent typical outputs of our model. Then, based on these examples and your experience with all parts of the experiment, please answer the following questions
What characteristics of natural codas are well represented in the synthetic ones?
What characteristics of natural codas are missing or different in the synthetic ones?
Did you observe any patterns in the synthetic codas that do not occur in natural ones?
When **only listening** to the audio (sections 1 and 2), what helped you distinguish between natural and synthetic codas?
When **visualizing** the audio pairs (section 3), what helped you distinguish between natural and synthetic codas?

Background Information

Task Instructions

To help contextualize the evaluations, please tell us about your experience working with sperm whale codas.
How many years have you spent professionally analyzing sperm whale codas (e.g., in research, conservation, or educational contexts)?
What types of coda work have you performed?

- *Recording of codas in the field*
- *Development of recording methods for codas*
- *Manual detection, classification or annotation of codas*
- *Development of automatic detection, classification or annotation systems*
- *Meta-analysis (e.g. methodology development, literature review)*
- *Other...*

In what contexts have you worked with coda recordings?

- *Academic research*
- *Conservation work*
- *Industry/commercial projects*
- *Educational/training contexts*
- *Government/regulatory work*

What is your primary field of expertise?

5.6.6 Utility of Embeddings for Downstream Tasks

Model Details. We run a forward pass through WhAM and AVES to obtain embeddings from the audio. Both WhAM and AVES output varying embeddings over time, so we average the embeddings over time to obtain 1 unified embedding for 1 audio snippet. After the embedding is obtained, we attach a two-layer feed-forward neural network as a classifier. The network consists of a fully connected layer that projects the embedding into a 128-dimensional hidden layer, followed by a ReLU activation. A second fully connected layer then generates class probabilities.

We evaluate embeddings from WhAM and AVES, comparing their performance against

Table 5.5: Dataset sizes for downstream classification tasks.

Task	Number of Samples
Coda Detection	3,100
Rhythm Type Classification	916
Social Unit Classification	2,659
Vowel Classification	486

a random embedding baseline as well as a majority baseline classifier.

Training Data. For downstream task evaluation, we leveraged annotations in the DSWP+CETI datasets. Using human-annotated timestamps, we identified and extracted audio segments containing codas, each spanning 1–2 seconds. Each coda was labeled for one of the following classification tasks:

- **Coda Detection**: Determine whether a given audio snippet contains a whale coda.
- **Rhythm Type Classification**: Classify codas according to their rhythmic patterns. For this task, we choose to include samples whose rhythm types are among the 5 most common, because the remaining ones appear too infrequently for classifiers to be accurate.
- **Social Unit Classification**: Identify the social unit associated with each coda.
- **Vowel Classification**: Detect vowel-like elements within whale vocalizations.

Table 5.5 summarizes dataset sizes for each task.

Training Process. We split the dataset into 80% training and 20% testing, using stratified sampling of labels to ensure consistent label distribution. The embedding model is frozen, and only the classifier parameters are trained. Training is performed on an NVIDIA A10G GPU for 10 epochs, using a learning rate of 10^{-4} and a batch size of 32. Model checkpoints are saved at each epoch, and the best-performing model is selected based on test set performance.

Bibliography

Agarwal, Alekh, Sham M. Kakade, Jason D. Lee, and Gaurav Mahajan (2021). "On the Theory of Policy Gradient Methods: Optimality, Approximation, and Distribution Shift". In: *J. Mach. Learn. Res.* 22, 98:1–98:76. URL: http://jmlr.org/papers/v22/19-736.html.

Agostinelli, Andrea, Timo I. Denk, Zalán Borsos, Jesse H. Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, Matthew Sharifi, Neil Zeghidour, and Christian Havnø Frank (2023). "MusicLM: Generating Music From Text". In: *CoRR* abs/2301.11325. DOI: 10.48550/ARXIV.2301.11325. arXiv: 2301.11325. URL: https://doi.org/10.48550/arXiv.2301.11325.

Ahyong, S., C.B. Boyko, N. Bailly, J. Bernot, R. Bieler, S.N. Brandão, M. Daly, S. De Grave, S. Gofas, F. Hernandez, L. Hughes, T.A. Neubauer, G. Paulay, W. Decock, S. Dekeyzer, L. Vandepitte, B. Vanhoorne, R. Adlard, S. Agatha, K.J. Ahn, N. Akkari, B. Alvarez, V. Amorim, A. Anderberg, G. Anderson, S. Andrés Sánchez, Y. Ang, D. Antic, L.S.. Antonietto, C. Arango, T. Artois, S. Atkinson, K. Auffenberg, B.G. Baldwin, R. Bank, A. Barber, J.P. Barbosa, I. Bartsch, D. Bellan-Santini, N. Bergh, A. Berta, T.N. Bezerra, S. Blanco, I. Blasco-Costa, ..., and A. Zullini (2022). *World Register of Marine Species (WoRMS)*. =https://www.marinespecies.org. Accessed: 2022-10-22. URL: https://www.marinespecies.org.

Alman, Josh and Lijie Chen (2022). "Efficient construction of rigid matrices using an NP oracle". In: *SIAM J. Comput.* 0.0, FOCS19–102. URL: https://doi.org/10.1137/20M1322297.

Alman, Josh and R. Ryan Williams (2017). "Probabilistic rank and matrix rigidity". In: *Proc. 49th ACM Symp. on Theory of Computing (STOC)*, pp. 641–652. eprint: 1611.05558. URL: https://doi.org/10.1145/3055399.3055484.

Alon, Noga, Oded Goldreich, Johan Håstad, and René Peralta (1992). "Simple Constructions of Almost $k-$wise Independent Random Variables". In: *Random Structures Algorithms* 3.3. (Preliminary version in *31st FOCS*, 1990), pp. 289–304. URL: https://doi.org/10.1002/rsa.3240030308.

Amit, Noga, Shafi Goldwasser, Orr Paradise, and Guy N. Rothblum (2024). "Models That Prove Their Own Correctness". In: *CoRR* abs/2405.15722. DOI: 10.48550/ARXIV.2405.15722. arXiv: 2405.15722. URL: https://doi.org/10.48550/arXiv.2405.15722.

Amphaeris, Jenny, Daniel T Blumstein, Graeme Shannon, Thora Tenbrink, and Arik Kershenbaum (2023). "A multifaceted framework to establish the presence of meaning in non-human communication". In: *Biological Reviews* 98.6, pp. 1887–1909.

Amphaeris, Jenny, Graeme Shannon, and Thora Tenbrink (2022). "Overlap not gap: Understanding the relationship between animal communication and language with Prototype Theory". In: *Lingua* 272, p. 103332. ISSN: 0024-3841. DOI: https://doi.org/10.1016/j.lingua.2022.103332. URL: https://www.sciencedirect.com/science/article/pii/S0024384122000936.

Andreas, Jacob, Gašper Beguš, Michael M Bronstein, Roee Diamant, Denley Delaney, Shane Gero, Shafi Goldwasser, David F Gruber, Sarah de Haas, Peter Malkin, et al. (2022a). "Toward understanding the communication in sperm whales". In: *Iscience* 25.6.

Andreas, Jacob, Gašper Beguš, Michael M. Bronstein, Roee Diamant, Denley Delaney, Shane Gero, Shafi Goldwasser, David F. Gruber, Sarah de Haas, Peter Malkin, Nikolay Pavlov, Roger Payne, Giovanni Petri, Daniela Rus, Pratyusha Sharma, Dan Tchernov, Pernille Tønnesen, Antonio Torralba, Daniel Vogt, and Robert J. Wood (2022b). "Toward understanding the communication in sperm whales". In: *iScience* 25.6, p. 104393. ISSN: 2589-0042. DOI: https://doi.org/10.1016/j.isci.2022.104393. URL: https://www.sciencedirect.com/science/article/pii/S2589004222006642.

Anil, Cem, Guodong Zhang, Yuhuai Wu, and Roger B. Grosse (2021). "Learning to Give Checkable Answers with Prover-Verifier Games". In: *CoRR* abs/2108.12099. arXiv: 2108.12099. URL: https://arxiv.org/abs/2108.12099.

Anthes, Emily (Aug. 2022). "The animal translators". In: *The New York Times*. URL: https://www.nytimes.com/2022/08/30/science/translators-animals-naked-mole-rats.html.

Antunes, Ricardo, Tyler Schulz, Shane Gero, Hal Whitehead, Jonathan Gordon, and Luke Rendell (2011). "Individually distinctive acoustic features in sperm whale codas". In: *Animal Behaviour* 81.4, pp. 723–730. ISSN: 0003-3472. DOI: https://doi.org/10.1016/j.anbehav.2010.12.019. URL: https://www.sciencedirect.com/science/article/pii/S0003347210005233.

Arora, Sanjeev and Boaz Barak (2009). *Computational Complexity: A Modern Approach*. Cambridge University Press. URL: https://doi.org/10.1017/CBO9780511804090.

Arora, Sanjeev, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy (May 1998). "Proof Verification and the Hardness of Approximation Problems". In: *J. ACM* 45.3. (Preliminary version in *33rd FOCS*, 1992), pp. 501–555. URL: https://doi.org/10.1145/278298.278306.

Arora, Sanjeev and Shmuel Safra (Jan. 1998). "Probabilistic Checking of Proofs: A New Characterization of NP". In: *J. ACM* 45.1. (Preliminary version in *33rd FOCS*, 1992), pp. 70–122. URL: https://doi.org/10.1145/273865.273901.

Artetxe, Mikel, Gorka Labaka, and Eneko Agirre (July 2018). "A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long*

Papers). Melbourne, Australia: Association for Computational Linguistics, pp. 789–798. DOI: 10.18653/v1/P18-1073. URL: https://aclanthology.org/P18-1073.

Artetxe, Mikel, Gorka Labaka, and Eneko Agirre (2019). "Unsupervised Neural Machine Translation, a new paradigm solely based on monolingual text". In: *Proces. del Leng. Natural* 63, pp. 151–154. URL: http://journal.sepln.org/sepln/ojs/ojs/index.php/pln/article/view/6107.

Austrin, Per, Jonah Brown-Cohen, and Johan Håstad (2021). "Optimal Inapproximability with Universal Factor Graphs". In: *Proc. 32nd Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pp. 434–453. URL: https://doi.org/10.1137/1.9781611976465.27.

Babai, László, Paul Erdo˝s, and Stanley M. Selkow (1980). "Random Graph Isomorphism". In: *SIAM Journal on Computing* 9.3, pp. 628–635. DOI: 10.1137/0209047. eprint: https://doi.org/10.1137/0209047. URL: https://doi.org/10.1137/0209047.

Babai, László, Lance Fortnow, Leonid A. Levin, and Mario Szegedy (1991). "Checking Computations in Polylogarithmic Time". In: *Proc. 23rd ACM Symp. on Theory of Computing (STOC)*, pp. 21–31. URL: https://doi.org/10.1145/103418.103428.

Backus, Richard H and William E Schevill (1966). "Physeter clicks". In: *Whales, dolphins and porpoises* 510, p. 527.

Barak, Boaz and Oded Goldreich (2008). "Universal Arguments and their Applications". In: *SIAM J. Comput.* 38.5. (Preliminary version in *17th Comput. Complexity Conf.*, 2002), pp. 1661–1694. URL: https://doi.org/10.1137/070709244.

Bartolo, Max, Alastair Roberts, Johannes Welbl, Sebastian Riedel, and Pontus Stenetorp (2020). "Beat the AI: Investigating Adversarial Human Annotation for Reading Comprehension". In: *Trans. Assoc. Comput. Linguistics* 8, pp. 662–678. DOI: 10.1162/tacl_a_00338. URL: https://doi.org/10.1162/tacl_a_00338.

Bartolo, Max, Tristan Thrush, Sebastian Riedel, Pontus Stenetorp, Robin Jia, and Douwe Kiela (2022). "Models in the Loop: Aiding Crowdworkers with Generative Annotation Assistants". In: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*. Ed. by Marine Carpuat, Marie-Catherine de Marneffe, and Iván Vladimir Meza Ruíz. Association for Computational Linguistics, pp. 3754–3767. DOI: 10.18653/V1/2022.NAACL-MAIN.275. URL: https://doi.org/10.18653/v1/2022.naacl-main.275.

Bartusek, James, Thiago Bergamaschi, Seri Khoury, Saachi Mutreja, and Orr Paradise (2024). "On the Communication Complexity of Secure Multi-Party Computation With Aborts". In: *Proceedings of the 43rd ACM Symposium on Principles of Distributed Computing, PODC 2024, Nantes, France, June 17-21, 2024*. Ed. by Ran Gelles, Dennis Olivetti, and Petr Kuznetsov. ACM, pp. 480–491. DOI: 10.1145/3662158.3662815. URL: https://doi.org/10.1145/3662158.3662815.

Baziotis, Christos, Barry Haddow, and Alexandra Birch (2020). "Language Model Prior for Low-Resource Neural Machine Translation". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-*

20, 2020. Ed. by Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu. Association for Computational Linguistics, pp. 7622–7634. DOI: 10.18653/v1/2020.emnlp-main.615. URL: https://doi.org/10.18653/v1/2020.emnlp-main.615.

Bedny, Marina, Jorie Koster-Hale, Giulia Elli, Lindsay Yazzolino, and Rebecca Saxe (2019). "There's more to "sparkle" than meets the eye: Knowledge of vision and light verbs among congenitally blind and sighted individuals". In: *Cognition* 189, pp. 105–115. ISSN: 0010-0277. DOI: https://doi.org/10.1016/j.cognition.2019.03.017. URL: https://www.sciencedirect.com/science/article/pii/S0010027719300721.

Beguš, Gašper, Andrej Leban, and Shane Gero (2023). "Approaching an unknown communication system by latent space exploration and causal inference". In: *arXiv preprint arXiv:2303.10931*.

Beguš, Gašper, Ronald Sprouse, Andrej Leban, Miles Silva, and Shane Gero (Dec. 2023). *Vowels and Diphthongs in Sperm Whales*. DOI: 10.31219/osf.io/285cs. URL: osf.io/285cs.

Bellare, Mihir, Shafi Goldwasser, Carsten Lund, and Alexander Russell (1994). "Efficient probabilistic checkable proofs and applications to approximation". In: *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*. Ed. by Frank Thomson Leighton and Michael T. Goodrich. ACM, p. 820. DOI: 10.1145/195058.195467. URL: https://doi.org/10.1145/195058.195467.

Ben-Sasson, Eli, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan (2005). "Short PCPs verifiable in polylogarithmic time". In: *Proc. 20th IEEE Conf. on Comput. Complexity*. Full version available at http://www.tcs.tifr.res.in/~prahladh/papers/BGHSV2/BGHSV2005.pdf, pp. 120–134. URL: https://doi.org/10.1109/CCC.2005.27.

— (2006). "Robust PCPs of proximity, shorter PCPs and applications to coding". In: *SIAM J. Comput.* 36.4. (Preliminary version in *36th STOC*, 2004), pp. 889–974. URL: https://doi.org/10.1137/S0097539705446810.

Ben-Sasson, Eli and Madhu Sudan (2008). "Short PCPs with Polylog Query Complexity". In: *SIAM J. Comput.* 38.2. (Preliminary version in *37th STOC*, 2005), pp. 551–607. URL: https://doi.org/10.1137/050646445.

Ben-Sasson, Eli, Madhu Sudan, Salil Vadhan, and Avi Wigderson (2003). "Randomness-efficient Low Degree Tests and short PCPs via epsilon-biased sets". In: *Proc. 35th ACM Symp. on Theory of Computing (STOC)*, pp. 612–621. URL: https://doi.org/10.1145/780542.780631.

Ben-Sasson, Eli and Emanuele Viola (2014). "Short PCPs with Projection Queries". In: *Proc. 42nd International Colloq. of Automata, Languages and Programming (ICALP), Part I*. Ed. by Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias. Vol. 8572. LNCS. Springer, pp. 163–173. URL: https://doi.org/10.1007/978-3-662-43948-7_14.

Bergler, Christian, Alexander Barnhill, Dominik Perrin, Manuel Schmitt, Andreas K. Maier, and Elmar Nöth (2022). "ORCA-WHISPER: An Automatic Killer Whale Sound Type

Generation Toolkit Using Deep Learning". In: *23rd Annual Conference of the International Speech Communication Association, Interspeech 2022, Incheon, Korea, September 18-22, 2022*. Ed. by Hanseok Ko and John H. L. Hansen. ISCA, pp. 2413–2417. DOI: 10.21437/INTERSPEECH.2022-846. URL: https://doi.org/10.21437/Interspeech.2022-846.

Bergler, Christian, Hendrik Schröter, Rachael Xi Cheng, Volker Barth, Michael Weber, Elmar Nöth, Heribert Hofer, and Andreas Maier (2019). "ORCA-SPOT: An automatic killer whale sound detection toolkit using deep learning". In: *Scientific reports* 9.1, p. 10997.

Bermant, Peter C, Michael M Bronstein, Robert J Wood, Shane Gero, and David F Gruber (2019). "Deep machine learning techniques for the detection and classification of sperm whale bioacoustics". In: *Scientific reports* 9.1, p. 12588.

Berthet, Mélissa, Camille Coye, Guillaume Dezecache, and Jeremy Kuhn (2022). "Animal linguistics: a primer". In: *Biological Reviews* n/a.n/a. DOI: https://doi.org/10.1111/brv.12897. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/brv.12897. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/brv.12897.

Berwick, Robert C and Noam Chomsky (2016). *Why only us: Language and evolution*. Cambridge, MA: MIT Press.

Best, Peter B (1979). "Social organization in sperm whales, Physeter macrocephalus". In: *Behavior of marine animals: Current perspectives in research*. Springer, pp. 227–289.

Bezout, E. (1779). *Theorie Generale Des Equations Algebriques*. Kessinger Publishing. ISBN: 9781162056128. URL: https://books.google.co.il/books?id=wQZvSwAACAAJ.

Bhangale, Amey, Prahladh Harsha, Orr Paradise, and Avishay Tal (2020). "Rigid matrices from rectangular PCPs or Hard Claims have Complex Proofs". In: *Proc. 61st IEEE Symp. on Foundations of Comp. Science (FOCS)*, pp. 858–869. eprint: 2005.03123. URL: https://doi.org/10.1109/FOCS46700.2020.00084.

— (2024). "Rigid Matrices from Rectangular PCPs". In: *SIAM J. Comput.* 53.2, pp. 480–523. DOI: 10.1137/22M1495597. URL: https://doi.org/10.1137/22m1495597.

Bhatia, Rhythm and Tomi H. Kinnunen (2022). "An Initial Study on Birdsong Re-synthesis Using Neural Vocoders". In: *Speech and Computer - 24th International Conference, SPECOM 2022, Gurugram, India, November 14-16, 2022, Proceedings*. Ed. by S. R. Mahadeva Prasanna, Alexey Karpov, K. Samudravijaya, and Shyam S. Agrawal. Vol. 13721. Lecture Notes in Computer Science. Springer, pp. 64–74. DOI: 10.1007/978-3-031-20980-2_7. URL: https://doi.org/10.1007/978-3-031-20980-2_7.

Blum, Manuel, Michael Luby, and Ronitt Rubinfeld (Dec. 1993). "Self-Testing/Correcting with Applications to Numerical Problems". In: *J. Comput. Syst. Sci.* 47.3. (Preliminary version in *22nd STOC*, 1990), pp. 549–595. URL: https://doi.org/10.1016/0022-0000(93)90044-W.

Blum, Manuel and Silvio Micali (1984). "How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits". In: *SIAM J. Comput.* 13.4, pp. 850–864. DOI: 10.1137/0213053. URL: https://doi.org/10.1137/0213053.

Bommasani, Rishi, Drew A. Hudson, Ehsan Adeli, Russ B. Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill,

Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen Creel, Jared Quincy Davis, Dorottya Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, and et al. (2021). "On the Opportunities and Risks of Foundation Models". In: *CoRR* abs/2108.07258. arXiv: 2108.07258. URL: https://arxiv.org/abs/2108.07258.

Borsos, Zalán, Raphaël Marinier, Damien Vincent, Eugene Kharitonov, Olivier Pietquin, Matthew Sharifi, Dominik Roblek, Olivier Teboul, David Grangier, Marco Tagliasacchi, and Neil Zeghidour (2023). "AudioLM: A Language Modeling Approach to Audio Generation". In: *IEEE ACM Trans. Audio Speech Lang. Process.* 31, pp. 2523–2533. DOI: 10.1109/TASLP.2023.3288409. URL: https://doi.org/10.1109/TASLP.2023.3288409.

Brants, Thorsten, Ashok C. Popat, Peng Xu, Franz Josef Och, and Jeffrey Dean (2007). "Large Language Models in Machine Translation". In: *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic*. Ed. by Jason Eisner. ACL, pp. 858–867. URL: https://aclanthology.org/D07-1090/.

Brown, Tom, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, ..., and Dario Amodei (2020). "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., pp. 1877–1901. URL: https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Brown-Cohen, Jonah, Geoffrey Irving, and Georgios Piliouras (2023). "Scalable AI Safety via Doubly-Efficient Debate". In: *CoRR* abs/2311.14125. DOI: 10.48550/ARXIV.2311.14125. arXiv: 2311.14125. URL: https://doi.org/10.48550/arXiv.2311.14125.

Bubeck, Sébastien, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott M. Lundberg, Harsha Nori, Hamid Palangi, Marco Túlio Ribeiro, and Yi Zhang (2023). "Sparks of Artificial General Intelligence: Early experiments with GPT-4". In: *CoRR* abs/2303.12712. DOI: 10.48550/arXiv.2303.12712. arXiv: 2303.12712. URL: https://doi.org/10.48550/arXiv.2303.12712.

Burnell, Ryan, Wout Schellaert, John Burden, Tomer D Ullman, Fernando Martinez-Plumed, Joshua B Tenenbaum, Danaja Rutar, Lucy G Cheke, Jascha Sohl-Dickstein, Melanie Mitchell, et al. (2023). "Rethink reporting of evaluation results in AI". In: *Science* 380.6641, pp. 136–138.

Campagna, Claudio and Daniel Guevara (2022). ""Save the whales" for their natural goodness". In: *Marine Mammals: The Evolving Human Factor*. Springer, pp. 397–424.

Cantor, Maurício and Hal Whitehead (2015). "How does social behavior differ among sperm whale clans?" In: *Marine Mammal Science* 31.4, pp. 1275–1290.

Carlile, Paul R (2002). "A pragmatic view of knowledge and boundaries: Boundary objects in new product development". In: *Organization science* 13.4, pp. 442–455.

Carroll, Micah, Orr Paradise, Jessy Lin, Raluca Georgescu, Mingfei Sun, David Bignell, Stephanie Milani, Katja Hofmann, Matthew J. Hausknecht, Anca D. Dragan, and Sam Devlin (2022). "Uni[MASK]: Unified Inference in Sequential Decision Problems". In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*. Ed. by Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh. URL: http://papers.nips.cc/paper_files/paper/2022/hash/e58fa6a7b431e634e0fd125e225ad10c-Abstract-Conference.html.

Chan, Timothy M. and R. Ryan Williams (2021). "Deterministic APSP, Orthogonal Vectors, and More: Quickly Derandomizing Razborov-Smolensky". In: *ACM Trans. Algorithms* 17.1. (Preliminary version in *27th SODA*, 2016), 2:1–2:14. URL: https://doi.org/10.1145/3402926.

Chang, Huiwen, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman (2022). "MaskGIT: Masked Generative Image Transformer". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*. IEEE, pp. 11305–11315. DOI: 10.1109/CVPR52688.2022.01103. URL: https://doi.org/10.1109/CVPR52688.2022.01103.

Charton, François (2022). "Linear algebra with transformers". In: *Trans. Mach. Learn. Res.* 2022. URL: https://openreview.net/forum?id=Hp4g7FAXXG.

— (2024). "Can transformers learn the greatest common divisor?" In: *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 6-11, 2024*. OpenReview.net.

Chen, Guobin, Tony X. Han, Zhihai He, Roland Kays, and Tavis Forrester (2014). "Deep convolutional neural network based species recognition for wild animal monitoring". In: *2014 IEEE International Conference on Image Processing, ICIP 2014, Paris, France, October 27-30, 2014*. IEEE, pp. 858–862. DOI: 10.1109/ICIP.2014.7025172. URL: https://doi.org/10.1109/ICIP.2014.7025172.

Chen, Ke, Xingjian Du, Bilei Zhu, Zejun Ma, Taylor Berg-Kirkpatrick, and Shlomo Dubnov (2022). "HTS-AT: A Hierarchical Token-Semantic Audio Transformer for Sound Classification and Detection". In: *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*.

Chen, Lijie, Xin Lyu, and R. Ryan Williams (2020). "Almost Everywhere Circuit Lower Bounds from Non-Trivial Derandomization". In: *Proc. 61st IEEE Symp. on Foundations of Comp. Science (FOCS)*, pp. 1–12. URL: https://doi.org/10.1109/FOCS46700.2020.00009.

Chen, Lijie and Xin Lyu (2021). "Inverse-exponential correlation bounds and extremely rigid matrices from a new derandomized XOR lemma". In: *Proc. 53rd ACM Symp. on Theory of Computing (STOC)*, pp. 761–771. URL: https://doi.org/10.1145/3406325.3451132.

Chen, Lijie and R. Ryan Williams (2019). "Stronger Connections Between Circuit Analysis and Circuit Lower Bounds, via PCPs of Proximity". In: *Proc. 34th Comput. Complexity Conf.* Vol. 137. LIPIcs. Schloss Dagstuhl, 19:1–19:43. URL: https://doi.org/10.4230/LIPIcs.CCC.2019.19.

Chen, Lili, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch (2021). "Decision Transformer: Reinforcement Learning via Sequence Modeling". In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*. Ed. by Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, pp. 15084–15097.

Chollet, François (2019). "On the Measure of Intelligence". In: *CoRR* abs/1911.01547. arXiv: 1911.01547. URL: http://arxiv.org/abs/1911.01547.

Chowdhery, Aakanksha, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, ..., and Noah Fiedel (2022). "PaLM: Scaling Language Modeling with Pathways". In: *arXiv preprint arXiv:2204.02311*. arXiv: 2204.02311 [cs.CL]. URL: https://arxiv.org/abs/2204.02311.

Christal, Jenny and Hal Whitehead (2001). "Social affiliations within sperm whale (Physeter macrocephalus) groups". In: *Ethology* 107.4, pp. 323–340.

Christal, Jenny, Hal Whitehead, and Erland Lettevall (1998). "Sperm whale social units: variation and change". In: *Canadian Journal of Zoology* 76.8, pp. 1431–1440.

Christiano, Paul F., Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei (2017). "Deep Reinforcement Learning from Human Preferences". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, pp. 4299–4307. URL: https://proceedings.neurips.cc/paper/2017/hash/d5e2c0adad503c91f91df240d0cd4e49-Abstract.html.

Coffey, Kevin R, Ruby E Marx, and John F Neumaier (2019). "DeepSqueak: a deep learning-based system for detection and analysis of ultrasonic vocalizations". In: *Neuropsychopharmacology* 44.5, pp. 859–868.

Comuzzo, Marie (2023). "Singing with Whales: Exploring Human and Non-Human Connections". In: *SEM Student News* 19.1. Special Issue: Music and the Anthropocene. ISSN: 2578-4242. URL: https://cdn.ymaws.com/ethnomusicology.site-ym.com/resource/group/dc75b7e7-47d7-4d59-a660-19c3e0f7c83e/publications/19_1musicanthropocene/comuzzo_semsn_19-1.pdf.

Condon, Anne, Joan Feigenbaum, Carsten Lund, and Peter W. Shor (1995). "Probabilistically Checkable Debate Systems and Nonapproximability of PSPACE-Hard Functions".

In: *Chic. J. Theor. Comput. Sci.* 1995. URL: http://cjtcs.cs.uchicago.edu/articles/1995/4/contents.html.

Condon, Anne and Richard J. Lipton (1989). "On the Complexity of Space Bounded Interactive Proofs (Extended Abstract)". In: *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989*. IEEE Computer Society, pp. 462–467. DOI: 10.1109/SFCS.1989.63519. URL: https://doi.org/10.1109/SFCS.1989.63519.

Conneau, Alexis and Guillaume Lample (2019). "Cross-lingual Language Model Pretraining". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, pp. 7057–7067. URL: https://proceedings.neurips.cc/paper/2019/hash/c04c19c2c2474dbf5f7ac4372c5b9af1-Abstract.html.

Cook, Stephen A. (Jan. 1988). "Short propositional formulas represent nondeterministic computations". In: *Inform. Process. Lett.* 26.5, pp. 269–270. ISSN: 0020-0190. URL: https://doi.org/10.1016/0020-0190(88)90152-4.

Daniely, Amit and Shai Shalev-Shwartz (2014). "Optimal learners for multiclass problems". In: *Proceedings of The 27th Conference on Learning Theory, COLT 2014, Barcelona, Spain, June 13-15, 2014*. Ed. by Maria-Florina Balcan, Vitaly Feldman, and Csaba Szepesvári. Vol. 35. JMLR Workshop and Conference Proceedings. JMLR.org, pp. 287–316. URL: http://proceedings.mlr.press/v35/daniely14b.html.

Deecke, Volker B (2006). "Studying marine mammal cognition in the wild: a review of four decades of playback experiments". In: *Aquatic mammals* 32.4, pp. 461–482.

Défossez, Alexandre, Jade Copet, Gabriel Synnaeve, and Yossi Adi (2023). "High Fidelity Neural Audio Compression". In: *Trans. Mach. Learn. Res.* 2023. URL: https://openreview.net/forum?id=ivCd8z8zR2.

Descartes, René (1637). *Discourse on the method: And, meditations on first philosophy*. 1996. Originally published as Discours de la Méthode Pour bien conduire sa raison, et chercher la vérité dans les sciences, 1637. Yale University Press.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805*.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Association for Computational Linguistics, pp. 4171–4186. DOI: 10.18653/V1/N19-1423. URL: https://doi.org/10.18653/v1/n19-1423.

Dhariwal, Prafulla, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever (2020). "Jukebox: A Generative Model for Music". In: *CoRR* abs/2005.00341. arXiv: 2005.00341. URL: https://arxiv.org/abs/2005.00341.

Dinur, Irit (2007). "The PCP Theorem by gap amplification". In: *J. ACM* 54.3. (Preliminary version in *38th STOC*, 2006), p. 12. URL: https://doi.org/10.1145/1236457.1236459.

Dinur, Irit and Prahladh Harsha (2013). "Composition of low-error 2-query PCPs using decodable PCPs". In: *SIAM J. Comput.* 42.6. (Preliminary version in *51st FOCS*, 2009), pp. 2452–2486. URL: https://doi.org/10.1137/100788161.

Dinur, Irit and Omer Reingold (2006). "Assignment Testers: Towards a Combinatorial Proof of the PCP Theorem". In: *SIAM J. Comput.* 36. (Preliminary version in *45th FOCS*, 2004), pp. 975–1024. URL: https://doi.org/10.1137/S0097539705446962.

Doerr, Benjamin (Nov. 2019). "Probabilistic Tools for the Analysis of Randomized Optimization Heuristics". In: *Natural Computing Series*. Springer International Publishing, pp. 1–87. DOI: 10.1007/978-3-030-29414-4_1. URL: https://doi.org/10.1007%2F978-3-030-29414-4_1.

Dong, Hao-Wen, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang (2018). "MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. Ed. by Sheila A. McIlraith and Kilian Q. Weinberger. AAAI Press, pp. 34–41. DOI: 10.1609/AAAI.V32I1.11312. URL: https://doi.org/10.1609/aaai.v32i1.11312.

Dvir, Zeev and Benjamin L. Edelman (2019). "Matrix Rigidity and the Croot-Lev-Pach Lemma". In: *Theory Comput.* 15.8, pp. 1–7. eprint: 1708.01646. URL: https://doi.org/10.4086/toc.2019.v015a008.

Dvir, Zeev and Allen Liu (2020). "Fourier and Circulant Matrices are Not Rigid". In: *Theory Comput.* 16. (Preliminary version in 34*th Comput. Complexity Conf.*, 2019), pp. 1–48. eprint: 1902.07334. URL: https://doi.org/10.4086/toc.2020.v016a020.

Edmiston, Daniel, Phillip Keung, and Noah A. Smith (2022). "Domain Mismatch Doesn't Always Prevent Cross-lingual Transfer Learning". In: *Proceedings of the Thirteenth Language Resources and Evaluation Conference, LREC 2022, Marseille, France, 20-25 June 2022*. Ed. by Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Jan Odijk, and Stelios Piperidis. European Language Resources Association, pp. 892–899. URL: https://aclanthology.org/2022.lrec-1.94.

Elangovan, Aparna, Jiayuan He, and Karin Verspoor (2021). "Memorization vs. Generalization : Quantifying Data Leakage in NLP Performance Evaluation". In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*. Ed. by Paola Merlo, Jörg Tiedemann, and Reut Tsarfaty. Association for Computational Linguistics, pp. 1325–

1335. DOI: 10.18653/v1/2021.eacl-main.113. URL: https://doi.org/10.18653/v1/2021.eacl-main.113.

Erata, Ferhat, Orr Paradise, Timos Antonopoulos, ThanhVu Nguyen, Shafi Goldwasser, and Ruzica Piskac (2024). "Learning Randomized Reductions and Program Properties". In: *CoRR* abs/2412.18134. DOI: 10.48550/ARXIV.2412.18134. arXiv: 2412.18134. URL: https://doi.org/10.48550/arXiv.2412.18134.

Feige, Uriel and Shlomo Jozeph (2012). "Universal Factor Graphs". In: *Proc. 39th International Colloq. of Automata, Languages and Programming (ICALP), Part I*. Ed. by Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer. Vol. 7391. LNCS. Springer, pp. 339–350. eprint: 1204.6484. URL: https://doi.org/10.1007/978-3-642-31594-7_29.

Fleiss, Joseph L (1971). "Measuring nominal scale agreement among many raters." In: *Psychological bulletin* 76.5, p. 378.

Fokum, Daniel T., Zaria Chen Shui, Kerene Wright, Orr Paradise, Gunjan Mansingh, and Daniel Coore (2024). "A High School Camp on Algorithms and Coding in a Small Island Developing State". In: *Proceedings of the 55th ACM Technical Symposium on Computer Science Education, SIGCSE 2024, Volume 1, Portland, OR, USA, March 20-23, 2024*. Ed. by Ben Stephenson, Jeffrey A. Stone, Lina Battestilli, Samuel A. Rebelsky, and Libby Shoop. ACM, pp. 352–358. DOI: 10.1145/3626252.3630762. URL: https://doi.org/10.1145/3626252.3630762.

Font, Frederic, Gerard Roma, and Xavier Serra (2013). "Freesound technical demo". In: *ACM Multimedia Conference, MM '13, Barcelona, Spain, October 21-25, 2013*. Ed. by Alejandro Jaimes, Nicu Sebe, Nozha Boujemaa, Daniel Gatica-Perez, David A. Shamma, Marcel Worring, and Roger Zimmermann. ACM, pp. 411–412. DOI: 10.1145/2502081.2502245. URL: https://doi.org/10.1145/2502081.2502245.

Forney Jr., George David (1965). "Concatenated Codes". PhD thesis. Massachusetts Institute of Technology. URL: http://hdl.handle.net/1721.1/13449.

Frieder, Simon, Luca Pinchetti, Alexis Chevalier, Ryan-Rhys Griffiths, Tommaso Salvatori, Thomas Lukasiewicz, Philipp Petersen, and Julius Berner (2023). "Mathematical Capabilities of ChatGPT". In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine. URL: http://papers.nips.cc/paper_files/paper/2023/hash/58168e8a92994655d6da3939e7cc0918-Abstract-Datasets_and_Benchmarks.html.

Friedman, Joel (1993). "A note on matrix rigidity". In: *Combinatorica* 13.2, pp. 235–239. URL: https://doi.org/10.1007/BF01303207.

García, Hugo Flores, Prem Seetharaman, Rithesh Kumar, and Bryan Pardo (2023). "VampNet: Music Generation via Masked Acoustic Token Modeling". In: *Proceedings of the 24th International Society for Music Information Retrieval Conference, ISMIR 2023, Milan, Italy, November 5-9, 2023*. Ed. by Augusto Sarti, Fabio Antonacci, Mark Sandler, Paolo

Bestagini, Simon Dixon, Beici Liang, Gaël Richard, and Johan Pauwels, pp. 359–366. DOI: 10.5281/ZENODO.10265299. URL: https://doi.org/10.5281/zenodo.10265299.

Gemmeke, Jort F., Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter (2017a). "Audio Set: An ontology and human-labeled dataset for audio events". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017*. IEEE, pp. 776–780. DOI: 10.1109/ICASSP.2017.7952261. URL: https://doi.org/10.1109/ICASSP.2017.7952261.

— (2017b). "Audio Set: An ontology and human-labeled dataset for audio events". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017*. IEEE, pp. 776–780. DOI: 10.1109/ICASSP.2017.7952261. URL: https://doi.org/10.1109/ICASSP.2017.7952261.

Gero, Shane, Anne Bøttcher, Hal Whitehead, and Peter Teglberg Madsen (2016). "Socially segregated, sympatric sperm whale clans in the Atlantic Ocean". In: *Royal Society Open Science* 3.6, p. 160061.

Gero, Shane, Hal Whitehead, and Luke Rendell (2016a). "Individual, unit and vocal clan level identity cues in sperm whale codas". In: *Royal Society Open Science* 3.1, p. 150372.

— (2016b). "Individual, unit and vocal clan level identity cues in sperm whale codas". In: *Royal Society Open Science* 3.1, p. 150372.

Goldreich, Oded (2008a). *Computational complexity - a conceptual perspective*. Cambridge University Press. ISBN: 978-0-521-88473-0. DOI: 10.1017/CBO9780511804106. URL: https://doi.org/10.1017/CBO9780511804106.

— (2008b). *Computational complexity - a conceptual perspective*. Cambridge University Press. ISBN: 978-0-521-88473-0. DOI: 10.1017/CBO9780511804106. URL: https://doi.org/10.1017/CBO9780511804106.

— (2008c). "Probabilistic Proof Systems: A Primer". In: *Found. Trends Theor. Comput. Sci.* 3.1, pp. 1–91. DOI: 10.1561/0400000023. URL: https://doi.org/10.1561/0400000023.

— (2011). "A Sample of Samplers: A Computational Perspective on Sampling". In: *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*. Ed. by Oded Goldreich. Vol. 6650. LNCS. Springer, pp. 302–332. URL: https://doi.org/10.1007/978-3-642-22670-0_24.

Goldreich, Oded and Johan Håstad (1998). "On the Complexity of Interactive Proofs with Bounded Communication". In: *Inf. Process. Lett.* 67.4, pp. 205–214. DOI: 10.1016/S0020-0190(98)00116-1. URL: https://doi.org/10.1016/S0020-0190(98)00116-1.

Goldreich, Oded, Brendan Juba, and Madhu Sudan (2012). "A theory of goal-oriented communication". In: *J. ACM* 59.2, 8:1–8:65. DOI: 10.1145/2160158.2160161. URL: https://doi.org/10.1145/2160158.2160161.

Goldreich, Oded and Guy N. Rothblum (2018). "Simple Doubly-Efficient Interactive Proof Systems for Locally-Characterizable Sets". In: *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*. Ed. by Anna R. Karlin. Vol. 94. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik,

18:1–18:19. DOI: 10.4230/LIPICS.ITCS.2018.18. URL: https://doi.org/10.4230/LIPIcs.ITCS.2018.18.

Goldreich, Oded and Madhu Sudan (2006). "Locally testable codes and PCPs of almost linear length". In: *J. ACM* 53.4. (Preliminary version in *43rd FOCS*, 2002), pp. 558–655. URL: https://doi.org/10.1145/1162349.1162351.

Goldreich, Oded and Avishay Tal (2018). "Matrix rigidity of random Toeplitz matrices". In: *Comput. Complexity* 27.2. Preliminary version in *48th STOC*, 2002), pp. 305–350. URL: https://doi.org/10.1007/s00037-016-0144-9.

Goldreich, Oded, Salil P. Vadhan, and Avi Wigderson (2002). "On interactive proofs with a laconic prover". In: *Comput. Complex.* 11.1-2, pp. 1–53. DOI: 10.1007/S00037-002-0169-0. URL: https://doi.org/10.1007/s00037-002-0169-0.

Goldwasser, Shafi, David F. Gruber, Adam Tauman Kalai, and Orr Paradise (2023). "A Theory of Unsupervised Translation Motivated by Understanding Animal Communication". In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine.

Goldwasser, Shafi, Dan Gutfreund, Alexander Healy, Tali Kaufman, and Guy N. Rothblum (2007). "Verifying and decoding in constant depth". In: *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*. Ed. by David S. Johnson and Uriel Feige. ACM, pp. 440–449. DOI: 10.1145/1250790.1250855. URL: https://doi.org/10.1145/1250790.1250855.

Goldwasser, Shafi, Yael Tauman Kalai, and Guy N. Rothblum (2015). "Delegating Computation: Interactive Proofs for Muggles". In: *J. ACM* 62.4, 27:1–27:64. DOI: 10.1145/2699436. URL: https://doi.org/10.1145/2699436.

Goldwasser, Shafi, Silvio Micali, and Charles Rackoff (1985). "The Knowledge Complexity of Interactive Proof-Systems (Extended Abstract)". In: *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*. Ed. by Robert Sedgewick. ACM, pp. 291–304. DOI: 10.1145/22145.22178. URL: https://doi.org/10.1145/22145.22178.

Goldwasser, Shafi, Guy N. Rothblum, Jonathan Shafer, and Amir Yehudayoff (2021). "Interactive Proofs for Verifying Machine Learning". In: *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*. Ed. by James R. Lee. Vol. 185. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 41:1–41:19. DOI: 10.4230/LIPICS.ITCS.2021.41. URL: https://doi.org/10.4230/LIPIcs.ITCS.2021.41.

Gransden, Thomas, Neil Walkinshaw, and Rajeev Raman (2015). "SEPIA: Search for Proofs Using Inferred Automata". In: *Automated Deduction - CADE-25 - 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings*. Ed. by Amy P. Felty and Aart Middeldorp. Vol. 9195. Lecture Notes in Computer Science. Springer, pp. 246–255. DOI: 10.1007/978-3-319-21401-6_16. URL: https://doi.org/10.1007/978-3-319-21401-6_16.

Guei, Axel-Christian, Sylvain Christin, Nicolas Lecomte, and Éric Hervet (2024). "ECOGEN: Bird sounds generation using deep learning". In: *Methods in Ecology and Evolution* 15.1, pp. 69–79.

Gui, Azalea, Hannes Gamper, Sebastian Braun, and Dimitra Emmanouilidou (2024). "Adapting Frechet Audio Distance for Generative Music Evaluation". In: *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2024, Seoul, Republic of Korea, April 14-19, 2024*. IEEE, pp. 1331–1335. DOI: 10.1109/ICASSP48485.2024.10446663. URL: https://doi.org/10.1109/ICASSP48485.2024.10446663.

Gunderson, James P. and Louise F. Gunderson (2008). "Intelligence (is not equal to) Autonomy (is not equal to) Capability". In: *Performance Metrics for Intelligent Systems, PERMIS*. URL: https://idioms.thefreedictionary.com/a+coin+toss.

Gunn, Sam, Doseok Jang, Orr Paradise, Lucas Spangher, and Costas J. Spanos (2022). "Adversarial poisoning attacks on reinforcement learning-driven energy pricing". In: *Proceedings of the 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation, BuildSys 2022, Boston, Massachusetts, November 9-10, 2022*. Ed. by Jorge Ortiz. ACM, pp. 262–265. DOI: 10.1145/3563357.3564075. URL: https://doi.org/10.1145/3563357.3564075.

Hafemann, Luiz G., Luiz S. Oliveira, and Paulo Rodrigo Cavalin (2014). "Forest Species Recognition Using Deep Convolutional Neural Networks". In: *22nd International Conference on Pattern Recognition, ICPR 2014, Stockholm, Sweden, August 24-28, 2014*. IEEE Computer Society, pp. 1103–1107. DOI: 10.1109/ICPR.2014.199. URL: https://doi.org/10.1109/ICPR.2014.199.

Hagiwara, Masato (2023). "AVES: Animal Vocalization Encoder Based on Self-Supervision". In: *IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP 2023, Rhodes Island, Greece, June 4-10, 2023*. IEEE, pp. 1–5. DOI: 10.1109/ICASSP49357.2023.10095642. URL: https://doi.org/10.1109/ICASSP49357.2023.10095642.

Hammond, Lewis and Sam Adam-Day (2024). "Neural Interactive Proofs". In: *ICML 2024 Next Generation of AI Safety Workshop*. URL: https://openreview.net/forum?id=RhEND1litL.

Han, Jesse Michael, Igor Babuschkin, Harrison Edwards, Arvind Neelakantan, Tao Xu, Stanislas Polu, Alex Ray, Pranav Shyam, Aditya Ramesh, Alec Radford, and Ilya Sutskever (2021). "Unsupervised Neural Machine Translation with Generative Language Models Only". In: *CoRR* abs/2110.05448. arXiv: 2110.05448. URL: https://arxiv.org/abs/2110.05448.

He, Zhiwei, Xing Wang, Rui Wang, Shuming Shi, and Zhaopeng Tu (2022). "Bridging the Data Gap between Training and Inference for Unsupervised Neural Machine Translation". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*. Ed. by Smaranda Muresan, Preslav Nakov, and Aline Villavicencio. Association for Computational Linguistics, pp. 6611–6623. DOI: 10.18653/v1/2022.acl-long.456. URL: https://doi.org/10.18653/v1/2022.acl-long.456.

Hendrycks, Dan, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt (2021). "Measuring Mathematical Problem Solving With the MATH Dataset". In: *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*. Ed. by Joaquin Vanschoren and Sai-Kit Yeung. URL: https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a1Abstract-round2.html.

Hersh, Taylor A., Shane Gero, Luke Rendell, Maurício Cantor, Lindy Weilgart, Masao Amano, Stephen M. Dawson, Elisabeth Slooten, Christopher M. Johnson, Iain Kerr, Roger Payne, Andy Rogan, Ricardo Antunes, Olive Andrews, Elizabeth L. Ferguson, Cory Ann Hom-Weaver, Thomas F. Norris, Yvonne M. Barkley, Karlina P. Merkens, Erin M. Oleson, Thomas Doniol-Valcroze, James F. Pilkington, Jonathan Gordon, Manuel Fernandes, Marta Guerra, Leigh Hickmott, and Hal Whitehead (2022). "Evidence from sperm whale clans of symbolic marking in non-human cultures". In: *Proceedings of the National Academy of Sciences* 119.37, e2201692119. DOI: 10.1073/pnas.2201692119. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.2201692119. URL: https://www.pnas.org/doi/abs/10.1073/pnas.2201692119.

Hershey, Shawn, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, R. Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron J. Weiss, and Kevin W. Wilson (2017). "CNN architectures for large-scale audio classification". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017*. IEEE, pp. 131–135. DOI: 10.1109/ICASSP.2017.7952132. URL: https://doi.org/10.1109/ICASSP.2017.7952132.

Hobaiter, Catherine, Kirsty E Graham, and Richard W Byrne (2022). "Are ape gestures like words? Outstanding issues in detecting similarities and differences between human language and ape gesture". In: *Philosophical Transactions of the Royal Society B* 377.1860, p. 20210301.

Honghui, Yang and Fang Lanhao (2022). "Simulation of Marine Mammal Calls in Deep-Sea Environment". In: *International Conference on Autonomous Unmanned Systems*. Springer, pp. 2911–2920.

Hsu, Wei-Ning, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed (2021). "HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units". In: *IEEE ACM Trans. Audio Speech Lang. Process.* 29, pp. 3451–3460. DOI: 10.1109/TASLP.2021.3122291. URL: https://doi.org/10.1109/TASLP.2021.3122291.

Hu, Edward J., Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen (2022). "LoRA: Low-Rank Adaptation of Large Language Models". In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net. URL: https://openreview.net/forum?id=nZeVKeeFYf9.

Hu, Yaofang, Wanjie Wang, and Yi Yu (2022). "Graph matching beyond perfectly-overlapping Erdős-Rényi random graphs". In: *Stat. Comput.* 32.1, p. 19. DOI: 10.1007/s11222-022-10079-1. URL: https://doi.org/10.1007/s11222-022-10079-1.

Huang, Xuangui and Emanuele Viola (2021). "Average-Case Rigidity Lower Bounds". In: *Proc. 16th International Comp. Science Symp. in Russia (CSR)*. Ed. by Rahul Santhanam and Daniil Musatov. Vol. 12730. LNCS. Springer, pp. 186–205. URL: https://doi.org/10.1007/978-3-030-79416-3_11.

Hutchinson, Ben, Negar Rostamzadeh, Christina Greer, Katherine A. Heller, and Vinodkumar Prabhakaran (2022). "Evaluation Gaps in Machine Learning Practice". In: *FAccT '22: 2022 ACM Conference on Fairness, Accountability, and Transparency, Seoul, Republic of Korea, June 21 - 24, 2022*. ACM, pp. 1859–1876. DOI: 10.1145/3531146.3533233. URL: https://doi.org/10.1145/3531146.3533233.

Impagliazzo, Russell, Valentine Kabanets, and Avi Wigderson (2002). "In search of an easy witness: exponential time vs. probabilistic polynomial time". In: *J. Comput. Syst. Sci.* 65.4. (Preliminary version in *16th Computational Complexity Conference*, 2002), pp. 672–694. URL: https://doi.org/10.1016/S0022-0000(02)00024-7.

Irving, Geoffrey, Paul F. Christiano, and Dario Amodei (2018). "AI safety via debate". In: *CoRR* abs/1805.00899. arXiv: 1805.00899. URL: http://arxiv.org/abs/1805.00899.

Jia, Robin and Percy Liang (2017). "Adversarial Examples for Evaluating Reading Comprehension Systems". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. Ed. by Martha Palmer, Rebecca Hwa, and Sebastian Riedel. Association for Computational Linguistics, pp. 2021–2031. DOI: 10.18653/v1/d17-1215. URL: https://doi.org/10.18653/v1/d17-1215.

Johnson, Mark P and Peter L Tyack (2003). "A digital acoustic recording tag for measuring the response of wild marine mammals to sound". In: *IEEE journal of oceanic engineering* 28.1, pp. 3–12.

Juba, Brendan and Madhu Sudan (2008). "Universal semantic communication I". In: *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*. Ed. by Cynthia Dwork. ACM, pp. 123–132. DOI: 10.1145/1374376.1374397. URL: https://doi.org/10.1145/1374376.1374397.

Kadavath, Saurav, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan (2022). "Language Models (Mostly) Know What They Know". In: *CoRR* abs/2207.05221. DOI: 10.48550/arXiv.2207.05221. arXiv: 2207.05221. URL: https://doi.org/10.48550/arXiv.2207.05221.

Kahl, Stefan, Connor M Wood, Maximilian Eibl, and Holger Klinck (2021). "BirdNET: A deep learning solution for avian diversity monitoring". In: *Ecological Informatics* 61, p. 101236.

Kano, Takatomo, Sakriani Sakti, and Satoshi Nakamura (2021). "Transformer-Based Direct Speech-To-Speech Translation with Transcoder". In: *IEEE Spoken Language Technology Workshop, SLT 2021, Shenzhen, China, January 19-22, 2021*. IEEE, pp. 958–965. DOI: 10.1109/SLT48900.2021.9383496. URL: https://doi.org/10.1109/SLT48900.2021.9383496.

Katz, Daniel Martin, Michael James Bommarito, Shang Gao, and Pablo Arredondo (2023). "GPT-4 passes the bar exam". In: *Available at SSRN 4389233*.

Katz, Jonathan and Luca Trevisan (2000). "On the efficiency of local decoding procedures for error-correcting codes". In: *Proc. 32nd ACM Symp. on Theory of Computing (STOC)*, pp. 80–86. URL: https://doi.org/10.1145/335305.335315.

Khan, Salman H., Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah (2022). "Transformers in Vision: A Survey". In: *ACM Comput. Surv.* 54.10s, 200:1–200:41. DOI: 10.1145/3505244. URL: https://doi.org/10.1145/3505244.

Kiela, Douwe, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, Zhiyi Ma, Tristan Thrush, Sebastian Riedel, Zeerak Waseem, Pontus Stenetorp, Robin Jia, Mohit Bansal, Christopher Potts, and Adina Williams (2021). "Dynabench: Rethinking Benchmarking in NLP". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*. Ed. by Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou. Association for Computational Linguistics, pp. 4110–4124. DOI: 10.18653/V1/2021.NAACL-MAIN.324. URL: https://doi.org/10.18653/v1/2021.naacl-main.324.

Kilgour, Kevin, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi (2019). "Fréchet Audio Distance: A Reference-Free Metric for Evaluating Music Enhancement Algorithms". In: *20th Annual Conference of the International Speech Communication Association, Interspeech 2019, Graz, Austria, September 15-19, 2019*. Ed. by Gernot Kubin and Zdravko Kacic. ISCA, pp. 2350–2354. DOI: 10.21437/INTERSPEECH.2019-2219. URL: https://doi.org/10.21437/Interspeech.2019-2219.

Kim, Yongcheol, Seunghwan Seol, Hojun Lee, Geunho Park, and Jaehak Chung (2024). "WhistleGAN for Biomimetic Underwater Acoustic Covert Communication". In: *Electronics* 13.5, p. 964.

Kim, Yunsu, Miguel Graça, and Hermann Ney (2020). "When and Why is Unsupervised Neural Machine Translation Useless?" In: *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation, EAMT 2020, Lisboa, Portugal, November 3-5, 2020*. Ed. by Mikel L. Forcada, André Martins, Helena Moniz, Marco Turchi, Arianna Bisazza, Joss Moorkens, Ana Guerberof Arenas, Mary Nurminen, Lena

Marg, Sara Fumega, Bruno Martins, Fernando Batista, Luísa Coheur, Carla Parra Escartín, and Isabel Trancoso. European Association for Machine Translation, pp. 35–44. URL: https://aclanthology.org/2020.eamt-1.5/.

King, Stephanie L and Frants H Jensen (2023). "Rise of the machines: Integrating technology with playback experiments to study cetacean social cognition in the wild". In: *Methods in Ecology and Evolution* 14.8, pp. 1873–1886.

Kirchner, Jan Hendrik, Yining Chen, Harri Edwards, Jan Leike, Nat McAleese, and Yuri Burda (2024). "Prover-Verifier Games improve legibility of LLM outputs". In: *CoRR* abs/2407.13692. DOI: 10.48550/ARXIV.2407.13692. arXiv: 2407.13692. URL: https://doi.org/10.48550/arXiv.2407.13692.

Kopets, Ekaterina, Tatiana Shpilevaya, Oleg Vasilchenko, Artur Karimov, and Denis Butusov (2024). "Generating Synthetic Sperm Whale Voice Data Using StyleGAN2-ADA". In: *Big Data and Cognitive Computing* 8.4, p. 40.

Kumar, Rithesh, Prem Seetharaman, Alejandro Luebs, Ishaan Kumar, and Kundan Kumar (2023). "High-Fidelity Audio Compression with Improved RVQGAN". In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine.

Ladosz, Pawel, Lilian Weng, Minwoo Kim, and Hyondong Oh (2022). "Exploration in deep reinforcement learning: A survey". In: *Inf. Fusion* 85, pp. 1–22. DOI: 10.1016/J.INFFUS.2022.03.003. URL: https://doi.org/10.1016/j.inffus.2022.03.003.

Lakhotia, Kushal, Eugene Kharitonov, Wei-Ning Hsu, Yossi Adi, Adam Polyak, Benjamin Bolte, Tu-Anh Nguyen, Jade Copet, Alexei Baevski, Abdelrahman Mohamed, and Emmanuel Dupoux (2021). "On Generative Spoken Language Modeling from Raw Audio". In: *Transactions of the Association for Computational Linguistics* 9. Ed. by Brian Roark and Ani Nenkova, pp. 1336–1354. DOI: 10.1162/tacl_a_00430. URL: https://aclanthology.org/2021.tacl-1.79.

Lample, Guillaume, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato (2018a). "Unsupervised Machine Translation Using Monolingual Corpora Only". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. URL: https://openreview.net/forum?id=rkYTTf-AZ.

Lample, Guillaume, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato (2018b). "Phrase-Based & Neural Unsupervised Machine Translation". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*. Ed. by Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii. Association for Computational Linguistics, pp. 5039–5049. URL: https://aclanthology.org/D18-1549/.

Lample, Guillaume, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato (Oct. 2018c). "Phrase-Based & Neural Unsupervised Machine Translation". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Process-*

ing. Brussels, Belgium: Association for Computational Linguistics, pp. 5039–5049. DOI: 10.18653/v1/D18-1549. URL: https://aclanthology.org/D18-1549.

Lee, Nayoung, Kartik Sreenivasan, Jason D. Lee, Kangwook Lee, and Dimitris Papailiopoulos (2024). "Teaching Arithmetic to Small Transformers". In: *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 6-11, 2024*. OpenReview.net.

Leigh Star, Susan (2010). "This is not a boundary object: Reflections on the origin of a concept". In: *Science, technology, & human values* 35.5, pp. 601–617.

Leitao, Antonio, Maxime Lucas, Simone Poetto, Taylor A. Hersh, Shane Gero, David F. Gruber, Michael Bronstein, and Giovanni Petri (May 2024). "Evidence of social learning across symbolic cultural barriers in sperm whales". In: *eLife*. DOI: 10.7554/elife.96362.1. URL: http://dx.doi.org/10.7554/eLife.96362.1.

Levy, Ido, Orr Paradise, Boaz Carmeli, Ron Meir, Shafi Goldwasser, and Yonatan Belinkov (2025). "Unsupervised Translation of Emergent Communication". In: *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*. Ed. by Toby Walsh, Julie Shah, and Zico Kolter. AAAI Press, pp. 23231–23239. DOI: 10.1609/AAAI.V39I22.34489. URL: https://doi.org/10.1609/aaai.v39i22.34489.

Lightman, Hunter, Vineet Kosaraju, Yura Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe (2024). "Let's Verify Step by Step". In: *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 6-11, 2024*. OpenReview.net.

Lin, John C, David N Younessi, Sai S Kurapati, Oliver Y Tang, and Ingrid U Scott (2023). "Comparison of GPT-3.5, GPT-4, and human user performance on a practice ophthalmology written examination". In: *Eye*, pp. 1–2.

Livi, Lorenzo and Antonello Rizzi (Aug. 2013). "The graph matching problem". In: *Pattern Analysis and Applications* 16, pp. 253–283. DOI: 10.1007/s10044-012-0284-8.

Lokam, Satyanarayana V. (2009). "Complexity Lower Bounds using Linear Algebra". In: *Found. Trends Theor. Comput. Sci.* 4.1-2, pp. 1–155. URL: https://doi.org/10.1561/0400000011.

Lynum, André, Erwin Marsi, Lars Bungum, and Björn Gambäck (2012). "Disambiguating Word Translations with Target Language Models". In: *Text, Speech and Dialogue - 15th International Conference, TSD 2012, Brno, Czech Republic, September 3-7, 2012. Proceedings*. Ed. by Petr Sojka, Ales Horák, Ivan Kopecek, and Karel Pala. Vol. 7499. Lecture Notes in Computer Science. Springer, pp. 378–385. DOI: 10.1007/978-3-642-32790-2_46. URL: https://doi.org/10.1007/978-3-642-32790-2_46.

Malach, Eran (2023). "Auto-Regressive Next-Token Predictors are Universal Learners". In: *CoRR* abs/2309.06979. DOI: 10.48550/ARXIV.2309.06979. arXiv: 2309.06979. URL: https://doi.org/10.48550/arXiv.2309.06979.

Marchisio, Kelly, Kevin Duh, and Philipp Koehn (2020). "When Does Unsupervised Machine Translation Work?" In: *Proceedings of the Fifth Conference on Machine Translation, WMT@EMNLP 2020, Online, November 19-20, 2020*. Ed. by Loïc Barrault, Ondrej

Bojar, Fethi Bougares, Rajen Chatterjee, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Alexander Fraser, Yvette Graham, Paco Guzman, Barry Haddow, Matthias Huck, Antonio Jimeno-Yepes, Philipp Koehn, André Martins, Makoto Morishita, Christof Monz, Masaaki Nagata, Toshiaki Nakazawa, and Matteo Negri. Association for Computational Linguistics, pp. 571–583. URL: https://aclanthology.org/2020.wmt-1.68/.

Miceli Barone, Antonio Valerio (Aug. 2016). "Towards cross-lingual distributed representations without parallel text trained with adversarial autoencoders". In: *Proceedings of the 1st Workshop on Representation Learning for NLP*. Berlin, Germany: Association for Computational Linguistics, pp. 121–126. DOI: 10.18653/v1/W16-1614. URL: https://aclanthology.org/W16-1614.

Mie, Thilo (2009). "Short PCPPs verifiable in polylogarithmic time with $O(1)$ queries". In: *Ann. Math. Artif. Intell.* 56.3-4, pp. 313–338. URL: https://doi.org/10.1007/s10472-009-9169-y.

Mitchell, Margaret, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru (2019). "Model Cards for Model Reporting". In: *Proceedings of the Conference on Fairness, Accountability, and Transparency, FAT* 2019, Atlanta, GA, USA, January 29-31, 2019*. Ed. by danah boyd and Jamie H. Morgenstern. ACM, pp. 220–229. DOI: 10.1145/3287560.3287596. URL: https://doi.org/10.1145/3287560.3287596.

Mitchell, Melanie and David C Krakauer (2023). "The debate over understanding in AI's large language models". In: *Proceedings of the National Academy of Sciences* 120.13, e2215907120.

Møhl, Bertel, Magnus Wahlberg, Peter T Madsen, Anders Heerfordt, and Anders Lund (2003). "The monopulsed nature of sperm whale clicks". In: *The Journal of the Acoustical Society of America* 114.2, pp. 1143–1154.

Mohri, Mehryar, Afshin Rostamizadeh, and Ameet Talwalkar (2018). *Foundations of machine learning*. MIT press.

Moore, Karen E, William A Watkins, and Peter L Tyack (1993). "Pattern similarity in shared codas from sperm whales (Physeter catodon)". In: *Marine Mammal Science* 9.1, pp. 1–9.

Moshkovitz, Dana and Ran Raz (2008). "Sub-Constant Error Low Degree Test of Almost-Linear Size". In: *SIAM J. Comput.* 38.1. (Preliminary version in *38th STOC*, 2006), pp. 140–180. URL: https://doi.org/10.1137/060656838.

Moussad, Bernard, Rahmatullah Roche, and Debswapna Bhattacharya (2023). "The transformative power of transformers in protein structure prediction". In: *Proceedings of the National Academy of Sciences* 120.32, e2303499120.

Murty, Shikhar, Orr Paradise, and Pratyusha Sharma (2023). "Pseudointelligence: A Unifying Lens on Language Model Evaluation". In: *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*. Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. Association for Computational Linguistics, pp. 7284–7290. DOI: 10.18653/V1/2023.FINDINGS-EMNLP.485. URL: https://doi.org/10.18653/v1/2023.findings-emnlp.485.

Nair, Ashvin, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel (2018). "Overcoming Exploration in Reinforcement Learning with Demonstrations". In: *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*. IEEE, pp. 6292–6299. DOI: 10.1109/ICRA.2018.8463162. URL: https://doi.org/10.1109/ICRA.2018.8463162.

Natarajan, B. K. (1989). "On Learning Sets and Functions". In: *Mach. Learn.* 4, pp. 67–97. DOI: 10.1007/BF00114804. URL: https://doi.org/10.1007/BF00114804.

Nie, Yixin, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela (2020). "Adversarial NLI: A New Benchmark for Natural Language Understanding". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*. Ed. by Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault. Association for Computational Linguistics, pp. 4885–4901. DOI: 10.18653/v1/2020.acl-main.441. URL: https://doi.org/10.18653/v1/2020.acl-main.441.

Nogueira, Rodrigo Frassetto, Zhiying Jiang, and Jimmy Lin (2021). "Investigating the Limitations of the Transformers with Simple Arithmetic Tasks". In: *CoRR* abs/2102.13019. arXiv: 2102.13019. URL: https://arxiv.org/abs/2102.13019.

Oord, Aäron van den, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu (2016). "WaveNet: A Generative Model for Raw Audio". In: *The 9th ISCA Speech Synthesis Workshop, SSW 2016, Sunnyvale, CA, USA, September 13-15, 2016*. Ed. by Alan W. Black. ISCA, p. 125. URL: https://www.isca-archive.org/ssw_2016/vandenoord16_ssw.html.

OpenAI (2023). "GPT-4 Technical Report". In: *CoRR* abs/2303.08774. DOI: 10.48550/arXiv.2303.08774. arXiv: 2303.08774. URL: https://doi.org/10.48550/arXiv.2303.08774.

Ouyang, Long, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe (2022). "Training language models to follow instructions with human feedback". In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*. Ed. by Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh. URL: http://papers.nips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.

Palamas, Theodoros (2017). "Investigating the ability of neural networks to learn simple modular arithmetic". MA thesis. University of Edinburgh. URL: https://project-archive.inf.ed.ac.uk/msc/20172390/msc_proj.pdf.

Paradise, Orr (2021a). "Smooth and Strong PCPs". In: *Comput. Complexity* 30.1. (Preliminary version in 11*th ITCS*, 2020), p. 1. URL: https://doi.org/10.1007/s00037-020-00199-3.

Paradise, Orr (2021b). "Smooth and Strong PCPs". In: *Comput. Complex.* 30.1, p. 1. DOI: 10.1007/S00037-020-00199-3. URL: https://doi.org/10.1007/s00037-020-00199-3.

Paradise, Orr, Liangyuan Chen, Pranav Muralikrishnan, Hugo Flores García, Bryan Pardo, Roee Diamant, David Gruber, Shane Gero, and Shafi Goldwasser (May 2025). "Towards A Translative Model of Sperm Whale Vocalization". In submission.

Payne, Roger S and Scott McVay (1971). "Songs of Humpback Whales: Humpbacks emit sounds in long, predictable patterns ranging over frequencies audible to humans." In: *Science* 173.3997, pp. 585–597.

Perez, Ethan, Sam Ringer, Kamile Lukosiute, Karina Nguyen, Edwin Chen, Scott Heiner, Craig Pettit, Catherine Olsson, Sandipan Kundu, Saurav Kadavath, Andy Jones, Anna Chen, Benjamin Mann, Brian Israel, Bryan Seethor, Cameron McKinnon, Christopher Olah, Da Yan, Daniela Amodei, Dario Amodei, Dawn Drain, Dustin Li, Eli Tran-Johnson, Guro Khundadze, Jackson Kernion, James Landis, Jamie Kerr, Jared Mueller, Jeeyoon Hyun, Joshua Landau, Kamal Ndousse, Landon Goldberg, Liane Lovitt, Martin Lucas, Michael Sellitto, Miranda Zhang, Neerav Kingsland, Nelson Elhage, Nicholas Joseph, Noemi Mercado, Nova DasSarma, Oliver Rausch, Robin Larson, Sam McCandlish, Scott Johnston, Shauna Kravec, Sheer El Showk, Tamera Lanham, Timothy Telleen-Lawton, Tom Brown, Tom Henighan, Tristan Hume, Yuntao Bai, Zac Hatfield-Dodds, Jack Clark, Samuel R. Bowman, Amanda Askell, Roger Grosse, Danny Hernandez, Deep Ganguli, Evan Hubinger, Nicholas Schiefer, and Jared Kaplan (July 2023). "Discovering Language Model Behaviors with Model-Written Evaluations". In: *Findings of the Association for Computational Linguistics: ACL 2023*. Toronto, Canada: Association for Computational Linguistics, pp. 13387–13434. DOI: 10.18653/v1/2023.findings-acl.847. URL: https://aclanthology.org/2023.findings-acl.847.

Pippenger, Nicholas and Michael J. Fischer (Apr. 1979). "Relations among complexity measures". In: *J. ACM* 26.2, pp. 361–381. ISSN: 0004-5411. URL: https://doi.org/10.1145/322123.322138.

Polu, Stanislas and Ilya Sutskever (2020). "Generative Language Modeling for Automated Theorem Proving". In: *CoRR* abs/2009.03393. arXiv: 2009.03393. URL: https://arxiv.org/abs/2009.03393.

Premoli, Marika, Daniele Baggi, Marco Bianchetti, Alessandro Gnutti, Marco Bondaschi, Andrea Mastinu, Pierangelo Migliorati, Alberto Signoroni, Riccardo Leonardi, Maurizio Memo, et al. (2021). "Automatic classification of mice vocalizations using Machine Learning techniques and Convolutional Neural Networks". In: *PloS one* 16.1, e0244636.

Raji, Inioluwa Deborah, Emily Denton, Emily M. Bender, Alex Hanna, and Amandalynne Paullada (2021). "AI and the Everything in the Whole Wide World Benchmark". In: *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*. Ed. by Joaquin Vanschoren and Sai-Kit Yeung. URL: https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/084b6fbb10729ed4da8c3d3f5a3ae7c9-Abstract-round2.html.

Ramya, C. (2020). "Recent Progress on Matrix Rigidity - A Survey". (manuscript). eprint: 2009.09460.

Ranathunga, Surangika, En-Shiun Annie Lee, Marjana Prifti Skenduli, Ravi Shekhar, Mehreen Alam, and Rishemjit Kaur (2021). "Neural machine translation for low-resource languages: A survey". In: *arXiv preprint arXiv:2106.15115*.

Ravi, Sujith and Kevin Knight (2011). "Deciphering Foreign Language". In: *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*. Ed. by Dekang Lin, Yuji Matsumoto, and Rada Mihalcea. The Association for Computer Linguistics, pp. 12–21. URL: https://aclanthology.org/P11-1002/.

Razborov, Alexander A. (1989). "On Rigid Matrices (in Russian)". (manuscript). URL: http://people.cs.uchicago.edu/~razborov/files/rigid.pdf.

Reilly, Johnny, John D Goodwin, Sihao Lu, and Andriy S Kozlov (2023). "Bidirectional Generative Adversarial Representation Learning for Natural Stimulus Synthesis". In: *Journal of Neurophysiology*.

Reingold, Omer, Guy N. Rothblum, and Ron D. Rothblum (2021). "Constant-Round Interactive Proofs for Delegating Computation". In: *SIAM J. Comput.* 50.3. DOI: 10.1137/16M1096773. URL: https://doi.org/10.1137/16M1096773.

Ren, Hanlin, Rahul Santhanam, and Zhikun Wang (2022). "On the Range Avoidance Problem for Circuits". In: *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*. IEEE, pp. 640–650. DOI: 10.1109/FOCS54457.2022.00067. URL: https://doi.org/10.1109/FOCS54457.2022.00067.

Rendell, Luke, Sarah L Mesnick, Merel L Dalebout, Jessica Burtenshaw, and Hal Whitehead (2012). "Can genetic differences explain vocal dialect variation in sperm whales, Physeter macrocephalus?" In: *Behavior genetics* 42, pp. 332–343.

Rendell, Luke E and Hal Whitehead (2003). "Vocal clans in sperm whales (Physeter macrocephalus)". In: *Proceedings of the Royal Society of London. Series B: Biological Sciences* 270.1512, pp. 225–231.

Ribeiro, Marco Túlio, Tongshuang Wu, Carlos Guestrin, and Sameer Singh (2021). "Beyond Accuracy: Behavioral Testing of NLP Models with Checklist (Extended Abstract)". In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*. Ed. by Zhi-Hua Zhou. ijcai.org, pp. 4824–4828. DOI: 10.24963/ijcai.2021/659. URL: https://doi.org/10.24963/ijcai.2021/659.

Rodríguez-Garavito, César, David F. Gruber, Ashley Nemeth, and Gasper Begus (Mar. 2025). "What If We Understood What Animals Are Saying? The Legal Impact Of AI-assisted Studies Of Animal Communication". In: *Ecology Law Quarterly* 52.1. DOI: 10.15779/Z383X83N5Q. URL: https://ssrn.com/abstract=5165527.

Romera-Paredes, Bernardino, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming

Wang, Omar Fawzi, et al. (2024). "Mathematical discoveries from program search with large language models". In: *Nature* 625.7995, pp. 468–475.

Rothblum, Guy N., Salil P. Vadhan, and Avi Wigderson (2013). "Interactive proofs of proximity: delegating computation in sublinear time". In: *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*. Ed. by Dan Boneh, Tim Roughgarden, and Joan Feigenbaum. ACM, pp. 793–802. DOI: 10.1145/2488608.2488709. URL: https://doi.org/10.1145/2488608.2488709.

Rozière, Baptiste, Jie Zhang, François Charton, Mark Harman, Gabriel Synnaeve, and Guillaume Lample (2022). "Leveraging Automated Unit Tests for Unsupervised Code Translation". In: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net. URL: https://openreview.net/forum?id=cmt-6KtR4c4.

Santhanam, Rahul and R. Ryan Williams (2014). "On Uniformity and Circuit Lower Bounds". In: *Comput. Complexity* 23.2. (Preliminary version in *28th Computational Complexity Conference*, 2013), pp. 177–205. URL: https://doi.org/10.1007/s00037-014-0087-y.

Saunders, William, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike (2022). "Self-critiquing models for assisting human evaluators". In: *CoRR* abs/2206.05802. DOI: 10.48550/arXiv.2206.05802. arXiv: 2206.05802. URL: https://doi.org/10.48550/arXiv.2206.05802.

Sayigh, Laela, Mary Ann Daher, Julie Allen, Helen Gordon, Katherine Joyce, Claire Stuhlmann, and Peter Tyack (2016). "The watkins marine mammal sound database: An online, freely accessible resource". In: *Proceedings of Meetings on Acoustics*. Vol. 27. AIP Publishing.

Schulz, Tyler M., Hal Whitehead, Shane Gero, and Luke Rendell (2011). "Individual vocal production in a sperm whale (Physeter macrocephalus) social unit". In: *Marine Mammal Science* 27.1, pp. 149–166. DOI: https://doi.org/10.1111/j.1748-7692.2010.00399.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1748-7692.2010.00399.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1748-7692.2010.00399.x.

Shalev-Shwartz, Shai and Shai Ben-David (2014). *Understanding Machine Learning - From Theory to Algorithms*. Cambridge University Press. ISBN: 978-1-10-705713-5. URL: http://www.cambridge.org/de/academic/subjects/computer-science/pattern-recognition-and-machine-learning/understanding-machine-learning-theory-algorithms.

Shamir, Adi (1992). "IP = PSPACE". In: *J. ACM* 39.4, pp. 869–877. DOI: 10.1145/146585.146609. URL: https://doi.org/10.1145/146585.146609.

Sharma, Pratyusha, Shane Gero, Roger Payne, David F Gruber, Daniela Rus, Antonio Torralba, and Jacob Andreas (2024a). "Contextual and combinatorial structure in sperm whale vocalisations". In: *Nature Communications* 15.1, p. 3617.

Sharma, Pratyusha, Shane Gero, Daniela Rus, Antonio Torralba, and Jacob Andreas (2024b). "WhaleLM: Finding Structure and Information in Sperm Whale Vocalizations and Behavior with Machine Learning". In: *bioRxiv*. DOI: 10.1101/2024.10.31.621071. eprint: https://www.biorxiv.org/content/early/2024/11/11/2024.10.31.621071.full.

pdf. URL: https://www.biorxiv.org/content/early/2024/11/11/2024.10.31.621071.

Shin, Taylor, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh (Nov. 2020). "AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, pp. 4222–4235. DOI: 10.18653/v1/2020.emnlp-main.346. URL: https://aclanthology.org/2020.emnlp-main.346.

Shirali, Ali, Rediet Abebe, and Moritz Hardt (2023). "A Theory of Dynamic Benchmarks". In: *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net. URL: https://openreview.net/pdf?id=i8L9qoeZOS.

Shokrollahi, Mohammad Amin, Daniel A. Spielman, and Volker Stemann (1997). "A Remark on Matrix Rigidity". In: *Inform. Process. Lett.* 64.6, pp. 283–285. URL: https://doi.org/10.1016/S0020-0190(97)00190-7.

Siu, Kai-Yeung and Vwani P. Roychowdhury (1992). "Optimal Depth Neural Networks for Multiplication and Related Problems". In: *Advances in Neural Information Processing Systems 5, [NIPS Conference, Denver, Colorado, USA, November 30 - December 3, 1992]*. Ed. by Stephen Jose Hanson, Jack D. Cowan, and C. Lee Giles. Morgan Kaufmann, pp. 59–64. URL: http://papers.nips.cc/paper/657-optimal-depth-neural-networks-for-multiplication-and-related-problems.

Song, Kaitao, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu (2019). "MASS: Masked Sequence to Sequence Pre-training for Language Generation". In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 5926–5936. URL: http://proceedings.mlr.press/v97/song19d.html.

Spielman, Daniel A. (Nov. 1996). "Linear-Time Encodable and Decodable Error-Correcting Codes". In: *IEEE Trans. Inform. Theory* 42.6. (Preliminary version in *27th STOC*, 1995), pp. 1723–1732. URL: https://doi.org/10.1109/18.556668.

Spielman, Daniel A. and Shang-Hua Teng (Oct. 2009). "Smoothed Analysis: An Attempt to Explain the Behavior of Algorithms in Practice". In: *Commun. ACM* 52.10, 76–84. ISSN: 0001-0782. DOI: 10.1145/1562764.1562785. URL: https://doi.org/10.1145/1562764.1562785.

Star, Susan Leigh and James R Griesemer (1989). "Institutional ecology,translations' and boundary objects: Amateurs and professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39". In: *Social studies of science* 19.3, pp. 387–420.

Suresh, Harini, Divya Shanmugam, Tiffany Chen, Annie G. Bryan, Alexander D'Amour, John V. Guttag, and Arvind Satyanarayan (2023). "Kaleidoscope: Semantically-grounded, context-specific ML model evaluation". In: *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems, CHI 2023, Hamburg, Germany, April 23-28, 2023*. Ed. by Albrecht Schmidt, Kaisa Väänänen, Tesh Goyal, Per Ola Kristensson, Ani-

cia Peters, Stefanie Mueller, Julie R. Williamson, and Max L. Wilson. ACM, 775:1–775:13. DOI: 10.1145/3544548.3581482. URL: https://doi.org/10.1145/3544548.3581482.

Sutton, Richard S., David A. McAllester, Satinder Singh, and Yishay Mansour (1999). "Policy Gradient Methods for Reinforcement Learning with Function Approximation". In: *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*. Ed. by Sara A. Solla, Todd K. Leen, and Klaus-Robert Müller. The MIT Press, pp. 1057–1063. URL: http://papers.nips.cc/paper/1713-policy-gradient-methods-for-reinforcement-learning-with-function-approximation.

Suzuki, Toshitaka N, David Wheatcroft, and Michael Griesser (2020). "The syntax–semantics interface in animal vocal communication". In: *Philosophical Transactions of the Royal Society B* 375.1789, p. 20180405.

Tafjord, Oyvind, Bhavana Dalvi, and Peter Clark (2020). "ProofWriter: Generating Implications, Proofs, and Abductive Statements over Natural Language". In: *Findings*. URL: https://api.semanticscholar.org/CorpusID:229371222.

Tran, Chau, Shruti Bhosale, James Cross, Philipp Koehn, Sergey Edunov, and Angela Fan (2021). "Facebook AI's WMT21 News Translation Task Submission". In: *Proceedings of the Sixth Conference on Machine Translation, WMT@EMNLP 2021, Online Event, November 10-11, 2021*. Ed. by Loïc Barrault, Ondrej Bojar, Fethi Bougares, Rajen Chatterjee, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Alexander Fraser, Markus Freitag, Yvette Graham, Roman Grundkiewicz, Paco Guzman, Barry Haddow, Matthias Huck, Antonio Jimeno-Yepes, Philipp Koehn, Tom Kocmi, André Martins, Makoto Morishita, and Christof Monz. Association for Computational Linguistics, pp. 205–215. URL: https://aclanthology.org/2021.wmt-1.19.

Trevisan, Luca (2013). "Inapproximability of Combinatorial Optimization Problems". In: *Paradigms of Combinatorial Optimization*. Ed. by Vangelis Th. Paschos. Wiley Publishers. Chap. 13, pp. 381–434. eprint: cs.CC/0409043. URL: https://doi.org/10.1002/9781118600207.ch13.

Trinh, Trieu H., Yuhuai Wu, Quoc V. Le, He He, and Thang Luong (2024). "Solving olympiad geometry without human demonstrations". In: *Nat.* 625.7995, pp. 476–482. DOI: 10.1038/S41586-023-06747-5. URL: https://doi.org/10.1038/s41586-023-06747-5.

Turing, Alan M. (1950). "Computing machinery and intelligence". In: *Mind* LIX.236, pp. 433–460. DOI: 10.1093/mind/LIX.236.433. URL: https://doi.org/10.1093/mind/LIX.236.433.

Turpin, Miles, Julian Michael, Ethan Perez, and Samuel R. Bowman (2023). "Language Models Don't Always Say What They Think: Unfaithful Explanations in Chain-of-Thought Prompting". In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine. URL: http://papers.nips.cc/paper_files/paper/2023/hash/ed3fea9033a80fea1376299fa7863f4a-Abstract-Conference.html.

Tyack, Peter (1983). "Differential response of humpback whales, Megaptera novaeangliae, to playback of song or social sounds". In: *Behavioral Ecology and Sociobiology* 13, pp. 49–55.

Uesato, Jonathan, Nate Kushman, Ramana Kumar, H. Francis Song, Noah Y. Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins (2022). "Solving math word problems with process- and outcome-based feedback". In: *CoRR* abs/2211.14275. DOI: 10.48550/ARXIV.2211.14275. arXiv: 2211.14275. URL: https://doi.org/10.48550/arXiv.2211.14275.

Valiant, Leslie G. (1977). "Graph-Theoretic Arguments in Low-Level Complexity". In: *Proc. 6th Symposium of Mathematical Foundations of Computer Science (MFCS)*. Ed. by Jozef Gruska. Vol. 53. LNCS. Springer, pp. 162–176. URL: https://doi.org/10.1007/3-540-08353-7_135.

— (1984). "A Theory of the Learnable". In: *Commun. ACM* 27.11, pp. 1134–1142. DOI: 10.1145/1968.1972. URL: https://doi.org/10.1145/1968.1972.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin (2017). "Attention is All you Need". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, pp. 5998–6008. URL: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Viola, Emanuele (2020). "New lower bounds for probabilistic degree and AC^0 with parity gates". (manuscript).

Wäldchen, Stephan, Kartikey Sharma, Berkant Turan, Max Zimmer, and Sebastian Pokutta (May 2024). "Interpretability Guarantees with Merlin-Arthur Classifiers". In: *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*. Ed. by Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li. Vol. 238. Proceedings of Machine Learning Research. PMLR, pp. 1963–1971. URL: https://proceedings.mlr.press/v238/waldchen24a.html.

Wang, Boshi, Xiang Yue, and Huan Sun (2023). "Can ChatGPT Defend its Belief in Truth? Evaluating LLM Reasoning via Debate". In: *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*. Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. Association for Computational Linguistics, pp. 11865–11881. DOI: 10.18653/V1/2023.FINDINGS-EMNLP.795. URL: https://doi.org/10.18653/v1/2023.findings-emnlp.795.

Watkins, William A and William E Schevill (1977). "Sperm whale codas". In: *The Journal of the Acoustical Society of America* 62.6, pp. 1485–1490.

Wei, Jason, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou (2022). "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models". In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*. Ed. by Sanmi Koyejo, S.

Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh. URL: http://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Weilgart, Linda and Hal Whitehead (1993). "Coda communication by sperm whales (Physeter macrocephalus) off the Galapagos Islands". In: *Canadian Journal of Zoology* 71.4, pp. 744–752.

— (1997). "Group-specific dialects and geographical variation in coda repertoire in South Pacific sperm whales". In: *Behavioral Ecology and Sociobiology* 40, pp. 277–285.

Welleck, Sean, Jiacheng Liu, Ximing Lu, Hannaneh Hajishirzi, and Yejin Choi (2022). "NaturalProver: Grounded Mathematical Proof Generation with Language Models". In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*. Ed. by Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh. URL: http://papers.nips.cc/paper_files/paper/2022/hash/1fc548a8243ad06616eee731e0572927-Abstract-Conference.html.

Whitehead, Hal (2003). *Sperm whales: social evolution in the ocean*. University of Chicago press.

Whitehead, Hal and Linda Weilgart (1991). "Patterns of visually observable behaviour and vocalizations in groups of female sperm whales". In: *Behaviour*, pp. 275–296.

Williams, R. Ryan (2013). "Improving Exhaustive Search Implies Superpolynomial Lower Bounds". In: *SIAM J. Comput.* 42.3. (Preliminary version in *42nd STOC*, 2010), pp. 1218–1244. URL: https://doi.org/10.1137/10080703X.

— (2014). "Nonuniform ACC Circuit Lower Bounds". In: *J. ACM* 61.1. (Preliminary version in *Computational Complexity Conference*, 2011), 2:1–2:32. URL: https://doi.org/10.1145/2559903.

— (2016). "Natural Proofs versus Derandomization". In: *SIAM J. Comput.* 45.2. (Preliminary version in *45th STOC*, 2013), pp. 497–529. eprint: 1212.1891. URL: https://doi.org/10.1137/130938219.

Wittgenstein, Ludwig (1953). *Philosophical Investigations*. Oxford: Basil Blackwell. ISBN: 0631119000.

Wu, Yusong, Ke Chen, Tianyu Zhang, Yuchen Hui, Taylor Berg-Kirkpatrick, and Shlomo Dubnov (2023). "Large-Scale Contrastive Language-Audio Pretraining with Feature Fusion and Keyword-to-Caption Augmentation". In: *IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP 2023, Rhodes Island, Greece, June 4-10, 2023*. IEEE, pp. 1–5. DOI: 10.1109/ICASSP49357.2023.10095969. URL: https://doi.org/10.1109/ICASSP49357.2023.10095969.

Wu*, Yusong, Ke Chen*, Tianyu Zhang*, Yuchen Hui*, Taylor Berg-Kirkpatrick, and Shlomo Dubnov (2023). "Large-scale Contrastive Language-Audio Pretraining with Feature Fusion and Keyword-to-Caption Augmentation". In: *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*.

Wunderlich, Henning (2012). "On a Theorem of Razborov". In: *Comput. Complexity* 21.3, pp. 431–477. URL: https://doi.org/10.1007/s00037-011-0021-5.

Xie, Jiangjian, Yujie Zhong, Junguo Zhang, Shuo Liu, Changqing Ding, and Andreas Triantafyllopoulos (2023). "A review of automatic recognition technology for bird vocalizations in the deep learning era". In: *Ecological Informatics* 73, p. 101927.

Xu, Lian, Mohammed Bennamoun, Senjian An, Ferdous Sohel, and Farid Boussaid (2019). "Deep learning for marine species recognition". In: *Handbook of deep learning applications*, pp. 129–145.

Yang, Kaiyu, Aidan M. Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan J. Prenger, and Animashree Anandkumar (2023). "LeanDojo: Theorem Proving with Retrieval-Augmented Language Models". In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine. URL: http://papers.nips.cc/paper_files/paper/2023/hash/4441469427094f8873d0fecb0c4e1cee-Abstract-Datasets_and_Benchmarks.html.

Yang, Mengjiao, Dale Schuurmans, Pieter Abbeel, and Ofir Nachum (2022). "Chain of Thought Imitation with Procedure Cloning". In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*. Ed. by Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh. URL: http://papers.nips.cc/paper_files/paper/2022/hash/ebdb990471f653dffb425eff03c7c980-Abstract-Conference.html.

Yao, Andrew Chi-Chih (1982). "Theory and Applications of Trapdoor Functions (Extended Abstract)". In: *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*. IEEE Computer Society, pp. 80–91. DOI: 10.1109/SFCS.1982.45. URL: https://doi.org/10.1109/SFCS.1982.45.

Yovel, Yossi and Oded Rechavi (2023). "AI and the Doctor Dolittle challenge". In: *Current Biology* 33.15, R783–R787.

Zák, Stanislav (1983). "A Turing Machine Time Hierarchy". In: *Theoret. Comput. Sci.* 26, pp. 327–333. URL: https://doi.org/10.1016/0304-3975(83)90015-4.

Zhang, Jiaheng, Tianyi Liu, Weijie Wang, Yinuo Zhang, Dawn Song, Xiang Xie, and Yupeng Zhang (2021). "Doubly Efficient Interactive Proofs for General Arithmetic Circuits with Linear Prover Time". In: *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*. Ed. by Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi. ACM, pp. 159–177. DOI: 10.1145/3460120.3484767. URL: https://doi.org/10.1145/3460120.3484767.

Zhang, Lue, Hai-Ning Huang, Li Yin, Bao-Qi Li, Di Wu, Hao-Ran Liu, Xi-Feng Li, and Yong-Le Xie (2022). "Dolphin vocal sound generation via deep WaveGAN". In: *Journal of Electronic Science and Technology* 20.3, p. 100171.

Zhang, Sarah J., Samuel Florin, Ariel N. Lee, Eamon Niknafs, Andrei Marginean, Annie Wang, Keith Tyser, Zad Chin, Yann Hicke, Nikhil Singh, Madeleine Udell, Yoon Kim, Tonio Buonassisi, Armando Solar-Lezama, and Iddo Drori (2023). "Exploring the MIT Mathematics and EECS Curriculum Using Large Language Models". In: *CoRR*

abs/2306.08997. DOI: 10.48550/arXiv.2306.08997. arXiv: 2306.08997. URL: https://doi.org/10.48550/arXiv.2306.08997.

Zion, Rotem Ben, Boaz Carmeli, Orr Paradise, and Yonatan Belinkov (2024). "Semantics and Spatiality of Emergent Communication". In: *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*. Ed. by Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang. URL: http://papers.nips.cc/paper_files/paper/2024/hash/c6f83c27a2223d817f9f1ade48d281a2-Abstract-Conference.html.

Zuberbühler, Klaus (2020). "Syntax and compositionality in animal communication". In: *Philosophical Transactions of the Royal Society B* 375.1789, p. 20190062.

Appendix A

Related Work

A.1 Work related to Chapter 1

There have been several works that construct PCPs with certain structural properties and then used these PCPs to construct "hard" objects. We first discuss some of the previous results and compare them to ours, then talk about subsequent works which are based on our construction of rectangular PCPs.

PCPs with structured queries. We view Theorem 1.3 as continuing a line of work that explores the connection between the randomness of a PCP and the structure of its queries. A prominent advance in this direction is the work of Ben-Sasson and Viola (2014). They constructed short and efficient PCPs in which queries are a function of the input and a simple projection of the randomness (namely, a 1-*local* function: for a fixed input, each bit in each query location is a fixed bit of the randomness or its negation).[1] Although the query structure in the PCPs due to Ben-Sasson and Viola (2014) (and follow-up by Viola (2020)) are very simple, it is unclear whether these PCPs are almost-rectangular or smooth, that is, whether it has the two main properties of our construction that are used for its application to rigid matrices.

In a different direction, Feige and Jozeph (2012) constructed PCPs in which the queries depend only on the randomness but not on the input. Recently, Austrin, Brown-Cohen, and Håstad (2021) improved this result to have optimal soundness error for certain verification predicates such as 3SAT and 3LIN.

Circuit Lower Bounds from Algorithms. The maxim "hard claims have complex proofs" is inspired by a result of Williams (2016), showing that witnesses for $\mathsf{NTIME}(2^n) \setminus$

[1] Interestingly, the construction of Ben-Sasson and Viola (2014) is also an adaptation of a PCP from Ben-Sasson et al. (2005), but not the same one as in our work. Namely, we build upon the Reed–Muller based PCP of Ben-Sasson et al. (2005), whereas Ben-Sasson and Viola (2014) build upon the Reed–Solomon based PCP in the same paper.

$\mathsf{NTIME}(2^n/n)$ cannot be truth-tables of certain small-size low-depth circuits (specifically, ACC^0 circuits). That work is a part of Williams's algorithmic approach to circuit lower bounds originating in the work of Williams (2013) and Williams (2014). Roughly speaking, Williams's framework shows how to obtain lower bounds against a certain circuit class by designing non-trivial (i.e., better than exhaustive search) SAT algorithms for circuits in the class. Williams (2013) also observed the usefulness of PCPs within this framework: using PCPs, one can obtain circuit lower bounds from any non-trivial derandomization.[2] Santhanam and Williams (2014), Ben-Sasson and Viola (2014), and Chen and Williams (2019) further explored and tightened this connection. In this light, the overall proof strategy of Alman and Chen (2022) can be seen as a surprising instantiation of Williams's framework for average-case hardness of the computational model of low-rank matrices.

Applications to Probabilistic Degree. Viola (2020) showed[3] the existence of functions on n variables in $\mathbb{E}^{\mathsf{NP}}$ with approximate probabilistic degree $\Omega(n/\log^2 n)$ over $\mathcal{F}_2$, for infinitely many $n \in \mathbb{N}$. Using the known relation between matrix rigidity and approximate rank (see Alman and Chen, 2022, Proposition 7.5), Theorem 1.2 implies a similar lower bound on the approximate probabilistic degree. Also, in Viola (2020) suggested that improved PCP constructions with better structural properties (query-wise) may yield improved matrix rigidity bounds.

Subsequent work using rectangular PCPs. Since the initial publication of this work (Bhangale et al., 2020), the rectangular PCPs constructed herein were used in several works obtaining better rigid matrices. We describe these subsequent works below, noting that all of them rely on the rectangular PCPs of Theorem 1.3 as a building block.

One limitation of Theorem 1.2 is that it offers $N \times N$ rigid matrices only for *infinitely many* N, as opposed to *all* N. Chen et al. (2020) removed this limitation, giving an FNP construction that, for all N, on input 1^N outputs an $N \times N$ matrix whose rigidity is comparable to Theorem 1.2 in both rank and distance.[4]

Later, Huang and Viola (2021) gave an infinitely-often FNP construction that improves on the distance parameter of Theorem 1.2: For any $\varepsilon \in (0,1)$, rigidity for rank $2^{\log^{1-\varepsilon} N}$ is attained with distance $(1/2 - \exp(-\log^{\varepsilon/2} N)) \cdot N^2$, as opposed to $\delta \cdot N^2$ for a small constant $\delta > 0$ in Theorem 1.2. More precisely, for any $k < \Theta(\sqrt{\log N})$ and $\rho < 2^{\log N/\Omega(k \cdot (\log\log N + k))}$, their construction outputs $N \times N$ matrices that are rigid for rank ρ and distance $(1/2 - \exp(-k)) \cdot N^2$, infinitely often.

[2] More precisely, from any non-trivial deterministic estimation of the acceptance probability of a circuit, up to a constant additive error.

[3] We note that while Viola's work appeared in the public domain prior to ours, the two results (ours and Viola's) were obtained independently of each other.

[4] Chen et al. (2020) state that this construction works for *almost all* N, i.e., for all but finitely many N. As is common in complexity theory, one can hard-wire outputs for these finitely many N to obtain a construction that works for all N.

Independently of the result of Huang and Viola (2021), Chen and Lyu (2021) improved the distance even further to $(1/2 - \exp(-\log^{2\varepsilon/3} N)) \cdot N^2$; furthermore, like the result of Chen et al. (2020), this construction offers $N \times N$ rigid matrices for *all* N.

Lastly, Ren, Santhanam, and Wang (2022) construct rectangular PCPs *of proximity* and use them to obtain $\mathsf{FP}^{\mathsf{NP}}$ algorithms for the range avoidance problem for De Morgan formulas

A.2 Work related to Chapter 2

Chapter 2 is situated at the intersection of machine learning (ML) and Interactive Proof systems (IPs). We briefly discuss recent relevant work from these literatures.

ML and IPs. IPs have found numerous applications in ML towards a diverse set of goals. Anil et al. (2021) introduce Prover–Verifier Games (PVGs), a game-theoretic framework for learned provers and learned verifiers. Since our paper initially appeared, PVGs were further investigated in at least two subsequent works: Hammond and Adam-Day (2024) study multi-prover and Zero Knowledge variants of PVGs. Additionally, Kirchner et al. (2024) successfully utilize PVGs towards obtaining human-legible outputs from LLMs. Notably, they require a relaxed completeness guarantee of their learned proof system—this requirement is the same as our Definition 2.4 of Self-Proving models.

Beyond PVGs, Wäldchen et al. (2024) cast the problem of model interpretability as a Prover–Verifier interaction between a learned feature selector and a learned feature classifier. Debate systems (Condon et al., 1995), a multiprover variant of IPs, were considered for aligning models with human values (Irving, Christiano, and Amodei, 2018; Brown-Cohen, Irving, and Piliouras, 2023). In such Debate systems, two competing models are each given an alleged answer $y \neq y'$, and attempt to prove the correctness of their answer to a (human or learned) judge. Lastly, Murty, Paradise, and Sharma (2023) define Pseudointelligence: a model learner L_M and an evaluator learner L_E are each given samples from a ground-truth; L_M learns a model of the ground-truth, while L_E learns an evaluator of such models; the learned evaluator then attempts to distinguish between the learned model and the ground-truth in a Turing Test-like interaction.

All of these works consider *learned verifiers*, whereas our work focuses on training models that interact with a manually-defined verifier. More related in this regard is IP-PAC (Goldwasser et al., 2021), in which a learner proves that she learned a model that is Probably Approximately Correct (Valiant, 1984). We, however, consider *models* that prove their own correctness on a *per-input basis*, rather than *learners* that prove *average-case correctness* of a model.

Models that generate formal proofs. Self-Proving models are verified by an algorithm with formal completeness and soundness guarantees (see Definition 2.2). In this sense, Self-Proving models generate a formal proof of the correctness of their output. Several works propose specialized models that generate formal proofs.

AlphaGeometry (Trinh et al., 2024) is capable of formally proving olympiad-level geometry problems; Others have trained models to produce proofs in Gransden, Walkinshaw, and Raman (2015) and Polu and Sutskever (2020) and others train models to produce proofs in Coq (Gransden, Walkinshaw, and Raman, 2015), Metamath (Polu and Sutskever, 2020), Lean (Yang et al., 2023), or manually-defined deduction rules (Tafjord, Dalvi, and Clark, 2020); FunSearch (Romera-Paredes et al., 2024) evolves LLM-generated programs by systematically evaluating their correctness. Indeed, all of these can be cast as Self-Proving models developed for *specific proof systems*. Meanwhile, this work defines and studies the class of such models *in general*. Several works (e.g. Welleck et al. 2022) consider models that generate natural language proofs or explanations, which are fundamentally different from formal proofs (or provers) verified by an algorithm.

Training on intermediate steps. Chain-of-Though (CoT, Wei et al. 2022) refers to additional supervision on a model in the form of intermediate reasoning steps. CoT is known to improve model performance whether included in-context (Wei et al., 2022) or in the training phase itself (Yang et al., 2022). Transcript Learning (TL, Section 2.2.1) can be viewed as training the model on a Chain-of-Thought induced by the interaction of a verifier and an honest prover (Definition 2.2).

To complete the analogy, let us adopt the terminology of Uesato et al. (2022), who consider *outcome supervision* and *process supervision*. In our case, the *outcome* is the decision of the verifier, and the *process* is the interaction between the verifier and the model. Thus, Reinforcement Learning from Verifier Feedback (RLVF, Section 2.2.2) is outcome-supervised while TL is process-supervised. In a recent work, Lightman et al. (2024) find that process-supervised transformers outperform outcome-supervised ones on the MATH dataset (Hendrycks et al., 2021).

Transformers for arithmetic. In Section 2.3 we train and evaluate Self-Proving transformers to generate the GCD of two integers and prove its correctness to a verifier. These experiments leverage a long line of work on neural models for arithmetic tasks originating with Siu and Roychowdhury (1992), and in particular modular arithmetic, which is known to be challenging (Palamas, 2017). Of particular relevance is the recent paper of Charton (2024), who trains transformers to generate the GCD—without a proof of correctness. We benefit from conclusions suggested in their work and start from a similar (scaled-down) experimental setup. Our main challenge (obtaining *Self-Proving* models) is overcome by introducing Annotated Transcript Learning (ATL).

We conduct ablation experiments to find two deciding factors in ATL. First, we study the effect of the amount of annotation given in the form of intermediate steps (Lee et al., 2024), which is related to autoregressive length complexity (Malach, 2023). Second, we characterize ATL efficacy in terms of an algebraic property of the tokenization scheme (cf. Nogueira, Jiang, and Lin 2021; Charton 2022; Charton 2024).

A.3 Work related to Chapter 4

Project CETI. The sperm whale data collection effort began with a longitudinal dataset from a community of whales off the coast of Dominica that revealed interesting communication findings, such as dialects and vocal clans (Gero, Whitehead, and Rendell, 2016b). A recent effort by the Cetacean Translation Initiative (Project CETI) has been to collect custom-built passive bioacoustic arrays (installed in Fall 2022) covering a 20×20 kilometer area where these whale families reside (collecting over 3 TB/month) in tandem with on-whale robotic acoustic and video tags, underwater (robotic swimming fish) and aerial drones as well as other observation techniques in effort to augment rich contextual communication data. CETI's scientific team consists of specialists in machine learning, robotics, natural language processing, marine biology, linguistics, cryptography, signal processing and bio-acoustics. Andreas et al. (2022b) present CETI's initial scientific roadmap for understanding sperm whale communication, identifying the potential for unsupervised translation to be applied to whale communication. That roadmap suggests training a full generative LM for whale communication (often using trillions of bits for parameters (Brown et al., 2020; Chowdhery et al., 2022)). In contrast, our analysis suggests that the data requirements for translation may be similar to those of supervised translation, which is often several orders of magnitude smaller (Tran et al., 2021).

With this setting in mind, our requirements from source and target language are not symmetric: it would be unreasonable (and unnecessary) for our framework to assume that any sentence in the target language could be expressed in the source language. Put simply: whales need not understand what a smartphone is for us to gain some understanding of their communication. Also note, regarding domain gap, that some (although not all) knowledge can be inferred by training data from, e.g., online catalogs of hundreds of thousands of marine species (Ahyong et al., 2022)). Of course, there are also data-collection and transcription challenges, a challenge also present in the setting of *low-resource* (human) language translation (Ranathunga et al., 2021). While these challenges are outside the scope of this paper, our theoretical bounds on the data requirements may inform how much and what types of data are collected. For instance, it is less expensive to acquire textual data alone than both textual and video data. Therefore, if it is believed that an adequate translation is statistically possible using textual data alone, then greater effort may be placed on collecting this data and on UMT algorithms.

Unsupervised translation. In unsupervised machine translation (Ravi and Knight, 2011), a translator between two languages is learned based only on monolingual corpora from each language. A body of work on UMT uses neural networks (Miceli Barone, 2016; Lample et al., 2018a; Artetxe, Labaka, and Agirre, 2019; Lample et al., 2018b; Song et al., 2019) or statistical methods (Lample et al., 2018c; Artetxe, Labaka, and Agirre, 2018) for this task. Empirical evaluation of UMT found that it is outperformed by supervised machine translation, even when UMT is trained on several orders of magnitude more data (Marchisio, Duh, and Koehn, 2020; Kim, Graça, and Ney, 2020). Among the key barriers for UMT iden-

tified in these evaluations are the domain gap and the data gap, and recent works propose techniques for bridging these gaps (Edmiston, Keung, and Smith, 2022; He et al., 2022). Our theory suggests that sample complexity should remain roughly the same between the supervised and unsupervised settings, barring computational constraints. This, we hope, will encourage practitioners to bridge the remaining gaps.

Language models (LMs). In recent years, LMs such as GPT (Brown et al., 2020), BERT (Devlin et al., 2018) and PaLM (Chowdhery et al., 2022) were shown to achieve state-of-the-art performance on many tasks in natural language processing (NLP) such as text generation, summarization, or (supervised) MT. These models are indeed large, with hundreds of billions of parameters, and are pre-trained on hundreds of billions of tokens.

LMs are useful for machine translation in a variety of ways (e.g. Brants et al. 2007; Han et al. 2021). Of particular relevance are empirical works that use target LMs as priors to improve machine translation (Lynum et al., 2012; Baziotis, Haddow, and Birch, 2020). To our knowledge, our work is the first *theoretical* work formally proving error bounds for prior-assisted translation. Section 4.7 discusses the use of LMs to establish priors for translation.

Goal-oriented communication. It is interesting to contrast our work with the work on goal-oriented communication, which was introduced by Juba and Sudan (2008) and extended by Goldreich, Juba, and Sudan (2012). They study the setting of two communicating parties (one of which is trying to achieve a verifiable goal) using each a language completely unknown to the other. They put forward a theory of goal-oriented communication, where communication is not an end in itself, but rather a means to achieving some goals of the communicating parties. Focusing on goals provides a way to address "misunderstanding" during communication, as in when one can verify whether the goal is (or is not) achieved. Their theory shows how to overcome any initial misunderstanding between parties towards achieving a given goal. Our setting is different: Informally, rather than be a participant in a communication with someone speaking a different language, we wish to translate communications between two external parties speaking in a language unknown to us and there is no verifiable goal to aid us in this process.

Subgraph isomoprhism. For simplicity, we model the knowledge graphs of Section 4.5 as a pair of correlated Erdős–Rényi (ER) graphs. The computational problem of identifying a subgraph of an ER graph has been studied by Babai, Erdo˝s, and Selkow (1980), Livi and Rizzi (2013), and Hu, Wang, and Yu (2022). In particular, Hu, Wang, and Yu (2022) consider a model in which two correlated graphs P, T are derived from a "parent graph" G by independently deleting rows and edges G, and then applying a permutation π^* to the vertices of T. Although their model differs from our knowledge graph model,[5] they propose efficient algorithms for recovering the latent permutation π^* and provide an empirical

[5]In the knowledge graph (a) the vertices of T are *always* a subset of the vertices of P, and (b) the deleted vertices are fixed rather than randomly chosen

evaluation on synthetic and real-world data. Given the similarity between our models, it would be interesting to see if their algorithm can be adapted to our setting, which would nicely complement our formally-proven-yet-inefficient algorithm.

A.4 Work related to Chapter 5

Audio Generation. The vast majority of studies on deep generative audio models focus on human speech or music (e.g. Oord et al. 2016; Dong et al. 2018; Dhariwal et al. 2020; Lakhotia et al. 2021; Agostinelli et al. 2023). Some works are dedicated to generating the vocalizations of animals (bioacoustics) such as birds (Bhatia and Kinnunen, 2022; Guei et al., 2024), mice (Reilly et al., 2023), cetaceans (Bergler et al., 2022; Zhang et al., 2022; Honghui and Lanhao, 2022; Kim et al., 2024), and in particular sperm whales (Beguš, Leban, and Gero, 2023; Kopets et al., 2024). However, to our knowledge, all techniques for bioacoustic generation are based on generative adversarial networks (GANs). Unlike our transformer-based WhAM, GANs do not allow for conditioning on context in the form of an audio prompt. We emphasize that WhAM enables translation of input sounds into the *acoustic style* of sperm whale vocalizations, operating purely at the signal level. This is distinct from *semantic* translation between communication systems, which remains a far more ambitious goal requiring a deep understanding of animal cognition and communication (e.g. Goldwasser et al. 2023; Yovel and Rechavi 2023; Amphaeris et al. 2023).

Animal Vocalization Modeling. Deep learning techniques have been applied towards other, non-generative, ends in bioacoustics research. Learned audio representations have been used for species recognition (Chen et al., 2014; Hafemann, Oliveira, and Cavalin, 2014; Xu et al., 2019; Kahl et al., 2021; Xie et al., 2023) and automatic annotation (i.e., vocalization detection and classification) of bioacoustic data (Bergler et al., 2019; Coffey, Marx, and Neumaier, 2019; Bermant et al., 2019; Premoli et al., 2021). AVES (Hagiwara, 2023) utilizes HuBERT's (Hsu et al., 2021) self-supervised learning framework towards state-of-the-art performance in species classification and detection tasks. While AVES demonstrates the power of learned audio representations, its encoder-only architecture limits it to analysis tasks, contrasting with WhAM's generative capabilities. As we show in Section 5.3.3, while AVES outperforms WhAM on classification tasks as expected given its specialized design, WhAM still learns meaningful representations as a byproduct of its generative training, outperforming baseline approaches despite having a different primary objective.

Sperm whale communication. Understanding sperm whale communication has been a central challenge in marine biology for over six decades (Backus and Schevill 1966; Watkins and Schevill 1977; Whitehead and Weilgart 1991; Andreas et al. 2022a; see also Section 5.1). Recent computational approaches have focused on analyzing click timing patterns within codas and do not directly address the acoustic properties of individual clicks within codas (Sharma et al., 2024a; Leitao et al., 2024; Sharma et al., 2024b). WhAM extends this

computational trajectory by enabling systematic manipulation of click acoustics, potentially allowing a quantitative analysis of acoustic variations between clan dialects and investigation of features that make codas recognizable. While WhAM's synthetic codas may not yet match the quality needed for playback experiments, WhAM represents progress towards stimuli generation in a responsible behavioral study (Tyack 1983; Deecke 2006; King and Jensen 2023; see also Appendix B).

A.5 Bibliographic notes

This dissertation comprises five research chapters, each based on published or submitted work:

Chapter 2: Amey Bhangale, Prahladh Harsha, Orr Paradise, and Avishay Tal (2024). "Rigid Matrices from Rectangular PCPs". In: *SIAM J. Comput.* 53.2, pp. 480–523. DOI: 10.1137/22M1495597. URL: https://doi.org/10.1137/22m1495597. Previously appeared as Amey Bhangale, Prahladh Harsha, Orr Paradise, and Avishay Tal (2020). "Rigid matrices from rectangular PCPs or Hard Claims have Complex Proofs". In: *Proc. 61st IEEE Symp. on Foundations of Comp. Science (FOCS)*, pp. 858–869. eprint: 2005.03123. URL: https://doi.org/10.1109/FOCS46700.2020.00084.

Chapter 3: Noga Amit, Shafi Goldwasser, Orr Paradise, and Guy N. Rothblum (2024). "Models That Prove Their Own Correctness". In: *CoRR* abs/2405.15722. DOI: 10.48550/ARXIV.2405.15722. arXiv: 2405.15722. URL: https://doi.org/10.48550/arXiv.2405.15722. In submission.

Chapter 4: Shikhar Murty, Orr Paradise, and Pratyusha Sharma (2023). "Pseudointelligence: A Unifying Lens on Language Model Evaluation". In: *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*. Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. Association for Computational Linguistics, pp. 7284–7290. DOI: 10.18653/V1/2023.FINDINGS-EMNLP.485. URL: https://doi.org/10.18653/v1/2023.findings-emnlp.485.

Chapter 5: Shafi Goldwasser, David F. Gruber, Adam Tauman Kalai, and Orr Paradise (2023). "A Theory of Unsupervised Translation Motivated by Understanding Animal Communication". In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine.

Chapter 6: Orr Paradise, Liangyuan Chen, Pranav Muralikrishnan, Hugo Flores García, Bryan Pardo, Roee Diamant, David Gruber, Shane Gero, and Shafi Goldwasser (May 2025). "Towards A Translative Model of Sperm Whale Vocalization". In submission.

During my doctoral studies, I contributed to several additional publications not included in this dissertation. These works either explored directions tangential to the central themes of Proofs and Translation, or were collaborations in which I was not the primary contributor:

- Sam Gunn, Doseok Jang, Orr Paradise, Lucas Spangher, and Costas J. Spanos (2022). "Adversarial poisoning attacks on reinforcement learning-driven energy pricing". In: *Proceedings of the 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation, BuildSys 2022, Boston, Massachusetts, November 9-10, 2022*. Ed. by Jorge Ortiz. ACM, pp. 262–265. DOI: 10.1145/3563357.3564075. URL: https://doi.org/10.1145/3563357.3564075.

- Micah Carroll, Orr Paradise, Jessy Lin, Raluca Georgescu, Mingfei Sun, David Bignell, Stephanie Milani, Katja Hofmann, Matthew J. Hausknecht, Anca D. Dragan, and Sam Devlin (2022). "Uni[MASK]: Unified Inference in Sequential Decision Problems". In: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*. Ed. by Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh. URL: http://papers.nips.cc/paper_files/paper/2022/hash/e58fa6a7b431e634e0fd125e225ad10c-Abstract-Conference.html.

- Daniel T. Fokum, Zaria Chen Shui, Kerene Wright, Orr Paradise, Gunjan Mansingh, and Daniel Coore (2024). "A High School Camp on Algorithms and Coding in a Small Island Developing State". In: *Proceedings of the 55th ACM Technical Symposium on Computer Science Education, SIGCSE 2024, Volume 1, Portland, OR, USA, March 20-23, 2024*. Ed. by Ben Stephenson, Jeffrey A. Stone, Lina Battestilli, Samuel A. Rebelsky, and Libby Shoop. ACM, pp. 352–358. DOI: 10.1145/3626252.3630762. URL: https://doi.org/10.1145/3626252.3630762.

- James Bartusek, Thiago Bergamaschi, Seri Khoury, Saachi Mutreja, and Orr Paradise (2024). "On the Communication Complexity of Secure Multi-Party Computation With Aborts". In: *Proceedings of the 43rd ACM Symposium on Principles of Distributed Computing, PODC 2024, Nantes, France, June 17-21, 2024*. Ed. by Ran Gelles, Dennis Olivetti, and Petr Kuznetsov. ACM, pp. 480–491. DOI: 10.1145/3662158.3662815. URL: https://doi.org/10.1145/3662158.3662815.

- Rotem Ben Zion, Boaz Carmeli, Orr Paradise, and Yonatan Belinkov (2024). "Semantics and Spatiality of Emergent Communication". In: *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*. Ed. by Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang. URL: http://papers.nips.cc/paper_files/paper/2024/hash/c6f83c27a2223d817f9f1ade48d281a2-Abstract-Conference.html.

- Ido Levy, Orr Paradise, Boaz Carmeli, Ron Meir, Shafi Goldwasser, and Yonatan Belinkov (2025). "Unsupervised Translation of Emergent Communication". In: *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*. ed. by Toby Walsh, Julie Shah, and Zico Kolter. AAAI Press, pp. 23231–23239. DOI: 10.1609/AAAI.V39I22.34489. URL: https://doi.org/10.1609/aaai.v39i22.34489.

- Ferhat Erata, Orr Paradise, Timos Antonopoulos, ThanhVu Nguyen, Shafi Goldwasser, and Ruzica Piskac (2024). "Learning Randomized Reductions and Program Properties". In: *CoRR* abs/2412.18134. DOI: 10.48550/ARXIV.2412.18134. arXiv: 2412.18134. URL: https://doi.org/10.48550/arXiv.2412.18134. In submission.

Remark. *The publications cited throughout this dissertation follow their respective field conventions for author ordering: theoretical publications traditionally list authors alphabetically, while others may list authors according to order of contribution.*

Appendix B

Ethics and Impact

Chapter 2 proposes a theoretically-grounded approach to enhancing trust in learned models. By ensuring that models not only generate outputs but also prove their correctness to a verification algorithm, we tackle fundamental issues of trust and accountability in machine learning.

Self-Proving models build trust between models and users by offering formal worst-case soundness guarantees. This is particularly beneficial in high-stakes applications, such as healthcare and finance, where incorrect outputs can have severe consequences. The ability to verify correctness on a per-instance basis helps prevent potentially harmful decisions. It allows any user to decide for herself whether she trusts a particular output generated by the model, rather than relying on average-case guarantees (e.g., high scores on benchmarks as reported by the model's developer).

Furthermore, Self-Proving models promote accountability by allowing stakeholders to independently verify the correctness of a model's outputs. In particular, lawmakers and regulators could require models used in sensitive settings to be Self-Proving.

With that said, Self-Proving models also introduce challenges which must be addressed. First, we expect Self-Proving models to be harder to learn (in practice), which may limit their applicability in more complex tasks. Second, as with any learned model, Self-Proving models could be used in harmful ways; developers of a model (and verification algorithm) must consider the impact of their systems in the specific context in which they are deployed (Suresh et al., 2023). In other words, the fact that a Self-Proving model's outputs are provably correct does not mean that these outputs were ought to be generated in the first place.

Chapter 3 is focused on motivating and defining pseudointelligence, as well as demonstrating its potential use for unifying and analysing LLM evaluation. Deeper analyses, such as provable bounds comparing model and evaluator sample complexities (m vs. n), are left for future work.

The impact of large language models extends far beyond their alleged (pseudo-)intelligence Bommasani et al., 2021. Pseudointelligence does not, for example, correspond to an ability to respond to queries in an ethical or responsible manner. In general, psueodintelligence is

concerned with the distinguishing ability of a class of evaluators, but does not consider the usage of a model in a real-world context which may not conform to this class (cf. Mitchell et al., 2019; Suresh et al., 2023). Finally, like all abstract definitions, it must not be used as a *rubber stamp*; that is, it cannot replace a case-by-case assessment of potential impacts of models prior to their deployment.

Chapter 5 has potential implications for both scientific understanding and conservation efforts. Historically, advances in understanding cetacean communication have played crucial roles in conservation—notably, the discovery of humpback whale song by Payne and McVay (1971) contributed significantly to public awareness and the subsequent "Save the Whales" movement (Campagna and Guevara, 2022; Comuzzo, 2023). While we maintain that sperm whales deserve protection regardless of our ability to understand their communication, we recognize that deeper scientific understanding often catalyzes public engagement with conservation efforts.

WhAM's capabilities might naturally suggest applications in behavioral experiments through playback studies. This is particularly tempting given that sperm whales often produce codas simultaneously—a behavior that our bidirectional model could theoretically capture by conditioning on one whale's clicks while generating the overlapping clicks of another. However, we strongly caution against such applications at this stage. Without a deeper understanding of coda semantics and functionality, playback experiments using synthetic vocalizations could have unintended and potentially harmful consequences for these social marine mammals. Instead, we propose that this work demonstrates the potential of learning from passive acoustic observation—studying these remarkable animals through careful listening rather than active intervention. With this approach, this chapter could potentially play a role in assisting efforts to reinforce existing protections or create new legal protections for whales (Rodríguez-Garavito et al., 2025).

As noted in Section 5.5, the methodological framework of Chapter 5 could extend beyond sperm whales, potentially benefiting research on other marine mammals and, more broadly, any species that communicates acoustically. This scalability is particularly relevant as biodiversity monitoring becomes increasingly critical in the face of environmental changes. However, our experience underscores the importance of deep collaboration with domain experts—the success of this work relied on guidance from marine biologists and acousticians with decades of experience studying sperm whales. We encourage future work in this direction to similarly prioritize partnerships with species-specific domain experts, as their insights are crucial for both model development and responsible deployment.