Steering How Deep Neural Networks Generalize

Katie Kang

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2025-91
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2025/EECS-2025-91.html

May 16, 2025

Steering How Deep Neural Networks Generalize

by

Katie Kang

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 Sergey Levine, Chair
Professor Claire Tomlin
Professor Aviral Kumar
Professor Jacob Steinhardt

Spring 2025

The dissertation of Katie Kang, titled Steering How Deep Neural Networks Generalize, is approved:

Chair _____ Date _____

_____ Date _____

_____ Date _____

_____ Date _____

University of California, Berkeley

Steering How Deep Neural Networks Generalize

Abstract

Steering How Deep Neural Networks Generalize

by

Katie Kang

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Sergey Levine, Chair

Deep learning models, particularly Large Language Models (LLMs), have achieved remarkable capabilities, yet their reliability is often hindered by a lack of understanding regarding their generalization to unseen data. Failures when encountering novel inputs, such as factual inaccuracies or deviations from instructions, can lead to safety vulnerabilities in real-world applications. This dissertation confronts this challenge by investigating how different aspects of the training process influence generalization and extrapolation in deep neural networks, with a specific focus on LLMs. The core objective is twofold: first, to characterize how elements of the learning recipe shape model behavior on both in-distribution and out-of-distribution data, and second, to develop strategies for steering generalization to enhance performance and robustness on unseen examples.

A model's generalization behavior varies depending on the training recipe and evaluation data. This thesis studies this behavior from different angles, progressing from standard deep neural networks, often optimized in a single stage, to modern LLMs, which typically undergo multiple stages of pretraining and finetuning. First, we study the extrapolation tendencies of standard deep networks when presented with inputs distributionally different from their training data. Challenging the assumption of erratic out-of-distribution behavior, this work demonstrates that these networks often exhibit structured and predictable extrapolation patterns, tending towards constant outputs that can be systematically linked to properties of the training data and the loss function used. Subsequently, this thesis examines hallucination in LLMs, finding that unfamiliar examples in finetuning data critically influence factually incorrect outputs and that modifying their supervision can mitigate these errors. Finally, the work explores the acquisition of generalizable mathematical reasoning skills, revealing that learning dynamics, particularly accuracy achieved before memorizing training steps, strongly correlate with the model's performance on heldout examples. Collectively, these investigations offer a more nuanced understanding of generalization, contributing towards the development of more predictable and reliable deep learning systems.

To my parents

Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, I want to express my deepest gratitude to my PhD advisors, Sergey Levine and Claire Tomlin. My research journey began as an undergraduate in Sergey's lab, arriving with no prior background in machine learning. Sergey has been an extraordinary mentor, teaching me everything from high-level research taste to a large variety of technical details. His depth of technical knowledge, unwavering dedication to research, and insightful guidance have been unparalleled, and I feel incredibly fortunate to have had the opportunity to be his student. When I started my PhD, I was interested in improving the safety and robustness of AI systems, and thought that control theory was an elegant and compelling avenue for tackling this challenge. I am very grateful to Claire for taking a chance on me as a student. Over the years, my research interests have evolved and taken various turns. Throughout this exploration, Claire has been a constant source of encouragement and support, fostering my desire to understand model behavior, even when such pursuits were not always productive or fruitful.

Next, I wish to express my sincere appreciation to my committee members, Aviral Kumar and Jacob Steinhardt, both of whom have been exceptional mentors. Aviral, a former member of our lab and someone I have always admired, has consistently been generous with his time and advice. I am particularly grateful for his knack for rescuing my projects when I found myself stuck. My collaboration with Jacob was instrumental in introducing me to new research areas, including AI safety and interpretability. I am also thankful to Jacob for welcoming me as an honorary member of his lab, allowing me to get to know a community interested in similar things, as well as being an extremely patient coach to our IM ultimate frisbee team.

I have also had the privilege of working with many talented research collaborators and mentors. A special thank you goes to Greg Kahn, my undergraduate research mentor, who taught me the ropes of research. I'd also like to thank Michael Janner, who mentored me during my first project in graduate school and patiently taught me the intricacies of getting RL algorithms to work. My gratitude extends to Eric Wallace, who introduced me to LLM research. Finally, I am grateful for Amrith Setlur, who has been my closest collaborator in graduate school; our shared excitement for the same research problems always make our collaborations a lot of fun.

I am incredibly grateful to the members of RAIL and BAIR for fostering such a vibrant and supportive community. It was here that I met some of my closest friends in graduate school. Special thanks to my cohort in RAIL – Dibya Ghosh, Colin Li, and Laura Smith – with whom I began this PhD journey during the crazy times of COVID and who are (mostly) graduating alongside me five years later. Thank you to Kevin Black and Homer Walke for always being down for blue bottle runs and fellow frisbee enthusiasts; to Ameesh Shah, for teaching me the Power of Friendship; and to Meena Jagadeesan, for our weekly weekend walks in North Berkeley. My thanks also extend to my roommates, Linnea Warburton, Daniel Klawson, and Yertay Zhiyenbayev, who made coming home each day a fun and

welcoming experience. Lastly, I'd like to thank Dibya Ghosh, for being there every step of the way.

Lastly, I would like to express my deepest gratitude to my parents for their unwavering support and encouragement throughout my PhD and life in general.

Chapter 1

Introduction

In recent years, deep learning and Large Language Models (LLMs) have experienced an unprecedented rate of advancement. However, their transformative capabilities often outpace our understanding of how they work. In particular, while LLMs excel at many complex tasks, their behavior beyond the scope of their training data remains poorly understood. Failures in generalization can manifest as factual inaccuracies, deviations from instructions, or unpredictable responses to novel inputs, which compromises their safety and reliability. This dissertation aims to take steps towards tackling this challenge by exploring how different characteristics of the training recipe governs generalization and extrapolation in deep neural networks, with a focus on LLMs.

At the heart of this challenge lies the gap between what models are optimized for and what we want them to be good at. Deep learning models are optimized to perform well on a finite set of training examples. Our true objective, however, is for these models to perform well on unseen data, including data similar to the training set (in-distribution or ID generalization) and data from related but different distributions (out-of-distribution or OOD generalization). Better performance on the training dataset does not necessarily translate to better performance on unseen examples. Thus, there is a need for a deeper understanding of the aspects of training which govern model behavior on unseen data. To this end, this thesis pursues two primary goals: 1) characterizing how elements of the learning recipe govern generalization, and 2) devising approaches for steering model generalization in order to improve model performance on unseen data.

Prior research has approached the generalization problem from a number of different perspectives. Classical learning theory offers bounds based on model complexity (e.g., VC-dimension [Vapnik and Chervonenkis, 2015, Blumer et al., 1989], Rademacher complexity [Bartlett and Mendelson, 2002, Chen, 2019]), though these often fail to explain the strong performance of highly overparameterized deep networks [Zhang et al., 2016, Neyshabur et al., 2017, Belkin et al., 2019, Nagarajan and Kolter, 2019b]. More recent work investigates the implicit regularization effects of optimization algorithms like SGD, suggesting they bias models towards generalizable solutions [Neyshabur et al., 2014, Hardt et al., 2016, Soudry et al., 2018]. Other studies highlight the roles of data augmentation [Zhang et al., 2017a,

Cubuk et al., 2019], explicit regularization (e.g. dropout [Srivastava et al., 2014, Gal and Ghahramani, 2016] and weight decay [Krogh and Hertz, 1991, McCandlish et al., 2018]), and architectural choices [He et al., 2016, Vaswani et al., 2017]. Research specifically on OOD generalization explores methods like domain adaptation [Ben-David et al., 2010, Gulrajani and Lopez-Paz, 2020] and invariant risk minimization [Arjovsky et al., 2019]. Within LLMs, studies often focus on empirical scaling laws [Kaplan et al., 2020, Hoffmann et al., 2022], probing internal representations [Belinkov and Glass, 2019, Tenney et al., 2019], understanding emergent abilities like in-context learning [Brown et al., 2020, Xie et al., 2021], and mitigating specific failures like hallucination [Ji et al., 2023, Shuster et al., 2021, Nakano et al., 2021]. While this body of work provides valuable empirical characterizations and benchmarks, there still remain many open questions regarding the fundamental mechanisms driving deep neural network and LLM generalization, and this thesis aims to contribute towards a deeper understanding of these mechanisms.

The first part of this thesis (Chapter 2) examines how deep neural networks behave when presented with inputs distributionally different from those encountered during training [Kang et al., 2023]. Challenging the common assumption that deep neural networks behave erratically and unpredictably when queried on OOD inputs, we present theoretical and empirical evidence demonstrating that the extrapolation patterns of deep networks can exhibit surprising structure and predictability. In particular, we observe that neural network predictions often tend towards a constant value as input data becomes increasingly OOD. Furthermore, we find that this constant value, which we call the optimal constant solution, can be systematically linked to properties of the training data distribution and the structure of the loss function. We present experiments showing that deep neural networks exhibit this behavior across different distributional shifts, loss functions, and architectures. We additionally propose an explanation for why this behavior happens, which we first validate empirically and then study theoretically in a simplified setting involving deep homogeneous networks. Finally, we show how one can leverage our insights in practice to enable neural networks to exhibit risk-sensitive decision-making behavior in the presence of OOD inputs. The analysis in this section offers a more nuanced understanding of OOD generalization in conventional deep neural networks, setting the stage for analyzing the more complex generalization behavior of large language models.

Building upon our understanding of extrapolation in conventional deep neural networks, the second part of this thesis (Chapter 3) delves into a prevalent failure mode in LLMs: the generation of plausible yet factually incorrect information, commonly referred to as "hallucination", when models encounter queries outside their scope of their knowledge [Kang et al., 2024b]. We specifically investigate how the finetuning process influences a model's tendency for hallucination on factual question-answering tasks. we find that unfamiliar examples in the models' finetuning data – those that introduce concepts beyond the pretrained model's scope of knowledge – are crucial in shaping these errors. In particular, we find that an LLM's hallucinated predictions tend to mirror the responses associated with its unfamiliar finetuning examples. This suggests that by modifying how unfamiliar finetuning examples are supervised, we can influence a model's responses to unfamiliar queries (e.g., say "I don't

know"). We empirically validate this observation in a series of controlled experiments, and further investigate reinforcement learning strategies for improving the factuality of long-form model generations. The analysis in this section offers insights into the mechanisms governing extrapolation in LLM question-answering, and provides practical strategies for improving the factuality of LLM responses.

Finally, the third part (Chapter 4) shifts focus from factual question answering to mathematical reasoning in LLMs [Kang et al., 2024a]. While LLMs must have encountered a specific fact in its training data in order to answer questions about it, they are able to answer completely novel math questions by knowing the rules of math and employing step-by-step reasoning. This part of the thesis studies the factors which influence how models learn generalizable behaviors for mathematical reasoning tasks. Our analysis reveals that a model's learning dynamics – changes in model behavior over the course of finetuning – provides insights into how such generalizable reasoning skills emerge. In particular, we find that a model's test accuracy can be effectively characterized by a training metric we call pre-memorization train accuracy: the accuracy of model samples on training queries before they begin to copy the exact reasoning steps from the training set. This observation allows us to connect a model's training behavior with its test generalization, thereby providing a tool for guiding targeted improvements to a model's training recipe. We focus on data curation as an example, and show that filtering training data using pre-memorization train accuracy can lead to significant improvements to a model's sample efficiency. The analysis in this section complements our findings from the previous section by shedding light on the mechanisms behind a model's generalization for reasoning-based capabilities.

Collectively, these studies aim to provide a more nuanced understanding of the factors governing how deep learning models generalize and behave on unseen examples. By analyzing extrapolation tendencies (Chapter 2), the influence of specific finetuning data on hallucination (Chapter 3), and the learning dynamics underlying the acquisition of reasoning skills (Chapter 4), this dissertation seeks to contribute insights towards the design and development of more predictable and robust machine learning systems.

Chapter 2

Deep Neural Network Extrapolation

2.1 Introduction

The prevailing belief in machine learning posits that deep neural networks behave erratically when presented with out-of-distribution (OOD) inputs, often yielding predictions that are not only incorrect, but incorrect with *high confidence* [Guo et al., 2017, Nguyen et al., 2015]. However, there is some evidence which seemingly contradicts this conventional wisdom – for example, Hendrycks and Gimpel [2016] show that the softmax probabilities outputted by neural network classifiers actually tend to be *less confident* on OOD inputs, making them surprisingly effective OOD detectors. In our work, we find that this softmax behavior may be reflective of a more general pattern in the way neural networks extrapolate: as inputs diverge further from the training distribution, a neural network's predictions often converge towards a *fixed* constant value. Moreover, this constant value often approximates the best prediction the network can produce without observing any inputs, which we refer to as the optimal constant solution (OCS). We call this the "reversion to the OCS" hypothesis: **Neural networks predictions on high-dimensional OOD inputs tend to revert towards the optimal constant solution**.

In classification, the OCS corresponds to the marginal distribution of the training labels, typically a high-entropy distribution. Therefore, our hypothesis posits that classifier outputs should become higher-entropy as the input distribution becomes more OOD, which is consistent with the findings in Hendrycks and Gimpel [2016]. Beyond classification, to the best of our knowledge, we are the first to present and provide evidence for the "reversion to the OCS" hypothesis in its full generality. Our experiments show that the amount of distributional shift correlates strongly with the distance between model outputs and the OCS across 8 datasets, including both vision and NLP domains, 3 loss functions, and for both CNNS and transformers.

Having made this observation, we set out to understand why neural networks have a tendency to behave this way. Our empirical analysis reveals that the feature representations corresponding to OOD inputs tend to have smaller norms than those of in-distribution inputs,

leading to less signal being propagated from the input. As a result, neural network outputs from OOD inputs tend to be dominated by the input-independent parts of the network (e.g., bias vectors at each layer), which we observe to often map closely to the OCS. We also theoretically analyze the extrapolation behavior of deep homogeneous networks with ReLU activations, and derived evidence which supports this mechanism in the simplified setting.

Lastly, we leverage our observations to propose a simple strategy to enable risk-sensitive decision-making in the face of OOD inputs. The OCS can be viewed as a "backup default output" to which the neural network reverts when it encounters novel inputs. If we design the loss function such that the OCS aligns with the desirable cautious behavior as dictated by the decision-making problem, then the neural network model will automatically produce cautious decisions when its inputs are OOD. We describe a way to enable this alignment, and empirically demonstrate that this simple strategy can yield surprisingly good results in OOD selective classification.

In summary, our key contributions are as follows. First, we present the observation that neural networks often exhibit a predictable pattern of extrapolation towards the OCS, and empirically illustrate this phenomenon for 8 datasets with different distribution shifts, 3 loss functions, and both CNNs and transformers. Second, we provide both empirical and theoretical analyses to better understand the mechanisms that lead to this phenomenon. Finally, we make use of these insights to propose a simple strategy for enabling cautious decision-making in face of OOD inputs. Although we do not yet have a complete characterization of precisely when, and to what extent, we can rely on "reversion to the OCS" to occur, we hope our observations will prompt further investigation into this phenomenon.

2.2 Related Work

A large body of prior works have studied various properties of neural network extrapolation. One line of work focuses on the failure modes of neural networks when presented with OOD inputs, such as poor generalization and overconfidence [Torralba and Efros, 2011, Gulrajani and Lopez-Paz, 2020, Recht et al., 2019, Ben-David et al., 2006, Koh et al., 2021]. Other works have noted that neural networks are ineffective in capturing epistemic uncertainty in their predictions [Ovadia et al., 2019, Lakshminarayanan et al., 2017, Nalisnick et al., 2018, Guo et al., 2017, Gal and Ghahramani, 2016], and that a number of techniques can manipulate neural networks to produce incorrect predictions with high confidence [Szegedy et al., 2013, Nguyen et al., 2015, Papernot et al., 2016, Hein et al., 2019]. However, Hendrycks et al. [2018] observed that neural networks assign lower maximum softmax probabilities to OOD than to in-distribution point, meaning neural networks may actually exhibit less confidence on OOD inputs. Our work supports this observation, while further generalizing it to arbitrary loss functions. Other lines of research have explored OOD detection via the norm of the learned features [Sun et al., 2022, Tack et al., 2020], the influence of architectural decisions on generalization [Xu et al., 2020, Yehudai et al., 2021, Cohen-Karlik et al., 2022, Wu et al., 2022], the relationship between in-distribution and OOD performance [Miller et al., 2021,

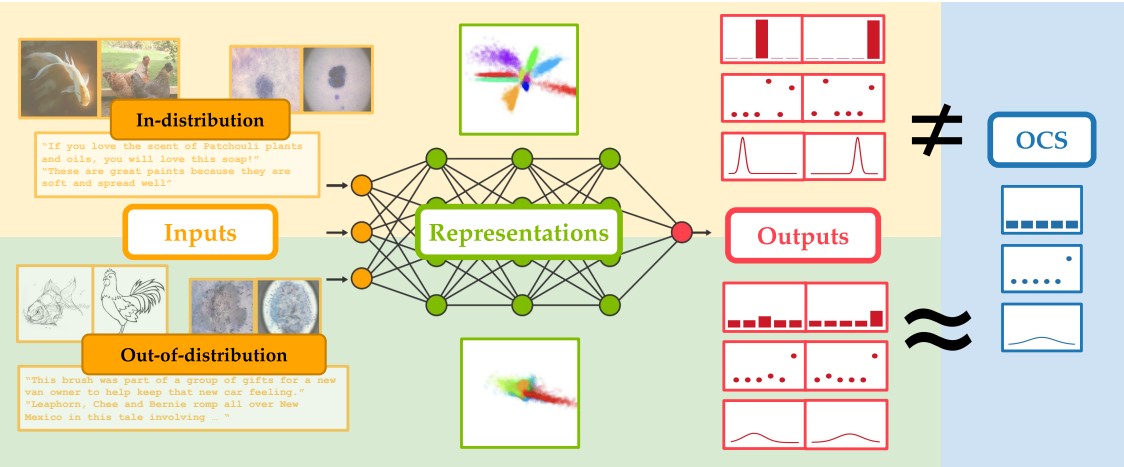

Figure 2.1: A summary of our observations. On in-distribution samples (top), neural network outputs tend to vary significantly based on input labels. In contrast, on OOD samples (bottom), we observe that model predictions tend to not only be more similar to one another, but also gravitate towards the optimal constant solution (OCS). We also observe that OOD inputs tend to map to representations with smaller magnitudes, leading to predictions largely dominated by the (constant) network biases, which may shed light on why neural networks have this tendency.

Baek et al., 2022, Balestriero et al., 2021], and the behavior of neural network representations under OOD conditions [Webb et al., 2020, Idnani et al., 2022, Pearce et al., 2021, Dietterich and Guyer, 2022, Huang et al., 2020]. While our work also analyzes representations in the context of extrapolation, our focus is on understanding the mechanism behind "reversion to the OCS", which differs from the aforementioned works.

Our work also explores risk-sensitive decision-making using selective classification as a testbed. Selective classification is a well-studied problem, and various methods have been proposed to enhance selective classification performance [Geifman and El-Yaniv, 2017, Feng et al., 2011, Charoenphakdee et al., 2021, Ni et al., 2019, Cortes et al., 2016, Xia and Bouganis, 2022]. In contrast, our aim is not to develop the best possible selective classification approach, but rather to providing insights into the effective utilization of neural network predictions in OOD decision-making.

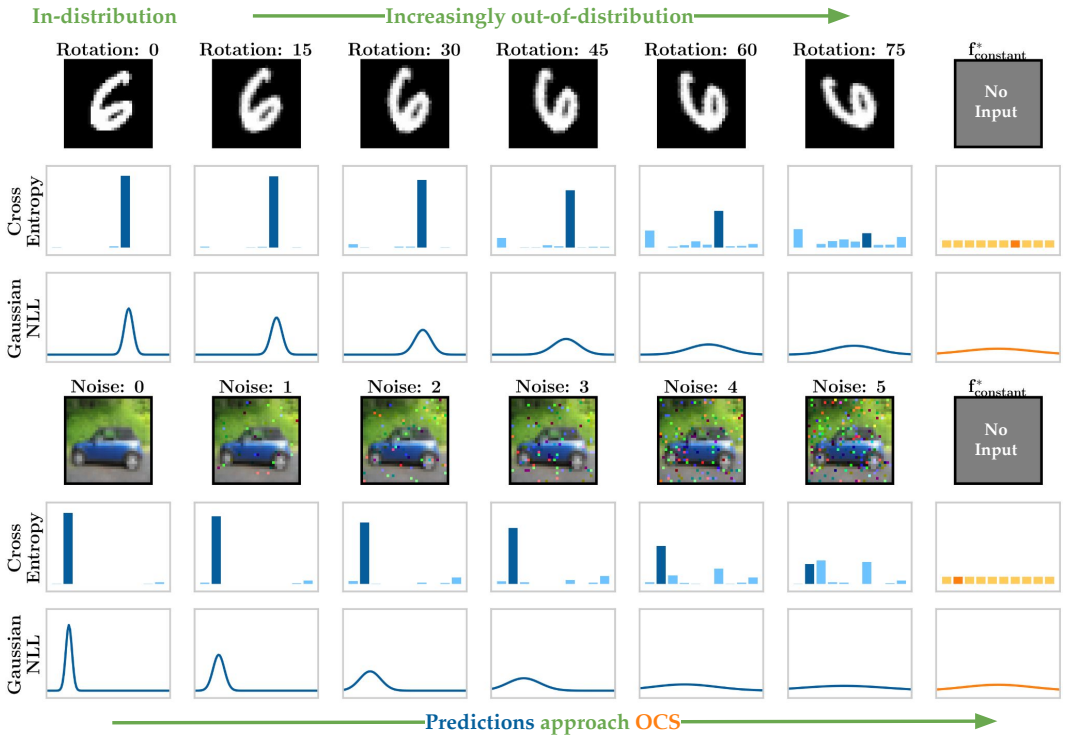

Figure 2.2: Neural network predictions from training with cross entropy and Gaussian NLL on MNIST (top 3 rows) and CIFAR10 (bottom 3 rows). The models were trained with 0 rotation/noise, and evaluated on increasingly OOD inputs consisting of the digit 6 for MNIST, and of automobiles for CIFAR10. The blue plots represent the average model prediction over the evaluation dataset. The orange plots show the OCS associated with each model. We can see that as the distribution shift increases (going left to right), the network predictions tend towards the OCS (rightmost column).

2.3 Reversion to the Optimal Constant Solution

In this work, we will focus on the widely studied covariate shift setting [Gretton et al., 2009, Sugiyama et al., 2007]. Formally, let the training data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ be generated by sampling $x_i \sim P_{\text{train}}(x)$ and $y_i \sim P(y|x_i)$. At test time, we query the model with inputs generated from $P_{\text{OOD}}(x) \neq P_{\text{train}}(x)$, whose ground truth labels are generated from the same conditional distribution, $P(y|x)$, as that in training. We will denote a neural network model as $f_\theta : \mathbb{R}^d \to \mathbb{R}^m$, where d and m are the dimensionalities of the input and output, and $\theta \in \Theta$ represents the network weights. We will focus on settings where d is high-dimensional. The neural network weights are optimized by minimizing a loss function $\mathcal{L}$ using gradient descent, $\hat{\theta} = \arg\min_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^N \mathcal{L}(f_\theta(x_i), y_i)$.

Main Hypothesis

In our experiments, we observed that as inputs become more OOD, neural network predictions tend to revert towards a constant prediction. This means that, assuming there is little label shift, model predictions will tend to be more similar to one another for OOD inputs than for the training distribution. Furthermore, we find that this constant prediction is often similar to the optimal constant solution (OCS), which minimizes the training loss if the network is constrained to ignore the input. The OCS can be interpreted as being the maximally cautious prediction, producing the class marginal in the case of the cross-entropy loss, and a high-variance Gaussian in the case of the Gaussian NLL. More precisely, we define the OCS as

$$f^*_{\text{constant}} = \arg\min_{f \in \mathbb{R}^m} \frac{1}{N} \sum_{1 \leq i \leq N} \mathcal{L}(f, y_i).$$

Based on our observations, we hypothesize that **as the likelihood of samples from $P_{\text{OOD}}(x)$ under $P_{\text{train}}(x)$ decreases, $f_{\hat{\theta}}(x)$ for $x \sim P_{\text{OOD}}(x)$ tends to approach f^*_{constant}.**

As an illustrative example, we trained models using either cross-entropy or (continuous-valued) Gaussian negative log-likelihood (NLL) on the MNIST and CIFAR10 datasets. The blue plots in Fig. 2.2 show the models' predictions as its inputs become increasingly OOD, and the orange plots visualize the OCS associated with each model. We can see that even though we trained on *different* datasets and evaluated on *different* kinds of distribution shifts, the neural network predictions exhibit the *same* pattern of extrapolation: as the distribution shift increases, the network predictions move closer to the OCS. Note that while the behavior of the cross-entropy models can likewise be explained by the network simply producing lower magnitude outputs, the Gaussian NLL models' predicted variance actually increases with distribution shift, which contradicts this alternative explanation.

Experiments

We will now provide empirical evidence for the "reversion to the OCS" hypothesis. Our experiments aim to answer the question: **As the test-time inputs become more OOD, do neural network predictions move closer to the optimal constant solution?**

Experimental setup. We trained our models on 8 different datasets, and evaluated them on both natural and synthetic distribution shifts. These datasets include CIFAR10 [Krizhevsky et al., 2009] / CIFAR10-C [Hendrycks and Dietterich, 2019], ImageNet [Deng et al., 2009] / ImageNet-R(endition) [Hendrycks et al., 2021a] / ImageNet-Sketch [Wang et al., 2019], DomainBed OfficeHome [Gulrajani and Lopez-Paz, 2020], SkinLesionPixels [Gustafsson et al., 2023], UTKFace [Zhang et al., 2017b], BREEDS living-17 / non-living-26 [Santurkar et al., 2020], and WILDS Amazon [Koh et al., 2021]. See Table 2.1 for a summary, and Appendix A.2 for a more detailed description of each dataset. Models with image inputs use ResNet [He et al., 2016] or VGG [Simonyan and Zisserman, 2014] style architectures, and models with

text inputs use DistilBERT [Sanh et al., 2019], a distilled version of BERT [Devlin et al., 2018].

Dataset	Label Type	Input Modality	Distribution Shift Type
CIFAR10 / CIFAR10-C	Discrete	Image	Synthetic
ImageNet / ImageNet-R(endition)	Discrete	Image	Natural
ImageNet / ImageNet-Sketch	Discrete	Image	Natural
DomainBed OfficeHome	Discrete	Image	Natural
SkinLesionPixels	Continuous	Image	Natural
UTKFace	Continuous	Image	Synthetic
BREEDS living-17	Discrete	Image	Natural
BREEDS non-living-26	Discrete	Image	Natural
WILDS Amazon	Discrete	Text	Natural

Table 2.1: Summary of the datasets that we train/evaluate on in our experiments.

We focus on three tasks, each using a different loss functions: classification with cross entropy (CE), selective classification with mean squared error (MSE), and regression with Gaussian NLL. Datasets with discrete labels are used for classification and selective classification, and datasets with continuous labels are used for regression. The cross entropy models are trained to predict the likelihood that the input belongs to each class, as is typical in classification. The MSE models are trained to predict rewards for a selective classification task. More specifically, the models output a value for each class as well as an abstain option, where the value represents the reward of selecting that option given the input. The ground truth reward is +1 for the correct class, -4 for the incorrect classes, and +0 for abstaining. We train these models by minimizing the MSE loss between the predicted and ground truth rewards. We will later use these models for decision-making in Section 2.5. The Gaussian NLL models predict a mean and a standard deviation, parameterizing a Gaussian distribution. They are trained to minimize the negative log likelihood of the labels under its predicted distributions.

Evaluation protocol. To answer our question, we need to quantify (1) the dissimilarity between the training data and the evaluation data, and (2) the proximity of network predictions to the OCS. To estimate the former, we trained a low-capacity model to discriminate between the training and evaluation datasets and measured the average predicted likelihood that the evaluation dataset is generated from the evaluation distribution, which we refer to as the OOD score. This score is 0.5 for indistinguishable train and evaluation data, and 1 for a perfect discriminator. To estimate the distance between the model's prediction and the OCS, we compute the KL divergence between the model's predicted distribution and the distribution parameterized by the OCS, $\frac{1}{N}\sum_{i=1}^{N} D_{\mathrm{KL}}(P_\theta(y|x_i)||P_{f^*_{\text{constant}}}(y))$, for models trained with cross-entropy and Gaussian NLL. For MSE models, the distance is measured using the mean squared error, $\frac{1}{N}\sum_{i=1}^{N} ||f_\theta(x_i) - f^*_{\text{constant}}||^2$. See Appendix A.2 for more details on our

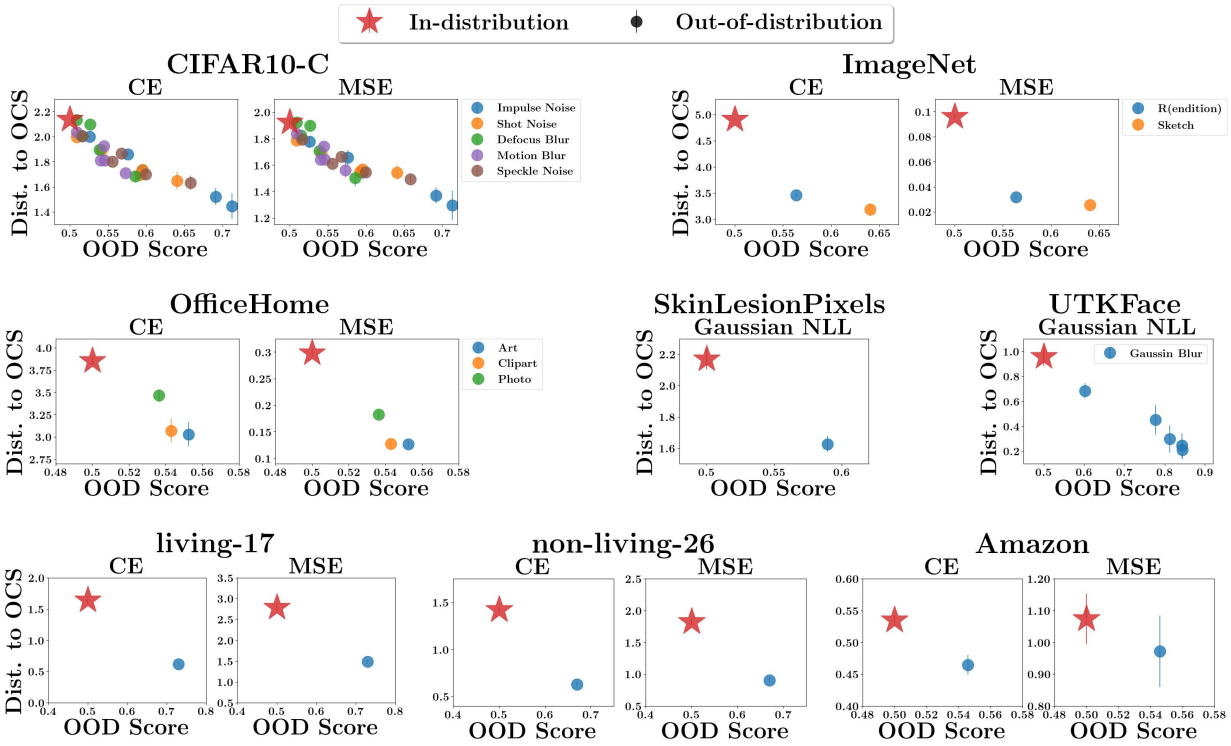

Figure 2.3: Evaluating the distance between network predictions and the OCS as the input distribution becomes more OOD. Each point represents a different evaluation dataset, with the red star representing the (holdout) training distribution, and circles representing OOD datasets. The vertical line associated with each point represents the standard deviation over 5 training runs. As the OOD score of the evaluation dataset increases, there is a clear trend of the neural network predictions approaching the OCS.

evaluation protocol, and Appendix A.2 for the closed form solution for the OCS for each loss.

Results. In Fig. 2.3, we plot the OOD score (x-axis) against the distance between the network predictions and the OCS (y-axis) for both the training and OOD datasets. Our results indicate a clear trend: as the OOD score of the evaluation dataset increases, neural network predictions move closer to the OCS. Moreover, our results show that this trend holds relatively consistently across different loss functions, input modalities, network architectures, and types of distribution shifts. We also found instances where this phenomenon did not hold, such as adversarial inputs, which we discuss in greater detail in Appendix A.1. However, the overall prevalence of "reversion to the OCS" across different settings suggests that it may capture a general pattern in the way neural networks extrapolate.

2.4 Why do OOD Predictions Revert to the OCS?

In this section, we aim to provide insights into why neural networks have a tendency to revert to the OCS. We will begin with an intuitive explanation, and provide empirical and theoretical evidence in Sections 2.4 and 2.4. In our analysis, we observe that weight matrices and network representations associated with training inputs often occupy low-dimensional subspaces with high overlap. However, when the network encounters OOD inputs, we observe that their associated representations tend to have less overlap with the weight matrices compared to those from the training distribution, particularly in the later layers. As a result, OOD representations tend to diminish in magnitude as they pass through the layers of the network, causing the network's output to be primarily influenced by the accumulation of model constants (e.g. bias terms). Furthermore, both empirically and theoretically, we find that this accumulation of model constants tend to closely approximate the OCS. We posit that reversion to the OCS for OOD inputs occurs due to the combination of these two factors: that accumulated model constants in a trained network tend towards the OCS, and that OOD points yield smaller-magnitude representations in the network that become dominated by model constants.

Empirical Analysis

We will now provide empirical evidence for the mechanism we describe above using deep neural network models trained on MNIST and CIFAR10. MNIST models use a small 4 layer network, and CIFAR10 models use a ResNet20 [He et al., 2016]. To more precisely describe the quantities we will be illustrating, let us rewrite the neural network as $f(x) = g_{i+1}(\sigma(W_i\phi_i(x) + b_i))$, where $\phi_i(x)$ is an intermediate representation at layer i, W_i and b_i are the corresponding weight matrix and bias, σ is a nonlinearity, and g_{i+1} denotes the remaining layers of the network. Because we use a different network architecture for each domain, we will use variables to denote different intermediate layers of the network, and defer details about the specific choice of layers to Appendix A.3. We present additional experiments analyzing the ImageNet domain, as well as the effects of batch and layer normalization in Appendix A.3.

First, we will show that $W_i\phi_i(x)$ tends to diminish for OOD inputs. The first column of plots in Fig. 2.4 show $\mathbb{E}_{x\sim P_{\text{OOD}}(x)}[||W_i\phi_i(x)||^2]/\mathbb{E}_{x\sim P_{\text{train}}(x)}[||W_i\phi_i(x)||^2]$ for P_{OOD} with different level of rotation or noise. The x-axis represents different layers in the network, with the leftmost being the input and the rightmost being the output. We can see that in the later layers of the network, $||W_i\phi_i(x)||^2$ consistently became smaller as inputs became more OOD (greater rotation/noise). Furthermore, the diminishing effect becomes more pronounced as the representations pass through more layers.

Next, we will present evidence that this decrease in representation magnitude occurs because $\phi_j(x)$ for $x \sim P_{\text{train}}(x)$ tend to lie more within the low-dimensional subspace spanned by the rows of W_j than $\phi_j(x)$ for $x \sim P_{\text{OOD}}(x)$. Let V_{top} denote the top (right) singular vectors of W_j. The middle plots of Fig. 2.4 show the ratio of the representation's norm at layer j

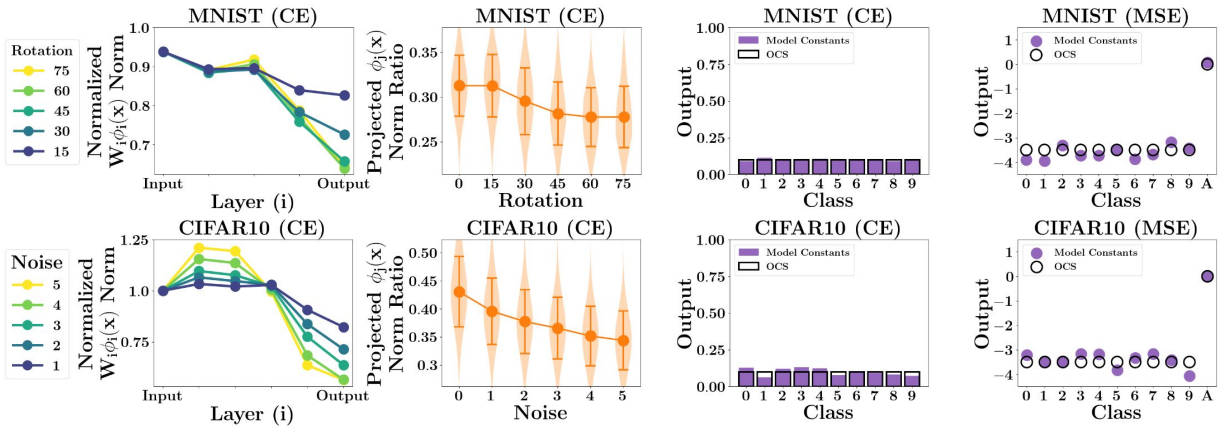

Figure 2.4: Analysis of the interaction between representations and weights as distribution shift increases. Plots in first column visualize the norm of network features for different levels of distribution shift at different layers of the network. In later layer of the network, the norm of features tends to decrease as distribution shift increases. Plots in second column show the proportion of network features which lie within the span of the following linear layer. This tends to decrease as distributional shift increases. Error bars represent the standard deviation taken over the test distribution. Plots in the third and fourth column show the accumulation of model constants as compared to the OCS for a cross entropy and a MSE model; the two closely mirror one another.

that is captured by projecting the representation onto V_{top} i.e., $||\phi_j(x)^\top V_{\text{top}} V_{\text{top}}^\top||^2/||\phi_j(x)||^2$, as distribution shift increases. We can see that as the inputs become more OOD, the ratio goes down, suggesting that the representations lie increasingly outside the subspace spanned by the weight matrix.

Finally, we will provide evidence for the part of the mechanism which accounts for the optimality of the OCS. Previously, we have established that OOD representations tend to diminish in magnitude in later layers of the network. This begs the question, what would the output of the network be if the input representation at an intermediary layer had a magnitude of 0? We call this the accumulation of model constants, i.e. $g_{k+1}(\sigma(b_k))$. In the third and fourth columns of Fig. 2.4, we visualize the accumulation of model constants at one of the final layers k of the networks for both a cross entropy and a MSE model (details in Sec. 2.3), along with the OCS for each model. We can see that the accumulation of model constants closely approximates the OCS in each case.

Theoretical Analysis

We will now explicate our empirical findings more formally by analyzing solutions of gradient flow (gradient descent with infinitesimally small step size) on deep homogeneous neural networks with ReLU activations. We adopt this setting due to its theoretical convenience in reasoning about solutions at convergence of gradient descent [Lyu and Li, 2019, Galanti et al., 2022, Huh et al., 2021], and its relative similarity to deep neural networks used in practice [Neyshabur et al., 2015a, Du et al., 2018].

Setup: We consider a class of homogeneous neural networks $\mathcal{F} := \{f(W; x) : W \in \mathcal{W}\}$, with L layers and ReLU activation, taking the functional form $f(W; x) = W_L\sigma(W_{L-1}\ldots\sigma(W_2\sigma(W_1x))\ldots)$, where $W_i \in \mathbb{R}^{m\times m}, \forall i \in \{2,\ldots,L-1\}$, $W_1 \in \mathbb{R}^{m\times 1}$ and $W_L \in \mathbb{R}^{1\times m}$. Our focus is on a binary classification problem where we consider two joint distributions $P_{\text{train}}, P_{\text{OOD}}$ over inputs and labels: $\mathcal{X} \times \mathcal{Y}$, where inputs are from $\mathcal{X} := \{x \in \mathbb{R}^d : \|x\|_2 \leq 1\}$, and labels are in $\mathcal{Y} := \{-1, +1\}$. We consider gradient descent with a small learning rate on the objective: $L(W; \mathcal{D}) := \sum_{(x,y)\in\mathcal{D}} \ell(f(W; x), y)$ where $\ell(f(W; x), y) \mapsto \exp(-yf(W; x))$ is the exponential loss and $\mathcal{D}$ is an IID sampled dataset of size N from P_{train}. For more details on the setup, background on homogeneous networks, and full proofs for all results in this section, please see Appendix A.4.

We will begin by providing a lower bound on the expected magnitude of intermediate layer features corresponding to inputs from the training distribution:

Proposition 2.4.1 (P_{train} observes high norm features) *When $f(\hat{W}; x)$ fits $\mathcal{D}$, i.e., $y_i f(\hat{W}; x_i) \geq \gamma$, $\forall i\in[N]$, then w.h.p $1-\delta$ over $\mathcal{D}$, layer j representations $f_j(\hat{W}; x)$ satisfy $\mathbb{E}_{P_{\text{train}}}[\|f_j(\hat{W}; x)\|_2] \geq (1/C_0)(\gamma - \tilde{\mathcal{O}}(\sqrt{\log(1/\delta)/N} + C_1 \log m/N\gamma))$, if $\exists$ constants C_0, C_1 s.t. $\|\hat{W}_j\|_2 \leq C_0^{1/L}$, $C_1 \geq C_0^{3L/2}$.*

Here, we can see that if the trained network perfectly fits the training data ($y_i f(\hat{W}; x_i)\geq\gamma$, $\forall i\in[N]$), and the training data size N is sufficiently large, then the expected ℓ_2 norm of layer j activations $f_j(\hat{W}; x)$ on P_{train} is large and scales at least linearly with γ.

Next, we will analyze the size of network outputs corresponding to points which lie outside of the training distribution. Our analysis builds on prior results for gradient flow on deep homogeneous nets with ReLU activations which show that the gradient flow is biased towards the solution (KKT point) of a specific optimization problem: minimizing the weight norms while achieving sufficiently large margin on each training point [Timor et al., 2023, Arora et al., 2019, Lyu and Li, 2019]. Based on this, it is easy to show that that the solution for this constrained optimization problem is given by a neural network with low rank matrices in each layer for sufficiently deep and wide networks. Furthermore, the low rank nature of these solutions is exacerbated by increasing depth and width, where the network approaches an almost rank one solution for each layer. If test samples deviate from this low rank space of weights in any layer, the dot products of the weights and features will collapse in the subsequent layer, and its affect rolls over to the final layer, which will output features with very small magnitude. Using this insight, we present an upper bound on the magnitude of the final layer features corresponding to OOD inputs:

Theorem 2.4.1 (Feature norms can drop easily on P_{OOD}) *Suppose $\exists$ a network $f'(W;x)$ with L' layers and m' neurons satisfying conditions in Proposition 2.4.1 (γ=1). When we optimize the training objective with gradient flow over a class of deeper and wider homogeneous networks $\mathcal{F}$ with $L > L', m > m'$, the resulting solution would converge directionally to a network $f(\hat{W};x)$ for which the following is true: $\exists$ a set of rank 1 projection matrices $\{A_i\}_{i=1}^{L}$, such that if representations for any layer j satisfy $\mathbb{E}_{P_{\text{OOD}}}\|A_j f_j(\hat{W};x)\|_2 \leq \epsilon$, then $\exists C_2$ for which $\mathbb{E}_{P_{\text{OOD}}}[|f(\hat{W};x)|] \lesssim C_0(\epsilon + C_2^{-1/L}\sqrt{L+1/L})$.*

This theorem tells us that for any layer j, there exists only a narrow rank one space A_j in which OOD representations may lie, in order for their corresponding final layer outputs to remain significant in norm. Because neural networks are not optimized on OOD inputs, we hypothesize that the features corresponding to OOD inputs tend to lie outside this narrow space, leading to a collapse in last layer magnitudes for OOD inputs in deep networks. Indeed, this result is consistent with our empirical findings in the first and second columns of Fig. 2.4, where we observed that OOD features tend to align less with weight matrices, resulting in a drop in OOD feature norms.

To study the accumulation of model constants, we now analyze a slightly modified class of functions $\tilde{\mathcal{F}} = \{f(W;\cdot) + b : b \in \mathbb{R}, f(W;\cdot) \in \mathcal{F}\}$, which consists of deep homogeneous networks with a bias term in the final layer. In Proposition 2.4.2, we show that there exists a set of margin points (analogous to support vectors in the linear setting) which solely determines the model's bias $\hat{b}$.

Proposition 2.4.2 (Analyzing network bias) *If gradient flow on $\tilde{\mathcal{F}}$ converges directionally to $\hat{W}, \hat{b}$, then $\hat{b} \propto \sum_k y_k$ for margin points $\{(x_k, y_k) : y_k \cdot f(\hat{W};x_k) = \arg\min_{j\in[N]} y_j \cdot f(\hat{W};x_j)\}$.*

If the label marginal of these margin points mimics that of the overall training distribution, then the learnt bias will approximate the OCS for the exponential loss. This result is consistent with our empirical findings in the third and fourth columns on Fig. 2.4, where we found the accumulation of bias terms tends to approximate the OCS.

2.5 Risk-Sensitive Decision-Making

Lastly, we will explore an application of our observations to decision-making problems. In many decision-making scenarios, certain actions offer a high potential for reward when the agent chooses them correctly, but also higher penalties when chosen incorrectly, while other more cautious actions consistently provide a moderate level of reward. When utilizing a learned model for decision-making, it is desirable for the agent to select the high-risk high-reward actions when the model is likely to be accurate, while opting for more cautious actions when the model is prone to errors, such as when the inputs are OOD. It turns out, if we leverage "reversion to the OCS" appropriately, such risk-sensitive behavior can emerge automatically. If the OCS of the agent's learned model corresponds to cautious actions, then

"reversion to the OCS" posits that the agent will take increasingly cautious actions as its inputs become more OOD. However, not all decision-making algorithms leverage "reversion to the OCS" by default. **Depending on the choice of loss function (and consequently the OCS), different algorithms which have similar in-distribution performance can have different OOD behavior.** In the following sections, we will use selective classification as an example of a decision-making problem to more concretely illustrate this idea.

Example Application: Selective Classification

In selective classification, the agent can choose to classify the input or abstain from making a decision. As an example, we will consider a selective classification task using CIFAR10, where the agent receives a reward of +1 for correctly selecting a class, a reward of -4 for an incorrect classification, and a reward of 0 for choosing to abstain.

Let us consider one approach that leverages "reversion to the OCS" and one that does not, and discuss their respective OOD behavior. An example of the former involves learning a model to predict the reward associated with taking each action, and selecting the action with the highest predicted reward. This reward model, $f_\theta : \mathbb{R}^d \to \mathbb{R}^{|\mathcal{A}|}$, takes as input an image, and outputs a vector the size of the action space. We train f_θ using a dataset of images, actions and rewards, $\mathcal{D} = \{(x_i, a_i, r_i)\}_{i=1}^N$, by minimizing the MSE loss, $\frac{1}{N}\sum_{1\leq i\leq N}(f_\theta(x_i)_{a_i} - r_i)^2$, and select actions using the policy $\pi(x) = \arg\max_{a\in\mathcal{A}} f_\theta(x)_a$. The OCS of f_θ is the average reward for each action over the training points, i.e., $(f^*_{\text{constant}})_a = \frac{\sum_{1\leq i\leq N} r_i \cdot \mathbb{1}[a_i=a]}{\sum_{1\leq j\leq N} \mathbb{1}[a_j=a]}$. In our example, the OCS is -3.5 for selecting each class and 0 for abstaining, so the policy corresponding to the OCS will choose to abstain. Thus, according to "reversion to the OCS", this agent should choose to abstain more and more frequently as its input becomes more OOD. We illustrate this behavior in Figure 2.5. In the first row, we depict the average predictions of a reward model when presented with test images of a specific class with increasing levels of noise (visualized in Figure 2.1). In the second row, we plot a histogram of the agent's selected actions for each input distribution. We can see that as the inputs become more OOD, the model's predictions converged towards the OCS, and consequently, the agent automatically transitioned from making high-risk, high-reward decisions of class prediction to the more cautious decision of abstaining.

One example of an approach which does not leverage "reversion to the OCS" is standard classification via cross entropy. The classification model takes as input an image and directly predicts a probability distribution over whether each action is the optimal action. In this case, the optimal action given an input is always its ground truth class. Because the OCS for cross entropy is the marginal distribution of labels in the training data, and the optimal action is never to abstain, the OCS for this approach corresponds to a policy that *never* chooses to abstain. In this case, "reversion to the OCS" posits that the agent will continue to make high-risk high-reward decisions even as its inputs become more OOD. As a result, while this approach can yield high rewards on the training distribution, it is likely to yield very low rewards on OOD inputs, where the model's predictions are likely to be incorrect.

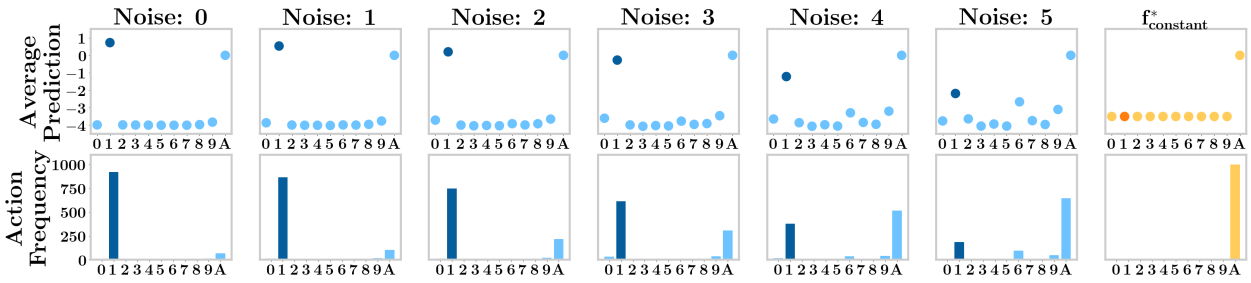

Figure 2.5: Selective classification via reward prediction on CIFAR10. We evaluate on holdout datasets consisting of automobiles (class 1) with increasing levels of noise. X-axis represents the agent's actions, where classes are indexed by numbers and abstain is represented by "A". We plot the average reward predicted by the model for each class (top), and the distribution of actions selected by the policy (bottom). The rightmost plots represent the OCS (top), and the actions selected by an OCS policy (bottom). As distribution shift increased, the model predictions approached the OCS, and the policy automatically selected the abstain action more frequently.

Experiments

We will now more thoroughly compare the behavior of a reward prediction agent with a standard classification agent for selective classification on a variety of different datasets. Our experiments aim to answer the questions: **How does the performance of a decision-making approach which leverages "reversion to the OCS" compare to that of an approach which does not?**

Experimental Setup. Using the same problem setting as the previous section, we consider a selective classification task in which the agent receives a reward of +1 for selecting the correct class, -4 for selecting an incorrect class, and +0 for abstaining from classifying. We experiment with 4 datasets: CIFAR10, DomainBed OfficeHome, BREEDS living-17 and non-living-26. We compare the performance of the reward prediction and standard classification approaches described in the previous section, as well as a third oracle approach that is optimally risk-sensitive, thereby providing an upper bound on the agent's achievable reward. To obtain the oracle policy, we train a classifier on the training dataset to predict the likelihood of each class, and then calibrate the predictions with temperature scaling on the OOD evaluation dataset. We then use the reward function to calculate the theoretically optimal threshold on the classifier's maximum predicted likelihood, below which the abstaining option is selected. Note that the oracle policy has access to the reward function and the OOD evaluation dataset for calibration, which the other two approaches do not have access to.

Results. In Fig. 2.6, we plot the frequency with which the abstain action is selected for each

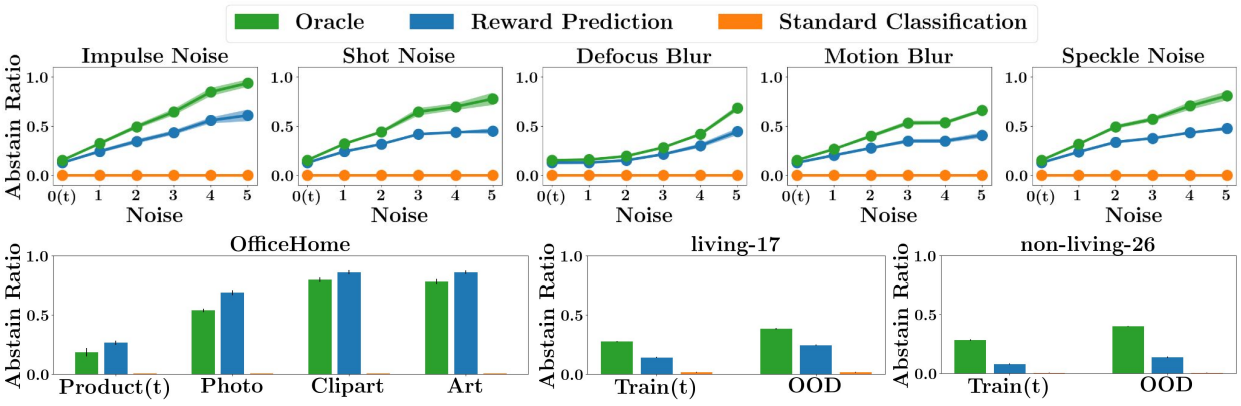

Figure 2.6: Ratio of abstain action to total actions; error bars represent standard deviation over 5 random seeds; (t) denotes the training distribution. While the oracle and reward prediction approaches selected the abstain action more frequently as inputs became more OOD, the standard classification approach almost never selected abstain.

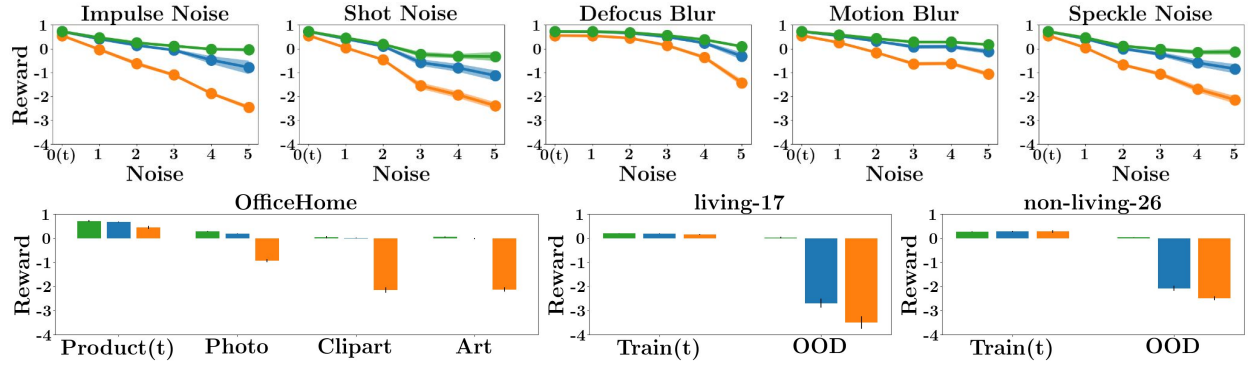

Figure 2.7: Reward obtained by each approach. While all three approaches performed similarly on the training distribution, reward prediction increasingly outperformed standard classification as inputs became more OOD.

approach. As distribution shift increased, both the reward prediction and oracle approaches selected the abstaining action more frequently, whereas the standard classification approach never selected this option. This discrepancy arises because the OCS of the reward prediction approach aligns with the abstain action, whereas the OCS of standard classification does not. In Fig. 2.7, we plot the average reward received by each approach. Although the performance of all three approaches are relatively similar on the training distribution, the reward prediction policy increasingly outperformed the classification policy as distribution shift increased. Furthermore, the gaps between the rewards yielded by the reward prediction and classification policies are substantial compared to the gaps between the reward prediction and the oracle policies, suggesting that the former difference in performance is nontrivial. Note that the goal of our experiments was not to demonstrate that our approach is the best possible method for selective classification (in fact, our method is likely not better than SOTA approaches), but rather to highlight how the OCS associated with an agent's learned model can influence its OOD decision-making behavior. To this end, this result shows that appropriately leveraging "reversion to the OCS" can substantial improve an agent's performance on OOD inputs.

2.6 Conclusion

We presented the observation that neural network predictions for OOD inputs tend to converge towards a specific constant, which often corresponds to the optimal input-independent prediction based on the model's loss function. We proposed a mechanism to explain this phenomenon and a simple strategy that leverages this phenomenon to enable risk-sensitive decision-making. Finally, we demonstrated the prevalence of this phenomenon and the effectiveness of our decision-making strategy across diverse datasets and different types of distributional shifts.

Our understanding of this phenomenon is not complete. Further research is needed to to discern the properties of an OOD distribution which govern when, and to what extent, we can rely on "reversion to the OCS" to occur. Another exciting direction would be to extend our investigation on the effect of the OCS on decision-making to more complex multistep problems, and study the OOD behavior of common algorithms such as imitation learning, Q-learning, and policy gradient.

As neural network models become more broadly deployed to make decisions in the "wild", we believe it is increasingly essential to ensure neural networks behave safely and robustly in the presence of OOD inputs. While our understanding of "reversion to the OCS" is still rudimentary, we believe it offers a new perspective on how we may predict and even potentially steer the behavior of neural networks on OOD inputs. We hope our observations will prompt further investigations on how we should prepare models to tackle the diversity of in-the-wild inputs they must inevitably encounter.

Chapter 3

Large Language Model Hallucinations

3.1 Introduction

Large language models (LLMs) have a tendency to hallucinate—generating seemingly unpredictable responses that are often factually incorrect. This behavior is especially prominent when models are queried on concepts that are scarcely represented in their pretraining corpora [Kandpal et al., 2023, Kalai and Vempala, 2023] (e.g. asking the model to generate the biography of a little-known person). We will refer to these queries as *unfamiliar* inputs. Rather than fabricating information when presented with unfamiliar inputs, models should instead recognize the limits of their own knowledge, and verbalize their uncertainty or confine their responses within the limits of their knowledge. The goal of our work is to teach models this behavior, particularly for long-form generation tasks.

Towards this goal, we first set out to better understand the underlying mechanisms that govern how LLMs hallucinate. Perhaps surprisingly, we observe that finetuned LLMs do not produce arbitrary predictions when presented with unfamiliar queries. Instead, their predictions tend to mimic the responses associated with the unfamiliar examples in the model's finetuning data (i.e. those unfamiliar to the pretrained model). More specifically, as test inputs become more unfamiliar, we observe that LLM predictions tend to default towards a hedged prediction. This hedged prediction, akin to an intelligent "blind guess", minimizes the *aggregate* finetuning loss over unfamiliar finetuning examples. For a model trained with standard supervised finetuning (SFT), where all finetuning examples are labeled with ground-truth answers, the model's default hedged prediction corresponds to the distribution of ground-truth answers associated with unfamiliar finetuning examples. As a result, when queried on a new example at test-time, the model hallucinates a plausibly sounding answer. Thus, our conceptual model provides an explanation for why, when faced with unfamiliar inputs at test-time, these models often generate responses that sound plausible but do not correctly answer the prompt. We provide evidence for our observation with controlled experiments on multi-choice answering (MMLU [Hendrycks et al., 2020]).

Rather than hallucinating incorrect responses (in the case of standard SFT), we would

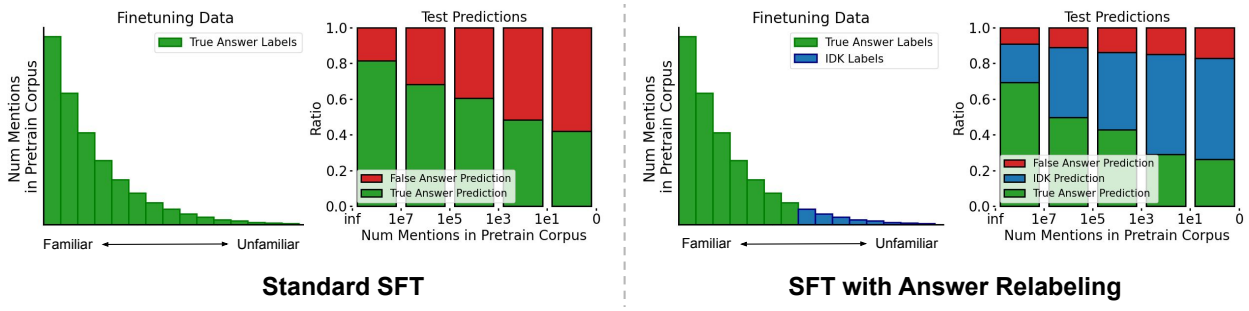

Figure 3.1: Prediction behavior of models finetuned with standard finetuning (left) and finetuning with answer relabeling (right) on the TriviaQA dataset. The "long-tail" figures conceptually visualize the data used to finetune the models, while the bar plot figures show results for the models' predictions. We can see that the standard model, finetuned with only true answer labels, tends to make false answer predictions more frequently as test inputs become more unfamiliar. The model with answer relabeling is finetuned with true answer labels for familiar inputs and "I don't know" labels for unfamiliar inputs. This model predicts "I don't know" more frequently as test inputs become more unfamiliar.

instead like for models to admit their ignorance in the face of unfamiliar inputs. Our conceptual model of LLM hallucinations reveals a general recipe for teaching models this behavior: by strategically manipulating the unfamiliar examples in the model's finetuning data, we can steer the form of the model's default hedged prediction, and consequently the model's generations towards more desirable (e.g. linguistically uncertain) responses for unfamiliar queries. One way to instantiate this strategy, illustrated in Fig. 3.1, is to finetune the model with SFT on a dataset whose unfamiliar queries are relabeled with "I don't know" responses. We can see that the resulting model, whose default hedged prediction is "I don't know", responds with "I don't know" more frequently as inputs became more unfamiliar at test-time. Another way to steer a model's predictions for unfamiliar queries is via reinforcement learning (RL) with a strategically designed reward function. If the reward function encourages uncertain responses over factually incorrect responses, the model's default hedged prediction will consist of uncertain responses, thereby teaching the model to produce more uncertain responses rather than incorrect responses on unfamiliar queries.

Our ultimate goal is to improve the factuality of long-form LLM generations. While the recipe we describe above offers a high-level solution, instantiating this recipe requires specialized (potentially model-specific) supervision during finetuning, which raises some practical challenges for long-form generation tasks. For instance, SFT-based approaches for controlling model hallucinations require target responses that take into account the pretrained model's knowledge (or lack thereof), which can be very expensive for long-form generations. In contrast, RL-based approaches make use of scalar rewards signals to supervise model generations. Using a reward model to predict the factuality of model responses offers a promising

way to scalably supervise RL factuality finetuning. However, reward models themselves can suffer from hallucinations in the face of unfamiliar inputs. To tackle this challenge, we draw on our previous insights to strategically control how reward models hallucinate. More specifically, we propose an approach for learning conservative reward models that avoid overestimating rewards in the face of unfamiliar inputs, thereby minimizing the adverse effects of reward hallucinations on RL factuality finetuning. On the tasks of biography and book/movie plot generation, we show that finetuning LLMs using RL with conservative reward models can reliably teach them to generate more factual long-form responses, outperforming standard SFT, and RL with standard reward models.

To summarize, the main contributions of our work are twofold: (1) we present a conceptual model of how finetuned LLMs make predictions in the face of unfamiliar queries and, (2) we propose a way to control reward model hallucinations in order to improve the performance of RL factuality finetuning. Our conceptual model highlights the important role of unfamiliar finetuning examples in determining the hallucination behavior of finetuned LLMs, and brings to light a general recipe for controlling how LLMs hallucinate. Drawing on our observations, our approach for RL factuality finetuning with conservative reward models enables models to more reliably produce factual long-form generations. We hope that the insights in our work contribute to a better understanding of the mechanisms that govern how LLMs hallucinate as well as the key principles for controlling these hallucinations.

3.2 Related Work

A large body of work has documented the tendency of LLMs to hallucinate factually incorrect responses [Kalai and Vempala, 2023, Bubeck et al., 2023, Kadavath et al., 2022, Agrawal et al., 2023]. Additionally, studies have investigated the conditions under which hallucinations occur and how LLMs behave in such instances. For example, LLMs tend to hallucinate more frequently when queried on knowledge that is rarely mentioned in their training data [Mallen et al., 2023, Kandpal et al., 2023]. Furthermore, LLM predictions generally tend to be moderately calibrated [Kadavath et al., 2022, Zhao et al., 2021, Tian et al., 2023b], and their internal representations seem to reflect some awareness of model uncertainty [Liu et al., 2023, Azaria and Mitchell, 2023]. Our work extends these findings by demonstrating that we can train models to make predictions based on their internal uncertainty.

A key insight of our work is that finetuned LLMs tend to default towards a hedged prediction when faced with unfamiliar queries. Prior work has observed a similar phenomena in standard neural networks (those without pretraining) [Kang et al., 2023, Hendrycks and Gimpel, 2016]. These works show that, as inputs become more out-of-distribution, neural network predictions tend to converge towards a constant value —much like the default hedged prediction in our work. However, because standard neural networks lack the initial foundation of a pretrained model, the constant prediction reflects the model's training distribution rather than unfamiliar data encountered during finetuning.

Finally, a number of works seek to address the challenges posed by LLM hallucinations. Active research areas include hallucination detection [Manakul et al., 2023, Mündler et al., 2023, Xu et al., 2023, Kuhn et al., 2023], automated evaluation of factuality [Min et al., 2023, Umapathi et al., 2023, Jing et al., 2023], and mitigation techniques. Common strategies for mitigating hallucinations include specialized sampling methods [Lee et al., 2022, Li et al., 2023a, Chuang et al., 2023, Zhang et al., 2023b], more reliable input prompts [Si et al., 2022], and retrieval augmentation to incorporate external knowledge [Gao et al., 2023, Peng et al., 2023, Varshney et al., 2023, Yao et al., 2023, Shuster et al., 2021]. Closest to our work, prior research has also used finetuning methods like SFT [Lin et al., 2022, Yang et al., 2023, Zhang et al., 2023a] and RL [Shulman, 2023, Goldberg, 2023, Tian et al., 2023a, Sun et al., 2023, Roit et al., 2023, Mesgar et al., 2020] to reduce hallucinations (discussed in more detail in Sec. 3.5). Our work proposes a conceptual model of LLM hallucinations that explains the underlying mechanisms behind a number of these different methods. Furthermore, we leverage our insights to tackle the understudied problem of reward model hallucinations in RL-based factuality finetuning, to enable more scalable factuality finetuning for long-form generation tasks.

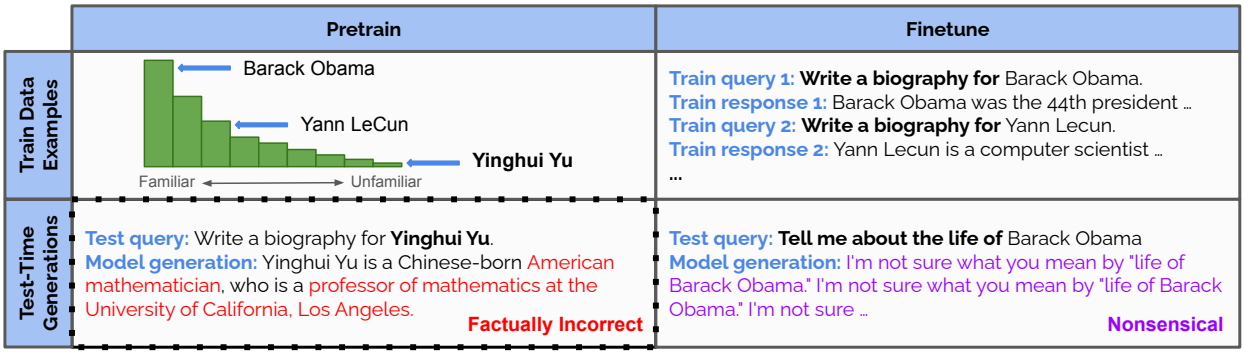

Figure 3.2: A visualization of how test-time queries that lie outside the model's pretraining and finetuning data distributions can manifest in different kinds of hallucinations. On the bottom left, we show an example of a model's generation when prompted with a query that is linguistically similar to those in the finetuning data, but requires knowledge (i.e. details about Yinghui Yu) that is scarcely present in the pretraining data; the model's response is full of factually incorrect information. On the bottom right, we show an example of a model's generation when prompted with a query that is linguistically different from those in the finetuning dataset ("Tell me about the life of ..." vs "Write a biography for ..."); there model's response is nonsensical. In this work, we focus on the scenario on the left.

3.3 Problem Setting

Modern LLMs are typically trained in a two-stage process: pretraining on broad-coverage corpora, followed by finetuning on more specialized instruction-following datasets [Ouyang et al., 2022]. These models can be prone to hallucinating when prompted with inputs that are not well represented in the model's training data. While distribution shift with respect to the finetuning data (e.g., phrasing questions differently as in bottom right of Fig. 3.2) can lead to hallucinatory generations, even queries that come from the finetuning distribution can lead to hallucinations, if the query requires knowledge of concepts scarcely represented in the pretraining corpus (bottom left of Fig. 3.2). We focus on the latter regime of hallucinations in our work. Our goal is to better understand and control how LLMs make predictions in the face of unfamiliar inputs, which require knowledge of concepts that model does not possess.

While our definition of unfamiliar inputs is imprecise, prior works have identified more concrete metrics that are representative of a model's familiarity to a query or concept in specific settings [Kandpal et al., 2023, Kadavath et al., 2022]. In our experiments, we quantify the familiarity of an input using the number of times concepts from the input are mentioned in the pretraining corpus [Kandpal et al., 2023] or, for structured generation tasks like classification, the confidence of the logits of the pretrained model's predicted response distribution [Kadavath et al., 2022]. For more details about our implementation of these

metrics, see Appendix B.1. We will use question-answer tasks as a testbed for our empirical evaluation in this work, though our analysis and method can apply to any prompted generation LLM task. To avoid confounding our observations the effects of distribution shift with respect to the finetuning distribution, we evaluate model predictions on held-out queries sampled from the same distribution as the finetuning data.

3.4 Understanding how LLMs Hallucinate

In this section, we investigate the underlying mechanisms that govern how finetuned LLMs hallucinate. We will first present a hypothesis for these mechanisms, and then validate our hypothesis with a series of controlled experiments.

Main Hypothesis

At a high level, our hypothesis posits that model hallucinations on unfamiliar inputs will mimic the distribution of responses associated with the unfamiliar examples in the model's finetuning dataset. To explain our hypothesis more precisely, let us consider an LLM f_θ, which maps a prompt x to a distribution of responses $P(y)$. We finetune this model on a dataset $\mathcal{D} = \{(x_i, s_i)\}_{1 \leq i \leq N}$ with a loss function $\sum_{(x_i,s_i)\in\mathcal{D}} \mathcal{L}(f_\theta(x_i), s_i)$. Depending on the choice of $\mathcal{L}$, this can represent SFT (where the supervision s_i is a a target response) or RL finetuning (where s_i is a reward function). We refer to examples in the finetuning dataset as being unfamiliar if the input associated with the example is unfamiliar to the pretrained model that is being finetuned.

While the optimal behavior that an LLM can learn during finetuning is to output the ground-truth answer to each query, this may not happen in practice for all finetuning examples. For familiar finetuning examples, the pretrained model's representations often encode useful associations between queries and responses, facilitating the finetuning optimization for those examples. However, for unfamiliar examples, which we refer to as $\mathcal{D}_{\text{unfamiliar}}$, such helpful associations in the pretrained representations are largely absent, making it more difficult to model these examples. Nonetheless, while an LLM may struggle to produce the optimal response for each query in $\mathcal{D}_{\text{unfamiliar}}$, it can still reduce the finetuning loss by learning the general pattern of optimal responses across these examples. This means producing a *hedged* prediction that, while not perfect for any single example, minimizes the *aggregate* loss over the set of unfamiliar finetuning examples, i.e., $P_{\text{hedged}}(y) = \arg\min_{P(y)} \sum_{(x_i,s_i)\in\mathcal{D}_{\text{unfamiliar}}} \mathcal{L}(P(y), s_i)$. Note that this hedged prediction, akin to an intelligent "blind guess", does not depend on a specific input. We hypothesize that LLMs learn to produce this kind of hedged prediction for unfamiliar examples during finetuning, and that they retain this behavior in the face of unfamiliar inputs during test time. More specifically, we posit that **as inputs become more unfamiliar, the predictions of finetuned LLMs will default towards the model's hedged prediction.**

In standard finetuning on multiple-choice tasks, the model's hedged prediction corresponds to the distribution of answer choices in the finetuning data. In long-form structured generation tasks, the model's hedged prediction corresponds to responses whose linguistic structure matches those of the finetuning responses, but whose details may not correctly correspond to the test-time prompt. Sampling a response from these output distributions is likely to produce a confident but incorrect answer. If, however, we instead strategically modulate the supervision associated with the model's unfamiliar finetuning examples, our hypothesis suggests that we may actually be able to control how LLMs hallucinate by steering the form of the model's hedged prediction. We will discuss these implications of our hypothesis in more detail in subsequent sections.

Experimental Verification of our Main Hypothesis

To empirically evaluate our hypothesis, we conduct controlled experiments analyzing the prediction behavior of a variety of models with different hedged predictions. The goal of our experiments is to answer the following question about these models: As inputs to a finetuned LLM become more unfamiliar, do model predictions default towards the model's hedged prediction?

Experimental setup. A model's hedged prediction is determined by the supervision associated with the unfamiliar examples in the model's finetuning data (i.e., s_i in $\mathcal{D}_{\text{unfamiliar}}$). Thus, to isolate their influence, we finetune pairs of models, varying the supervision associated with unfamiliar examples while keeping all other training variables and procedures fixed. We finetune models using both SFT and RL to evaluate our hypothesis for different training objectives.

We conduct our experiments with a multiple-choice question answering task (MMLU), where it is tractable to calculate a model's default hedged prediction. We finetune Llama2-7B models [Touvron et al., 2023] on a subset of the MMLU dataset [Hendrycks et al., 2020], and evaluate its predictions on a held out test set. To quantify a query's unfamiliarity, we measure the uncertainty in the logits predicted by the pretrained model. More specifically, we use negative log likelihood (NLL) of the correct answer under the distribution (normalized over A-D tokens) predicted by the pretrained model when prompted with the query and 5 other example query/response pairs. For further experimental details, see Appendix B.2.

Supervised finetuning. First, we investigate the prediction behavior of models finetuned with SFT. For these models, the hedged prediction corresponds to the marginal distribution of target responses in the set of unfamiliar finetuning examples. We will consider two different finetuning data distributions in our experiments. In the first distribution, the target responses for both familiar and unfamiliar queries are distributed uniformly (over A-D tokens), which is standard for multiple choice finetuning. In the second distribution, the target responses for familiar queries are distributed uniformly, while the target responses for

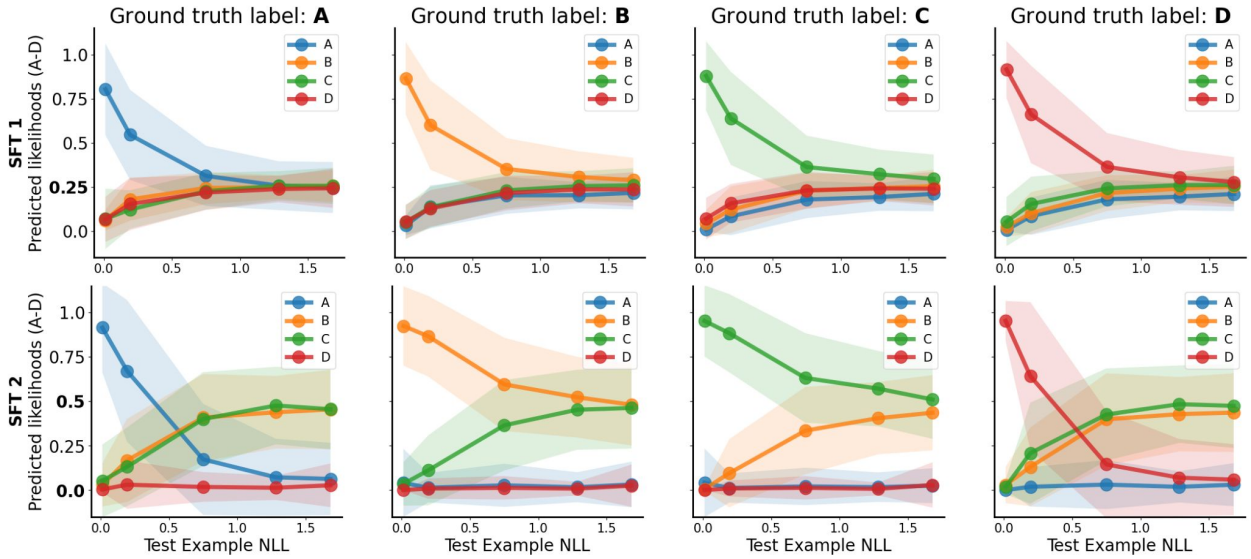

Figure 3.3: Prediction behavior of models finetuned with SFT. The model in the top row is finetuned on a dataset with uniformly distributed labels, while model on the bottom row is finetuned on dataset for which the labels for unfamiliar examples are sampled from a distribution given by 50% B and 50% D. Within each column, only test inputs with a specific ground truth label (A-D) are evaluated. Within each plot, going left to right on the x-axis corresponds to test inputs becoming more unfamiliar. Solid line represents the average predicted likelihood, and error bars represent standard deviation within the test set. Note that as inputs become more unfamiliar, the predictions of the different models differ, because the two models have different default hedged predictions.

unfamiliar queries are distributed 50% B and 50% C. Thus, the hedged prediction of models finetuned on these datasets are uniform and 50% B/50% C respectively.

In Fig. 3.3, we plot the predicted distribution (over tokens A-D) of the two models as its test inputs become more unfamiliar (left to right on the x-axis). We can see that for familiar inputs, both models tend to predict higher likelihoods for the letter associated with the ground truth answer. However, as inputs become more unfamiliar, the predictions of the first model approached the uniform distribution, while the predictions of the second model approached the 50% B/50% C distribution. Our results show that model predictions indeed default towards the hedged prediction as inputs become more unfamiliar. Furthermore, because all training details are kept constant across the two models except for the distribution of unfamiliar examples in the models' finetuning data, this experiment illustrates that these examples are indeed the ones that determine how the LLM hallucinates.

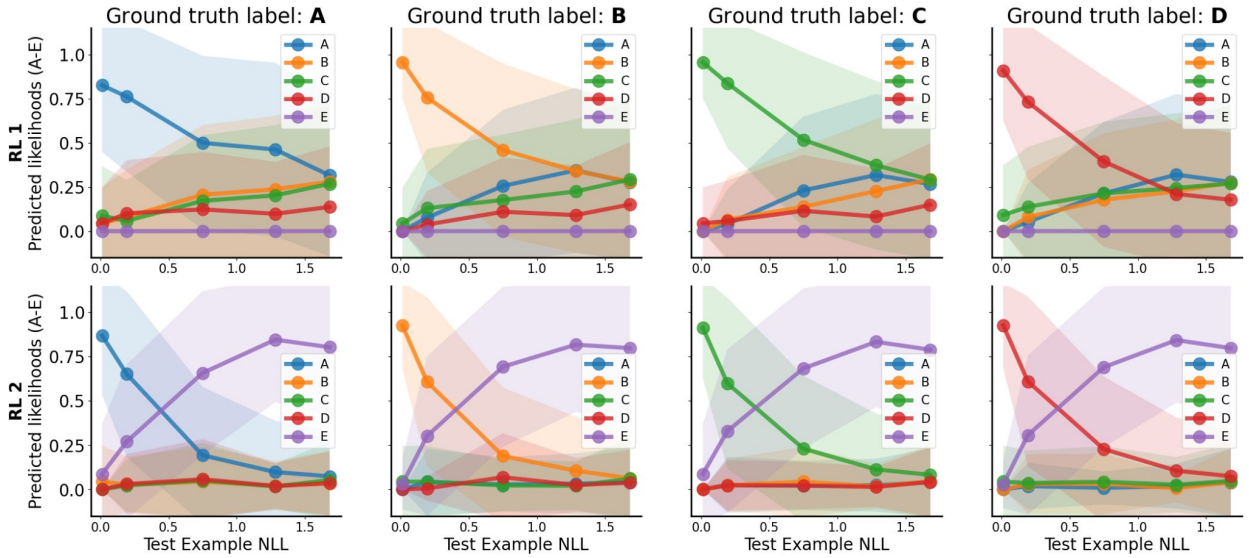

Figure 3.4: Prediction behavior of models finetuned with RL. The model on the top row is finetuned with a reward function which does not favor an abstain answer (E) over an incorrect answer, while the model on the bottom row is finetuned with a reward function which does. As inputs become more unfamiliar, the first model increasing produced random guesses while the second model produced abstain answers, because the two models have different default hedged predictions.

Reinforcement learning. Next, we investigate the prediction behavior of models finetuned with RL, using PPO [Schulman et al., 2017]. In this case, the model's hedged prediction is determined by the reward function used for RL training. More specifically, the hedged prediction maximizes the aggregate reward functions associated with unfamiliar finetuning examples. To highlight the influence of the reward function on model predictions, we add a fifth answer option, E, to the standard multiple-choice task, which represents abstaining from answering. We will consider two different reward functions for RL finetuning. The first reward function we consider outputs a reward of +2 for the correct answer, -3 for an incorrect answer, and -3 for abstaining. The second reward function we consider outputs +2 for the correct answer, -3 for an incorrect answer, and 0 for abstaining. The hedged prediction of a model trained to maximize the first reward function is to randomly guess an answer, whereas hedged prediction corresponding to the second reward function is to abstain from answering. This is because, in the first reward function, randomly guessing an answer yields a higher expected reward than abstaining from answering, whereas in the second reward function, abstaining from answering yields higher expected reward than randomly guessing an answer.

We plot the RL model's predictions as inputs become more unfamiliar in Fig. 3.4. Similar to the previous SFT experiments, the RL models predict higher likelihoods for the ground truth answer when faced with familiar inputs. As inputs become more unfamiliar, we see that two RL models exhibit different behavior. While the model with the first reward function increasingly produced random guesses, the model with the second reward model increasingly produced abstaining answers. These results show that models finetuned with an RL loss also default towards the hedged prediction as inputs become more unfamiliar. In addition, these results illustrate how RL finetuning with strategically designed reward functions can teach models to produce abstaining responses on unfamiliar queries.

3.5 Controlling Language Model Hallucinations

Our conceptual model of LLM hallucinations from the previous section reveals a general recipe for teaching models to admit their uncertainty in their responses. By strategically steering the model's default hedged prediction, we can control how the model hallucinates in the face of unfamiliar examples. In this section, we reinterpret two classes of existing approaches for factuality finetuning within the context of our conceptual model. We will show how the underlying design decisions utilized by these works implicitly or explicitly modify the supervision associated with unfamiliar examples during finetuning in order to obtain better factuality performance. While our experiments in the previous section illustrated this strategy for multiple-choice answering, the discussion in this section shows that our conceptual framework is also applicable to more general question-answer tasks.

SFT with Answer Relabeling. We will first consider a class of prior methods that use SFT to teach models to abstain from answering questions for which they do not know the answer [Zhang et al., 2023a, Yang et al., 2023, Lin et al., 2022]. Specifically, these methods (1) identify examples in the finetuning data for which the model cannot generate the correct answer, (2) relabel those examples to have a hedged or abstaining response (e.g., "I don't know."), and (3) finetune the base model with this relabeled dataset using SFT. The results in these works show that on held out queries, models finetuned with this approach are able to achieve higher selective accuracy compared to models with standard finetuning. We show an example of this approach in Fig. 3.1 (implementation details in Appendix B.3).

Our conceptual model, which posits that model predictions revert towards a default hedged prediction as test inputs become more unfamiliar, provides an explanation for the effectiveness of this approach. By identifying examples in the finetuning data for which the base model cannot generate the correct answer (i.e., the unfamiliar finetuning examples) and relabeling their target responses to be an abstaining response, this approach manipulates the model's default hedged prediction to this abstaining response. Thus, according to our conceptual model, the model's predictions should revert towards this abstaining response in the face of unfamiliar inputs at test-time, which leads to higher selective accuracy. Indeed,

in Fig. 3.1, we can see that the SFT model with relabeled finetuning generated more "I don't know" responses as queries became more unfamiliar.

RL with Factuality Rewards Next, we discuss a second class of methods, which uses RL to finetune models to produce more factual responses [Shulman, 2023, Roit et al., 2023, Sun et al., 2023, Tian et al., 2023a, Mesgar et al., 2020]. The reward functions used for factuality finetuning typically captures the overall "correctness" of a generated response, though the specific instantiation differs across methods. Models optimized for such reward functions have been shown to improve the factuality of model generations.

According to our conceptual model, the key driver behind the effectiveness of this class of approach is use of reward functions that encourage uncertain or less informative responses over factually incorrect responses. This is because, for models finetuned with RL using these reward functions, the default hedged prediction will correspond to uncertain or less informative responses. As an example, let us consider a reward function that decomposes a response into individual facts, assigns a positive score to every correct fact, a negative score to every incorrect fact, and outputs the sum of the scores [Min et al., 2023]. For this reward function, a randomly fabricated fact in a response is likely to contribute negatively to the reward, whereas omitting the fact from the response does not have an effect on the reward. Thus, the default hedged behavior prescribed by this reward function is to generate shorter, less-informative responses in the face of unfamiliar inputs, which leads to more factual responses overall. In contrast, RL finetuning with a reward function that does not encourage uncertain or less informative responses over incorrect responses (e.g., one that outputs the total number of correct facts in a response) may be less effective at improving the factuality of model generations, because the model's default hedged prediction will instead correspond to random guessing. Our RL experiments in the previous section provide an empirical example illustrating this principle. In the next section, we will illustrate the efficacy of this class of approaches for long-form factuality finetuning.

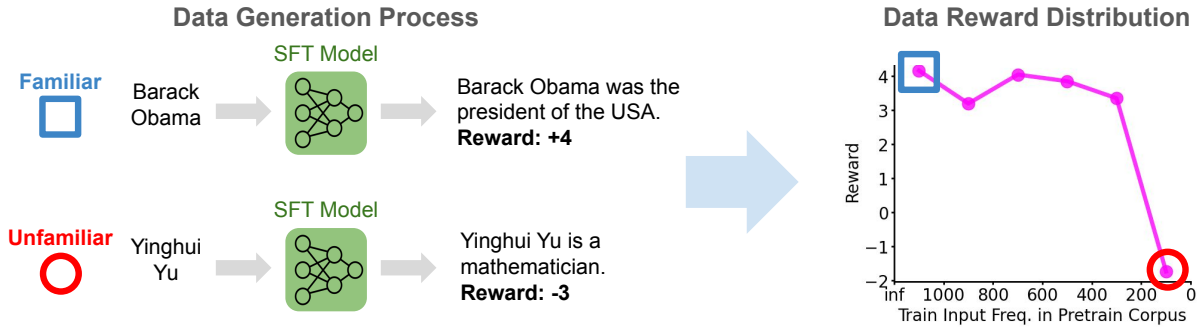

Figure 3.5: On the left, we visualize our proposed data generation process for training conservative reward models. The key insight is to use the same base model for generating samples as the model used to finetune the reward model, because examples from this data generation process tend to be more factually correct for familiar inputs, and more incorrect for unfamiliar inputs. As a result, unfamiliar examples in the dataset will be associated with lower rewards. We can see this is indeed the case in the plot on the right, where we plot the average reward of samples collected by this procedure.

3.6 Towards Scalable Long-Form Factuality Finetuning

Our ultimate goal is to develop finetuning approaches that enhance the factuality of long-form model generations. While the previously discussed methods offer promising solutions, scaling these approaches to long-form generation tasks presents some challenges. For example, SFT-based approaches require target finetuning responses that linguistically express uncertainty for unfamiliar queries. Constructing these responses for long-form generation tasks often necessitates human labeling, which can be expensive and tedious. Additionally, because different models have varying knowledge bases, this approach would demand custom finetuning datasets for each model, limiting its practicality in real-world scenarios where new models are frequently introduced. RL-based approaches offer an alternative, eliminating the need for model-specific, human-labeled target responses by using rewards to assess the factuality of model-generated text. However, evaluating the factuality of long-form unstructured responses is also difficult (often requiring human labeling), making RL finetuning with the direct use of ground-truth reward supervision impractical. In the following sections, we will develop an RL-based approach for factuality finetuning that addresses these scalability challenges by drawing on our previous insights, and empirically evaluate our approach on two long-form generation tasks: biography generation and book/movie plot generation.

RL Factuality Finetuning with Conservative Reward Models

Analogous to the methodology in RLHF [Ouyang et al., 2022], using a reward model to provide supervision for RL offers a promising way to scalably instantiate factuality finetuning for long-form generation tasks. This is because reward models can be learned from an offline dataset, resulting in far more sample-efficient and manageable source of reward supervision than querying for ground-truth (e.g. human-labeled) rewards during RL finetuning. Using reward models to predict *factuality*, however, presents a new challenge: when presented with unfamiliar query/response pairs during RL finetuning, the reward model itself might lack the internal knowledge to determine the accuracy of a response. For example, a reward model unaware of Yinghui Yu might mistakenly assign a positive reward to a model-generated biography "Yinghui Yu is an American mathematician," even though the response is completely incorrect. Such reward hallucinations are likely to drive the policy towards incentivizing hallucinations, inhibiting the efficacy of the approach.

Existing methods have proposed to mitigate reward model hallucinations by incorporating external knowledge sources into the reward model [Sun et al., 2023]. However, the problem persists when these sources are unavailable. In this section, we propose a principled solution that does not rely on external sources of knowledge. The key insight behind our approach is that, while reward model hallucinations are inevitable, strategically controlling *how* reward models hallucinate can significantly reduce their negative effects on RL factuality finetuning. We will first discuss the desired hallucination behavior for our reward model and then outline a concrete method for training models with this characteristic.

Desired reward model hallucination behavior. To determine the best way for a reward model to respond to unfamiliar inputs, let us revisit a key insight from Sec. 3.5: the success of RL factuality finetuning relies on a reward signal that prioritizes uncertain or less informative responses over those containing factual errors. Consider a scenario when the reward model overestimates the reward for a factually incorrect response, resulting in a reward prediction for this incorrect response that is higher than that of a less-informative response, which chooses to abstain from hallucinating. This reward function will thus no longer fulfill the core function of the reward supervision that guaranteed the success of factuality finetuning. RL finetuning with these inflated and erroneous rewards may unintentionally encourage the model to generate more factually incorrect information, consequently undermining the goal of the finetuning. Now suppose the reward model instead *underestimates* reward predictions when it cannot accurately predict the factuality of part of a response. Here, the essential function of our reward supervision remains intact, because the reward prediction will still penalize factually incorrect responses more than less-informative responses. In this scenario, RL finetuning will still steer the model's generations towards less-detailed responses in the face of unfamiliar queries, which is our desired outcome. Therefore, we would like for reward model to consistently underestimate rewards (rather than overestimate) when it cannot determine the factuality of a model response. We will refer to reward models with this behavior as conservative reward models.

Learning conservative reward models. Standard reward models trained on offline datasets do not typically exhibit the desired conservative behavior, because their finetuning data is collected independently of the base model being finetuned. Consequently, the finetuning data might contain examples with high rewards that the base model lacks the knowledge to understand or verify. This causes the model's default hedged prediction to consist of high rewards, which leads to overestimated reward predictions at test time. To learn conservative reward models, we can apply our general strategy for steering LLM hallucinations. More specifically, by steering the reward model's default hedged prediction toward low rewards, we can enable the model to only produce low reward predictions when faced with unfamiliar inputs. Recall from Sec. 3.4 that a model's default prediction is determined by the unfamiliar examples in its finetuning data. Therefore, the key to learning conservative reward models is using a finetuning dataset where all unfamiliar examples have reward labels with low values.

One straightforward way to collect this kind of dataset is to sample responses from the same pretrained model that the reward model is finetuned on, and label these responses with rewards. This strategy is effective, because model samples tend to be more factually correct when queried on familiar examples, and mostly factually incorrect when queried on unfamiliar inputs. Thus, as illustrated in Fig. 3.5, the unfamiliar examples in this dataset tend to be associated with low reward labels, satisfying our desirada. The specific procedure we instantiate, is (1) finetune the base model (same as the one used for the reward model) with standard SFT , (2) generate samples from the finetuned model, (3) label the samples with the ground truth reward, and (4) train the reward model on the labeled samples. Note that while this procedure requires labeling model samples with ground truth rewards in order to train the reward model, this is *much* more sample efficient than using ground truth rewards for RL training, because RL training typically requires much more data than reward model training.

Experiments with Conservative Reward Models

We now present experiments where we investigate the efficacy of RL factuality finetuning with conservative reward models. We compare this approach with standard SFT, as well as RL with standard reward models (trained on data independently collected from the pretrained model). Concretely, the questions we aim to answer with our experiments include: (1) Do conservative reward models trained with our approach produce fewer overestimated reward predictions than standard reward models? (2) Do LLMs finetuned with RL and conservative reward models generate more factual responses than those finetuned with RL with standard reward models and standard SFT?

Experimental setup. We consider two long-form generation tasks in our experiments: biography generation and film/book plot generation. We use the WikiBios [Stranisci et al., 2023] and WikiPlots [Bell, 2017] datasets as sources of names/title and target responses. We use FActScore [Min et al., 2023], an automated retrieval augmentation pipeline, to evaluate

the factuality generated responses and as a source of ground truth reward labels to train reward models. More specifically, given a query and a generated response, FActScore outputs the number of true facts and the number of false facts in the response. Note that querying FActScore is relatively slow and expensive, making the use of FActScore to directly provide rewards in online RL impractical. The ground-truth reward function that we use for RL training assigns a score of +2 for every correct fact and a score of -3 for every incorrect fact in a response, and outputs the sum of the scores as the reward. We use Llama2 7B as the base model for both the finetuned response generation model as well as the reward model.

Our proposed approach uses samples from the base model, in this case Llama2 7B, to train a conservative reward model. To compare against the standard paradigm of using an offline dataset to train reward models, we collect an offline dataset by prompting GPT-3.5 for responses, label them based on their factuality, and use this dataset to train a standard reward model. We use samples from GPT-3.5, because it provides a source of (both factually correct and incorrect) responses that is independent of the model being finetuned. Note that samples from both Llama2 7B and GPT-3.5 were collected using the same set of prompts. The standard SFT models were finetuned directly with the set of target responses provided by WikiBios and WikiPlots. To train the RL models, we initialize the model with the standard SFT model, and continue to do RL finetuning with PPO, using the same set of finetuning prompts as those used for standard SFT finetuning, and the reward models for reward supervision. In order to ensure a fair comparison, we keep all training details fixed across the two RL methods except for the data used to train the reward model. At test time, we evaluate all three methods with queries at different levels of unfamiliarity, measured by the number of times the subject of the query is mentioned in the pretrain corpus. For more experimental details, see Appendix B.4.

Results. To answer our first question, we evaluate both the standard and conservative reward models on held out samples generated from the SFT model. In particular, we used samples from the SFT model because the RL finetuning procedure is initialized with this SFT model, so responses sampled from the SFT model are representative of the kind of responses that the reward model will be asked to score during RL training. In Fig. 3.6, we plot each models' predicted rewards and the ground truth reward, as inputs become more unfamiliar. We can see that for unfamiliar inputs, the standard reward model vastly overestimates the reward, while the conservative reward model does not, showing that the conservative reward models learned with our approach indeed produce more conservative predictions.

To answer our second question, we evaluate standard SFT, as well as RL with a standard reward model and a conservative reward model on a heldout set of queries for each task. In Fig. 3.8, we plot the number of true facts and false facts generated by each model, as inputs become more unfamiliar. We can see that as inputs became more unfamiliar, the standard SFT model generated fewer truth facts and more false facts, as expected. Comparing the RL model trained with the conservative reward model with the standard SFT model, we can see that the RL model generated the same or more true facts while generating significantly

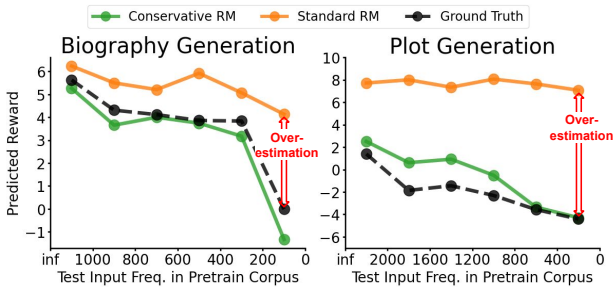

Figure 3.6: Average reward predicted by a standard reward model and a conservative reward model as inputs become more unfamiliar, as well as the average ground truth reward. The standard reward model tends to overestimate rewards as input become more unfamiliar, whereas the conservative reward model does not.

	Std. SFT	RL+ Std. RM	RL+ Csv. RM
Bio	0.47	0.50	**0.59**
Plot	0.45	0.54	**0.80**

Figure 3.7: Average fraction of true facts generated by each model.

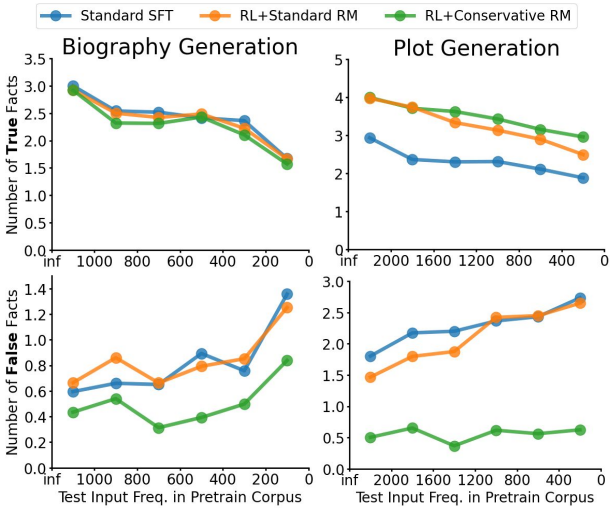

Figure 3.8: Average number of true and false facts generated by models finetuned with standard SFT, RL with a standard reward model, and RL with a conservative reward model, as inputs become more unfamiliar. The responses generated by model finetuned with s conservative reward model consisted of fewer false facts and and equal number or more truth facts.

fewer false facts across all levels of input unfamiliarity. Comparing the two RL models, we can see that while the two generated around the same number of true facts, the model trained with the conservative reward model generated much fewer false facts across all levels of input unfamiliarity. We summarize our results in Table 3.7 with the average percentage of true facts generated by each method. We can see that the RL model trained with the conservative reward model achieved the highest percentage of true facts for both tasks. In Fig. 3.9, we additionally provide some qualitative examples of responses generated by the standard SFT model and the RL model trained with conservative reward model. We can see that as the query became more unfamiliar, responses from the SFT model contained about the same amount of detail but became more factually incorrect, while responses from the RL model with conservative supervision defaulted towards less-informative responses. In conclusion, our results show that RL with conservative reward models is most effective at reducing hallucinations in long-form generation tasks, outperforming standard SFT and RL with standard reward models.

What is the premise of "Home Alone"? (Num mentions in pretrain corpus: **14653**)	**What is the premise of "The Odessa File"?** (Num mentions in pretrain corpus: **551**)	**What is the premise of "Cannonball Run II"?** (Num mentions in pretrain corpus: **3**)
Standard SFT: In 1990, the McCallister family is preparing for a Christmas vacation to Paris, France.	**Standard SFT:** The film opens with a scene of a Nazi SS officer, Colonel Heinrich Müller, being shot by a firing squad in 1945.	**Standard SFT:** The film begins with a brief recap of the events of the first film, with the death of Jenson (Jerry Reed) and the arrest of Jackie (Burt Reynolds).
RL+Conservative RM: Kevin McCallister (Macaulay Culkin) is an eight-year-old boy who is accidentally left behind when his family goes on Christmas vacation to Paris, France.	**RL+Conservative RM:** The Odessa File is a thriller set in post-World War II Germany.	**RL+Conservative RM:** The Cannonball Run II is a cross-country car race.

Figure 3.9: Examples of generated responses from models finetuned with standard SFT and RL with a conservative reward model. False information is highlighted in red.

3.7 Conclusion

In this work, we presented a new conceptual understanding of how LLMs make predictions in the face of unfamiliar inputs, providing us with a principled strategy for controlling how LLMs hallucinate. Using our insights, we propose a scalable approach for improving the factuality of long-form LLM generations, using RL with conservative reward models. Nonetheless, there still remains many open questions and challenges regarding LLM halluciantions. While our conceptual model explains model behavior with entirely unfamiliar examples, many real-world queries fall within a spectrum of partial familiarity. A more nuanced characterization of model predictions in this "middle ground" would be valuable. Furthermore, while our experiments focused on models finetuned for specific applications (e.g., biography generation), extending factuality finetuning to more general prompted generation tasks would be useful. We hope that our work, by offering a deeper understanding of LLM hallucinations, provides a useful step towards building more trustworthy and reliable LLMs.

Chapter 4

Large Language Model Reasoning Generalization

4.1 Introduction

Large language models (LLMs) have demonstrated remarkable problem-solving capabilities, yet the mechanisms by which they learn and generalize remain largely opaque. For instance, consider a set of LLMs, each derived from the same pretrained model and finetuned on the same reasoning dataset but with varying learning rates (Fig. 4.1). While several of these models reach near-perfect accuracy on training data, their test performances were vastly different. This raises the question: *what factors in an LLM's finetuning process lead to differences in its generalization behavior?* Understanding these factors could help us design better training methods that foster genuine reasoning abilities in models, rather than mere pattern matching.

We focus on mathematical reasoning tasks, whose problem structure is particularly amenable for investigating this question. In reasoning tasks, models are trained to generate both a final answer and intermediate reasoning steps. Although each problem has a single correct answer, the reasoning steps in the target solution trace represent just one of many valid ways to solve a problem. Therefore, a model that has memorized the training data is likely to replicate exact reasoning steps from the training data, while a model with general problem-solving skills may produce the correct final answer but follow a different reasoning path. By analyzing model responses on training queries, focusing on both the accuracy of the final answer and the similarity of the response to the target solution trace, we can gain insights into the generalizability of the model's learned solution.

Our findings reveal that, while LLMs often fully memorize the finetuning dataset by the end of training, model predictions for training queries *prior to memorization* are strongly indicative of final test performance. For certain examples, models first learn to generate diverse solution traces (distinct from the target solution trace) that lead to the correct final answer, before later memorizing the target solution trace. For other training examples,

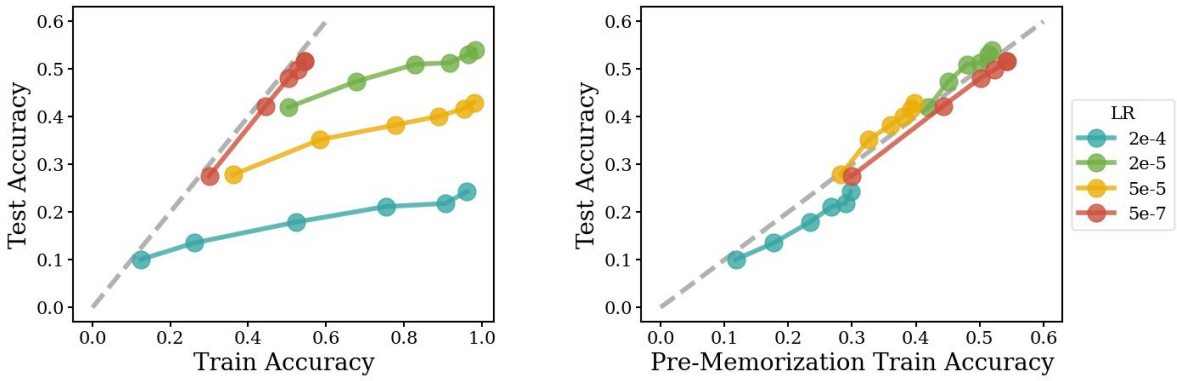

Figure 4.1: Relationship between train accuracy (left), pre-memorization train accuracy (right), and test accuracy for models finetuned on GSM8k using Llama3 8B. Each line represents a training run, and each point represents an intermediate checkpoint. Pre-memorization train accuracy strongly correlates with test accuracy, while train accuracy does not.

models only produce incorrect responses before memorizing the target trace. To capture this distinction, we introduce the concept of *pre-memorization train accuracy*: the highest accuracy a model achieves on a training example through the course of training before exactly memorizing the target solution trace. We find that a model's average pre-memorization train accuracy is highly predictive of the model's test accuracy, as illustrated in Fig. 4.1. Our experiments show that this phenomenon holds across different models (e.g., Llama3 8B [Dubey et al., 2024], Gemma2 9B [Team et al., 2024]), tasks (e.g., GSM8k [Cobbe et al., 2021], MATH [Hendrycks et al., 2021b]), dataset sizes, and hyperparameter settings, with coefficients of determination around or exceeding 0.9.

We further find that the pre-memorization train accuracy can provide insights into the robustness of model predictions at a per-example level. For train examples with low pre-memorization accuracies, adding small perturbations to the training prompt causes the accuracy of model predictions to significantly degrade. In contrast, for train examples with high pre-memorization accuracies, models are generally able to maintain high performance under perturbations. Thus, by measuring pre-memorization accuracy, we can identify specific training examples for which a model's predictions are not robust, which can inform targeted improvements to the training strategy. As an example, we leverage our findings to guide data curation. Our experiments show that training on data distributions that prioritize examples with low pre-memorization accuracy leads to a 1.5-2$\times$ improvement in sample efficiency over i.i.d sampling, and outperforms other standard curation techniques.

The main contributions of this work are as follows: (1) we introduce the concept of pre-memorization train accuracy, and show that it is highly predictive of test accuracy for LLM reasoning problems, (2) we show that pre-memorization train accuracy can also predict the robustness of individual model predictions for train examples, and (3) we leverage our obser-

vations to improve the sample efficiency of data curation. By offering a deeper understanding of how specific aspects of a model's learning dynamics shape its generalization behavior, we hope our work can bring about more targeted and principled interventions for improving a model's reasoning capabilities.

4.2 Related Works

A number of works have studied the phenomenon of memorization during training, but consider different definitions of memorization. One definition quantifies memorization with the "leave-one-out" gap, i.e., how much a model's prediction for an example changes if we were to remove it from the training data [Feldman and Zhang, 2020, Arpit et al., 2017, Zhang et al., 2021]. Using this definition, some works argue that more memorization during training leads to worse generalization [Bousquet and Elisseeff, 2000], while others contend that memorization is actually necessary for generalization in long-tail distributions [Feldman, 2020]. These works generally produce bounds on generalization error under worst cases instances within some class of training distributions. In contrast, our work presents a direct, empirical connection between learning dynamics and generalization without relying on the computationally expensive "leave-one-out" metric. In the context of language models, others have defined memorized examples as those where the model's output closely matches examples in the training data [Carlini et al., 2021, Tirumala et al., 2022, Inan et al., 2021, Hans et al., 2024], which is similar to our definition of memorization. However, these works mainly focus on privacy and copyright concerns, rather than connections to generalization.

Beyond memorization, a number of prior works have studied how other aspects of the learning process relate to generalization. Some works focus on metrics related to model complexity, such as VC dimension or parameter norms [Neyshabur et al., 2015b, Bartlett et al., 2019], while other works focus on empirically motivated measures, such as gradient noise [Jiang et al., 2019] or distance of trained weights from initialization [Nagarajan and Kolter, 2019a]. Jiang et al. [2019] conducted a comprehensive comparison of these measures and found that none were consistently predictive of generalization, though their work primarily focused on image classification. Other approaches have used unlabeled, held-out data to predict generalization, leveraging metrics such as the entropy of model predictions or the disagreement between different training runs [Garg et al., 2022, Platanios et al., 2016, Jiang et al., 2021]. Our findings show that pre-memorization accuracy can be a much stronger predictor of generalization in LLM reasoning tasks.

Finally, our work seeks to improve data curation, which has also been studied in a number of prior works. Specific to LLM finetuning, prior approaches largely fall into three categories: optimization-based, model-based, and heuristic-based approaches. Optimization-based methods frame data selection as an optimization problem, where the objective is model performance, and the search space consists of the training data distribution [Engstrom et al., 2024, Grosse et al., 2023]. Model-based approaches, on the other hand, leverage characteristics of the learning process [Mekala et al., 2024, Liu et al., 2024], such as comparing

the perplexity of examples [Li et al., 2023b]. Lastly, heuristic-based methods rely on simpler criteria, such as difficulty scores generated by GPT, to classify desirable training data [Chen et al., 2023, Lu et al., 2023, Zhao et al., 2023]. Our data curation approach aligns most closely with model-based strategies, as we use the model's pre-memorization accuracy, a characteristic of the learning process, to inform the selection of training examples. Our experiments show that pre-memorization train accuracy can serve as an effective metric for data curation which outperforms previous approaches.

4.3 Preliminaries

We focus on training LLMs to perform reasoning tasks via finetuning. We are provided with a training dataset $D_{\text{train}} = \{(x_i, y_i)\}$, where queries x_i are drawn from $P(x)$ and solution traces y_i are drawn from $P(y|x)$. We assume the test dataset, D_{test}, is generated from the same distribution as the training data. The model is finetuned by minimizing next-token prediction loss. We denote the finetuned model as $f_\theta(y|x)$, and model predictions as $\hat{y} \sim f_\theta(y|x)$.

In reasoning tasks, solution traces y consist of both intermediate reasoning steps and a final answer, denoted as $\mathbf{Ans}(y)$. The goal of reasoning tasks is for the model to generate solution traces with the correct final answer when faced with previously unseen queries. We measure the accuracy of model samples for a given query x_i using $\mathbf{Acc}(f_\theta(y|x_i), y_i) = \mathbb{E}_{\hat{y}_i \sim f_\theta(y|x)}[\mathbb{1}(\mathbf{Ans}(\hat{y}_i) = \mathbf{Ans}(y_i))]$. In our experiments, we approximate this accuracy by sampling from the model with a temperature of 0.8 and averaging the correctness attained by the samples.

While different solution traces drawn from $P(y|x)$ should all have the same final answer, the target solution trace y_i of an example represents only one of many valid solution traces for solving x_i. Thus, model samples for a train query may contain reasoning steps that differ from the target solution trace, while still arriving at the correct final answer. To quantify this difference, we will measure the distance between a model's prediction $f_\theta(y|x_i)$ and the target reasoning trace y_i with perplexity, defined as $\mathbf{Perp}(f_\theta(y|x_i), y_i) = \exp(\frac{-1}{n_i} \log(f_\theta(y_i|x_i)))$, where n_i is the number of tokens in y_i.

4.4 Connecting Learning Dynamics to Generalization

In this section, we explore the relationship between a model's learning dynamics during finetuning and its ability to generalize. Our findings show that, while models tend to memorize most of the training data after some number of epochs, their generated samples display varying levels of accuracy before memorization occurs. We find that this accuracy before memorization has a strong connection to the model's downstream generalization behavior.

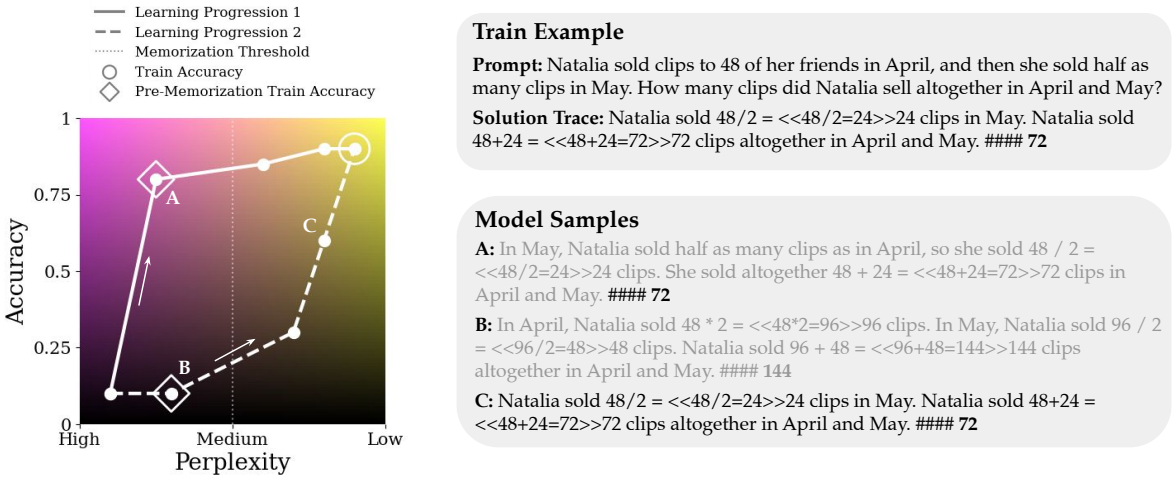

Figure 4.2: Visualizations of different learning progressions, as measured by the accuracy of model samples (light vs. dark) and the perplexity of target solution traces under model predictions (pink vs. yellow). Right side presents examples of model samples with (A) high accuracy+high perplexity, (B) low accuracy+high perplexity, and (C) high accuracy+low perplexity. Black text represents exact match with the target solution trace, while grey text represents parts that do not match.

Characterizing the Learning Dynamics of LLM Reasoning Finetuning

We begin by more precisely characterizing an LLM's learning process when finetuning on reasoning tasks. We focus on two key aspects of the model's behavior when presented with train queries: 1) whether the model's samples arrive at the correct final answer, and 2) the distance between the model's prediction and the target solution trace, measured by perplexity. These two metrics, visualized in Fig. 4.2, offer different perspectives on the model's behavior, because while there is only one correct final answer for each query, there may exist many different valid reasoning traces. Tracking both metrics through the course of training allows us to measure how effectively the model is able to solve training queries, and the extent to which this is accomplish by replicating the target solution trace.

In Fig. 4.3, we visualize the learning progression, as characterized by the two metric described above, for three models finetuned on GSM8K. Each model is trained for six epochs, with a distinct peak learning rate that decays to zero by the end of training. As expected, training accuracy improves over time as the model minimizes the loss (color gradient from dark to light), and the distance between predictions and target solution traces decrease (from pink to yellow). For some learning rate settings, models approach near-perfect accuracy by the end of training, and their predictions closely match the target reasoning traces (mostly yellow in bottom row). However, during early stages of training, we observe significant

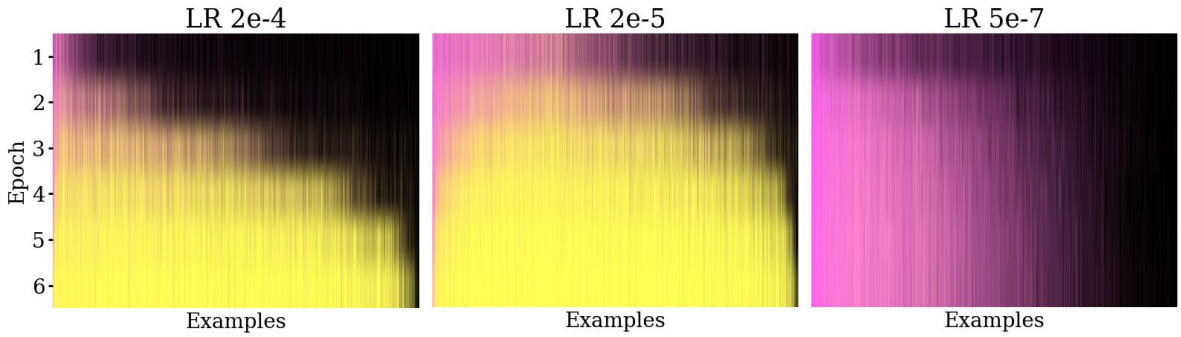

Figure 4.3: Predictions of 3 different models through the course of training. The x-axis represents individual training examples, the y-axis represents the epoch of training, and the color represents model predictions for each example in terms of accuracy and perplexity (legend in Fig. 4.2).

differences in model behavior. For some train queries, models initially produce incorrect samples (black), and then directly transition to replicating the target trace (yellow). For other examples, models first learn to generate correct answers with solution traces that differ from the target trace (pink), before later transitioning to fully replicating the target trace (yellow).

In this work, we will refer to model predictions with low distance to target solution traces as *memorization*. We can see that when finetuned with different learning rates, different models exhibit different capacities for generating accurate samples before memorizing target solution traces (amount of pink). For models with low accuracy before memorization, they may be largely learning verbatim mappings from training queries to target traces, which would not generalize to new queries. In contrast, models with high accuracy before memorization demonstrate an ability to arrive at correct answers through varied reasoning paths, suggesting that they have developed more generalizable problem-solving capabilities.

To better quantify this phenomenon, we introduce a new metric called pre-memorization accuracy. We consider a train example $(x_i, y_i) \in D_{\text{train}}$ to be memorized by $f_\theta(y|x)$ if $\textbf{Perp}(f_\theta(y|x_i), y_i) \leq p$, where p is a threshold (fixed across examples). We further define a modified measure of accuracy, whose value is masked to zero if the model's prediction for that example is considered memorized, as follows:

$$\textbf{MaskedAcc}(f_\theta(y|x_i), y_i, p) = \textbf{Acc}(f_\theta(y|x_i), y_i) \cdot \mathbb{1}[\textbf{Perp}(f_\theta(y|x_i), y_i) > p].$$

Now let f_{θ_m} denote the model at epoch m of training. Using our definition of masked accuracy, we define the **pre-memorization accuracy** as follows:

$$\textbf{PreMemAcc}(f_{\theta_{1:m}}(y|x_i), y_i, p) = \min \left\{ \max_{1 \leq m' \leq m} \textbf{MaskedAcc}(f_{\theta_{m'}}(y|x_i), y_i, p), \textbf{Acc}(f_{\theta_m}(y|x_i), y_i) \right\}$$

This quantity can be roughly interpreted as the best accuracy that the model achieves for a training prompt thus far in training before it memorizes the target trace. Unlike standard

accuracy or masked accuracy, which evaluate performance at specific training checkpoints, pre-memorization accuracy evaluates the entire training process up to epoch m. There is an additional minimum taken with the accuracy of model predictions at epoch m, which compensates for examples whose accuracies decrease through training (though this is uncommon).

Pre-Memorization Train Accuracy Strongly Predicts Test Accuracy

We next use pre-memorization accuracy to analyze the connection between learning dynamics and downstream generalization. We find that **a model's average pre-memorization train accuracy is highly predictive of its test accuracy** across a variety of training runs and checkpoints. More concretely, we find that there exists a value of p for which a model's average pre-memorization train accuracy, $\mathbb{E}_{D_{\text{train}}}[\textbf{PreMemAcc}(f_{\theta_{1:m}}(y|x_i), y_i, p)]$, closely approximates the model's test accuracy, $\mathbb{E}_{D_{\text{test}}}[\textbf{Acc}(f_{\theta_m}(y|x_i), y_i)]$. The value of the memorization threshold p is fixed across examples and training parameters, but may need to be recalibrated for different tasks or models. We calibrate p by sweeping across a range of values (see Appendix C.1).

In Fig. 4.4, we plot the pre-memorization training accuracy and test accuracy across different training runs. We used Llama3 8B and Gemma2 9B as base models and GSM8K and MATH as the reasoning tasks. To evaluate different generalization behaviors, we finetuned the models by adjusting the peak learning rate (ranging from 5e-7 to 5e-4), the number of training epochs (1, 3, 6), and the dataset size (full, half, or quarter of the original dataset). We use the same value for p within each plot. A full list of the training runs in our experiments and other details can be found in Appendix C.2. We observe a strong linear relationship between pre-memorization training accuracy and test accuracy, with the results closely following the $y = x$ line across different models, tasks, and hyperparameter settings. More quantitatively, the coefficients of determination associated with each plot are 0.94 (GSM8k Llama), 0.95 (MATH Llama), 0.97 (GSM8k Gemma), and 0.88 (MATH Gemma). Our results show that pre-memorization training accuracy is a reliable predictor of test accuracy.

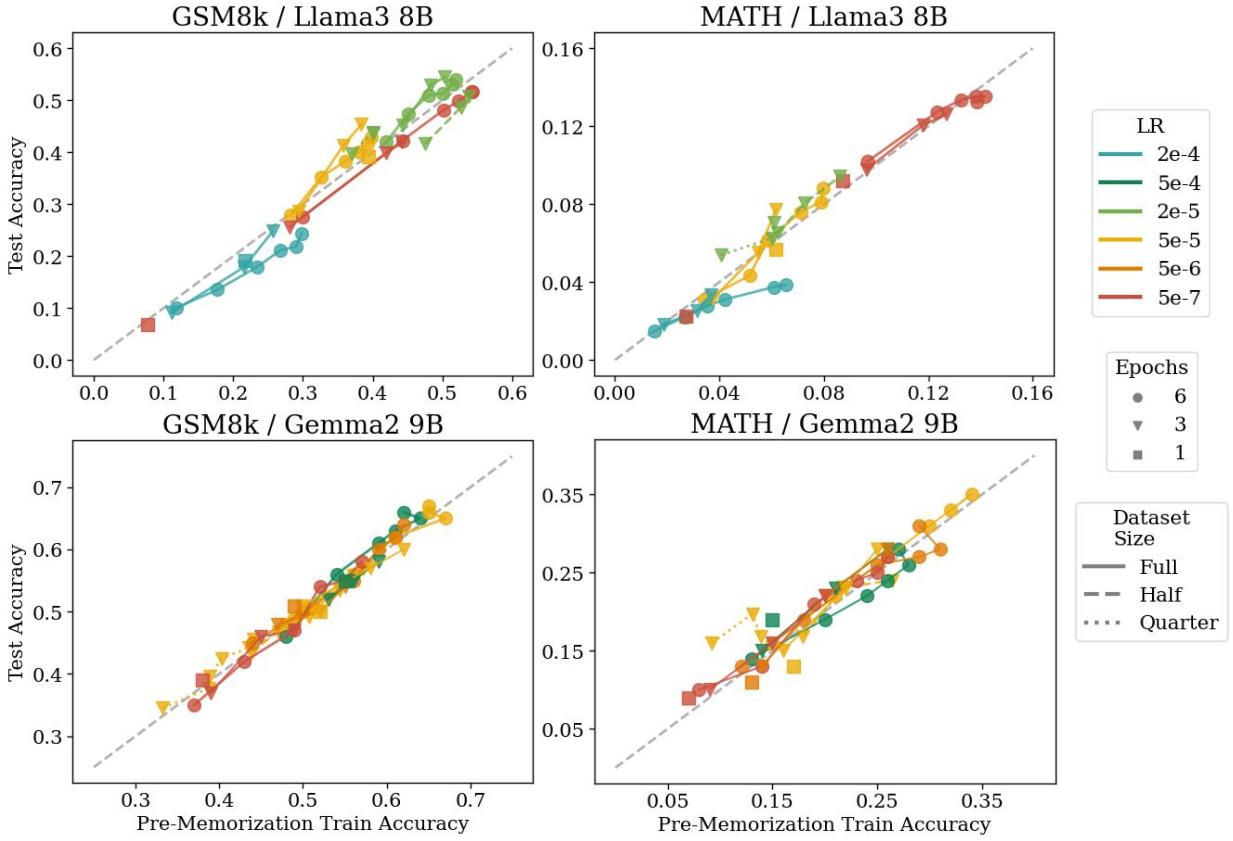

Figure 4.4: Evaluating the relationship between pre-memorization train accuracy and test accuracy. Each line corresponds to a training run, and each marker corresponds to a specific checkpoint. Pre-memorization train accuracy strongly predict test accuracy across tasks, models, and training settings.

As discussed in Section 4.2, various metrics have been proposed in previous studies to predict the generalization gap, the difference between train and test accuracy. In Fig. 4.5, we compare several of these existing metrics, including gradient variance [Jiang et al., 2019], distance between current model weights and initialization [Nagarajan and Kolter, 2019a], and an estimate of test accuracy via Average Thresholded Confidence (ATC) [Garg et al., 2022] (details in Appendix C.3). The correlation coefficients associated with each metric (left to right) are 0.98, -0.72, 0.59, -0.04, which shows that the prior metrics do not correlate as strongly with test accuracy as our proposed metric.

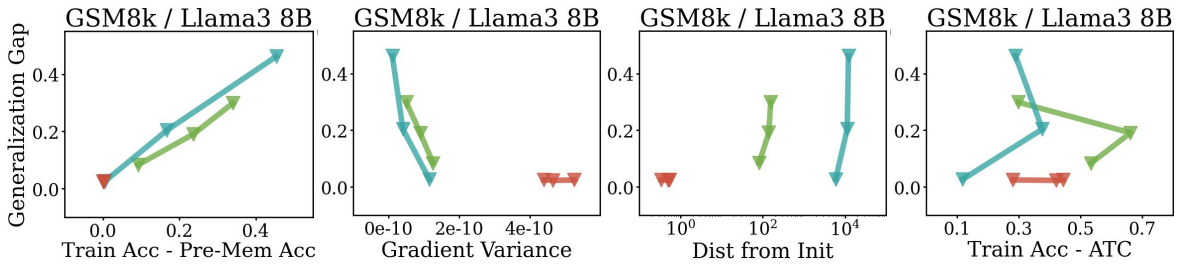

Figure 4.5: Evaluating different generalization metrics vs. the ground truth generalization gap for models finetuned on GSM8k using Llama3 8B (legend in Fig. 4.4).

4.5 Per-Example Analysis of Generalization

In this section, we go beyond aggregate test accuracy and show that tracking per-example pre-memorization accuracy offers a window into the model's behavior at the level of individual training examples. Specifically, we find that the pre-memorization train accuracy of a given example is predictive of the robustness of the model's prediction for that example. This example-level accuracy helps us identify subsets of the training data for which the model struggles to learn robust solutions and offers opportunities to improve training through targeted interventions. We explore how this insight can inform data curation strategies, showing that prioritizing examples with low pre-memorization train accuracy during data collection can lead to significant improvements over i.i.d. data collection and other common data curation methods.

Predicting Model Robustness with Pre-Memorization Train Accuracy

We begin by examining the relationship between an individual example's pre-memorization train accuracy and the robustness of the model's predictions for that example. Our findings show that **model predictions tend to be less robust for train examples with low pre-memorization accuracy**.

To assess the robustness of model predictions, we analyze how the model responds to small perturbations in the input prompt. We present the model with both the original training queries, as well as training queries appended with short preambles to the solution trace—phrases such as "First" or "We know that"—that deviate from the target solution trace, which we visualize in Fig. 4.6. Because these generic phrases are plausible preambles to valid reasoning traces, we would expect a model which has learned a robust solution to an example to still be able to arrive at the correct final answer. In contrast, if the model is unable to produce the correct final answer given these generic phrases, then the model is likely to have learned to only regurgitate the training response.

Figure 4.6: Visualization of the robustness of model predictions to perturbations in the prompt, including the original training prompt (purple), original prompt + "First" (pink), and original prompt + "We know that" (teal). A robust model prediction would arrive at the correct final answer even if the perturbations changes the reasoning steps. In contrast, a non-robust model prediction produces incorrect final answer when the prompt diverges from the training data.

In Fig. 4.6, we show the prediction behavior of two models, both trained for six epochs with a learning rate of 2e-5, on the GSM8K and MATH datasets. We can see that while model predictions are near-perfect for unaltered training prompts, their accuracy significantly degrades when presented with perturbed prompts. Furthermore, we see that the accuracy of train examples with low pre-memorization train accuracy tends to degrade much more than those with high pre-memorization train accuracy. These findings suggest that pre-memorization train accuracy can predict the robustness of model predictions for individual train examples. Note that while our perturbation analysis makes use of manually-constructed, task-dependent preambles, pre-memorization train accuracy does not require any domain knowledge. Therefore, pre-memorization train accuracy provides a practical way to identify fragile examples where the model may have learned overly specific or non-robust patterns, which offers practical applications for improving model generalization.

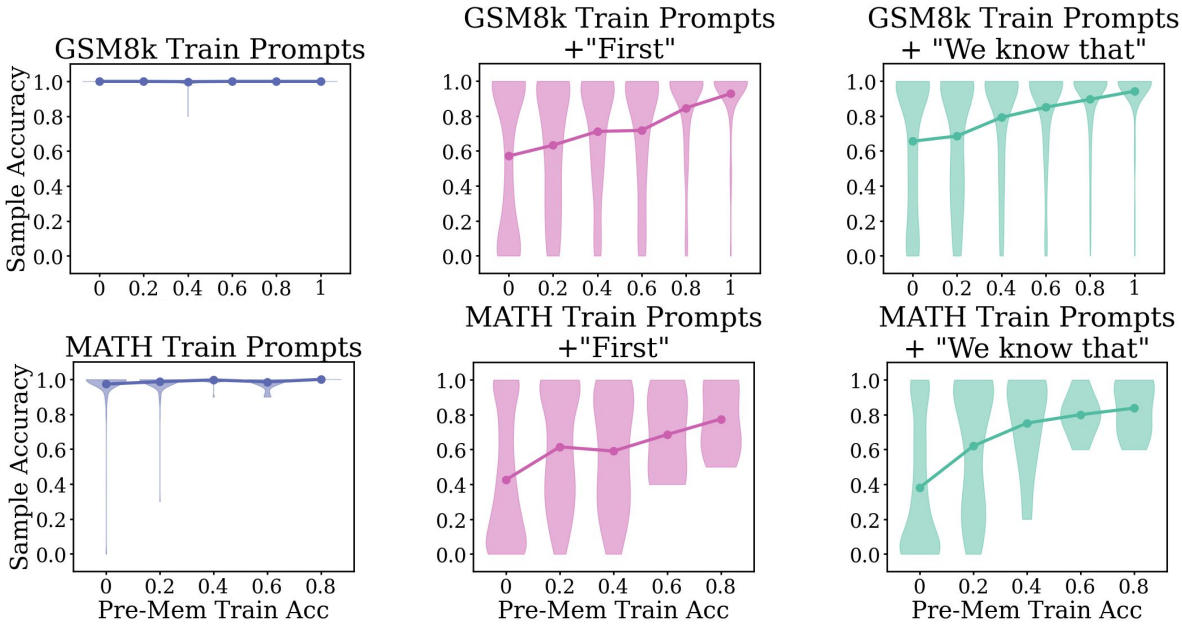

Figure 4.7: Accuracies of model samples (y-axis) when faced with the original prompt (left) and prompts with perturbations (middle, right). The x-axis represents bins of pre-memorization train accuracies associated with each prompt. Solid line denotes the average, and violins denote distributions within each bin. While the accuracy of model samples is almost perfect when faced with original prompts, it significantly degrades when faced with prompts with perturbations. Furthermore, the degradation of accuracy is much more significant for train examples with low pre-memorization accuracy than those with high pre-memorization accuracy, showing that per-example pre-memorization train accuracy can provide insight into the robustness of a model's individual predictions.

Curating Data with Pre-Memorization Train Accuracy

By offering insights into the robustness of individual model predictions, pre-memorization train accuracy can provide targeted guidance for improving a model's generalization. In this section, we explore data curation as a practical application of our findings. Prior work has suggested that focusing on "harder" examples, where the model struggles to learn robust solutions, can lead to more sample-efficient improvements [Li et al., 2023b, Chen et al., 2023]. However, identifying useful metrics for determining example difficulty remains an open challenge. We investigate the use of pre-memorization train accuracy as a metric for guiding data curation, and find that it outperforms i.i.d. sampling and other standard data curation approach in sample efficiency for reasoning tasks.

We will first more precisely define our data curation problem. Given an existing set of N training examples with queries distributed as $P(x)$, we aim to collect N' examples, denoted as D'_{train}, to augment the dataset. The goal is to specify a new distribution $P'(x)$

Algorithm 1 Our Data Collection Process

Input: $N' = N'_1 + \cdots + N'_n$, t
Output: Updated dataset D'_{train}
Initialize $D'_{\text{train}} = \{\}$
for $i = 1$ to n **do**
    Train model on $D_{\text{train}} + D'_{\text{train}}$
    Evaluate model on D_{train} and compute pre-memorization accuracy for each example
    Set $P'_i(x)$ as the distribution of examples with pre-memorization accuracy below t
    Collect N'_i new examples from $P'_i(x)$ and add them to D'_{train}
end for

that maximizes the test performance of a model trained on both the original and the newly collected examples. While defining the true distribution of queries can be challenging, we assume that by approximating it with an empirical distribution from the current dataset, we can collect new data with similar properties. In our experiments, we take D_{train} to be the original dataset, and collect new examples by using GPT to rephrase examples in the original dataset, similar to the procedure in [Setlur et al., 2024]. By only collecting new examples that derive from the specified empirical distribution, we can ensure the new dataset approximates $P'(x)$. This setup can also be used when collecting new human-generated data, by providing the specified empirical distribution of examples as references for human labelers.

Our approach for data collection prioritizes examples with low pre-memorization accuracy. First, we calculate the pre-memorization accuracy for each example in the current dataset and then define $P'(x)$ as the distribution of examples whose pre-memorization accuracy falls below a certain threshold t. We then collect new data according to this distribution. If N' is large, we can split the data collection process into multiple iterations ($N'_1 + ... + N'_n = N'$). In each iteration, we collect N'_i new examples according to $P'_i(x)$, retrain a model on the combined dataset, calculate the pre-memorization accuracy with the model, and update $P'_{i+1}(x)$ for the next round of data collection. This process is summarized in Algorithm 1.

We compare our strategy to i.i.d. sampling and two existing approaches commonly used in data curation. Both of these approaches propose a metric of example difficulty and prioritize difficult examples during data collection. The first metric, called Instruction-Following Difficulty (IFD) [Li et al., 2023b], computes the ratio between the perplexity of training labels given inputs and the perplexity of only labels using a model finetuned for the task. The second metric uses heuristic notions of difficulty measured by external sources such as humans or more capable models [Chen et al., 2023, Lu et al., 2023, Zhao et al., 2023]. For GSM8K, we use the number of lines in the target solution traces as a heuristic for difficulty, while for MATH, we use the difficulty levels provided in the dataset itself.

In Fig. 4.8, we evaluate the different data curation approaches for finetuning on GSM8k with Llama3 8B and MATH levels 1-3 with DeepSeekMath 7B [Shao et al., 2024]. Our approach outperforms all three prior approaches, achieving 2× the sample efficiency for the

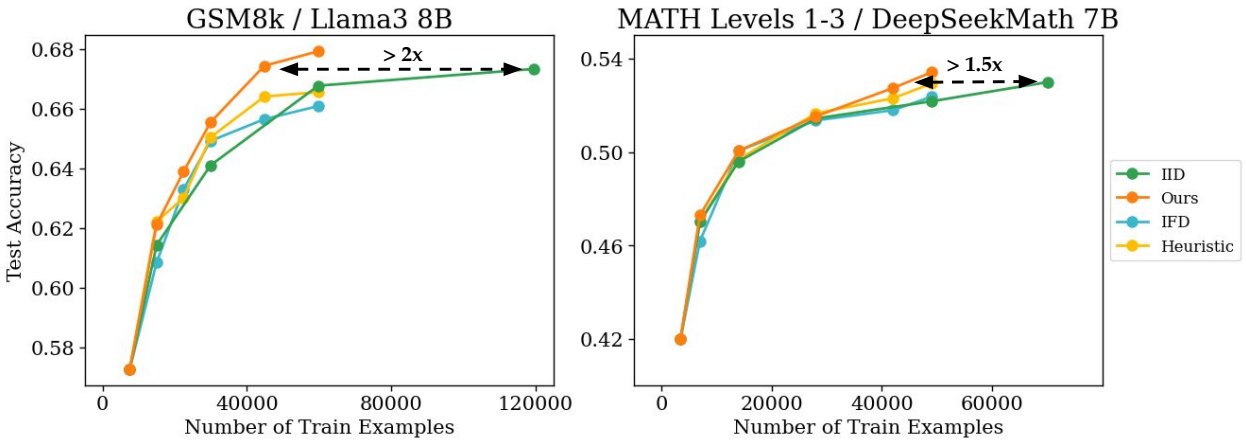

Figure 4.8: Comparison of different approaches for data curation. Each line represents a different data curation approach at varying scales of training dataset size, and each point represents a different training run. Our approach acheived the best sample efficiency compared to the other approaches.

same target test accuracy compared to i.i.d scaling in GSM8k, and 1.5× sample efficiency on MATH levels 1-3. Furthermore, we find the gap in performances increases with dataset size, which suggests that better data curation metrics may become more important as models become more capable. These results highlight the effectiveness of pre-memorization accuracy as a criterion for targeted data collection, leading to enhanced generalization with fewer data points. We provide more details about our implementations in Appendix C.4.

4.6 Conclusion

Our work studies the relationship between learning dynamics and generalization in LLMs finetuned for reasoning tasks. We introduce the concept of pre-memorization train accuracy and show that it is a strong predictor of its test accuracy. We further show that a model's per-example pre-memorization train accuracy can be an indicator of the robustness model predictions for those examples. Finally, we leverage this insight for data curation, and show that prioritizing examples with low pre-memorization train accuracy can be more effective than i.i.d. data scaling and other data curation techniques. We hope that by providing a way attribute a model's generalization to specific aspects of the training process, our work can enable the design of more effective and principled training strategies.

Chapter 5

Conclusion

This dissertation explored the relationship between the training processes of deep neural networks, particularly large language models, and their ability to generalize beyond the confines of their training data. Motivated by the need for more reliable and predictable AI systems, we investigated the gap between optimizing performance on training examples and achieving robust generalization, encompassing both in-distribution and out-of-distribution scenarios. Our core objectives were twofold: first, to characterize how specific elements of the learning recipe shape generalization and extrapolation behavior, and second, to leverage these insights to devise strategies for steering model behavior towards improved performance on unseen data.

Our investigation yielded several key insights across different facets of deep learning generalization. In Chapter 2, we studied out-of-distribution generalization in conventional neural networks, where our findings challenged the conventional wisdom that deep neural networks extrapolate erratically and unpredictably. We presented theoretical and empirical evidence demonstrating that network extrapolation often exhibits structure, tending towards a predictable "optimal constant solution" linked to the training data distribution and loss function. This finding provides a more nuanced understanding of OOD behavior and offers a strategy for building risk-aware systems, setting the stage for analyzing the complexities of LLMs.

Transitioning to LLMs, Chapter 3 studies the pervasive issue of hallucination, specifically how finetuning influences factual accuracy when models encounter unfamiliar queries at test-time. We identified a crucial link between a model's hallucination behavior and the "unfamiliar examples" within its finetuning dataset – examples introducing concepts beyond the pretrained model's knowledge base. Our findings revealed that hallucinations often mirror the responses associated with these unfamiliar examples, suggesting a direct mechanism through which finetuning shapes OOD factual recall. This insight provides a practical lever: by carefully managing the supervision of unfamiliar data, we can potentially steer models towards more truthful responses, such as expressing uncertainty ("I don't know"), thereby enhancing their reliability in question-answering contexts.

Chapter 4 shifted the focus from factual recall to the acquisition of generalizable reason-

ing skills, using mathematical problem-solving as a case study. Unlike fact retrieval, mathematical reasoning requires applying learned rules to entirely novel problems. Our analysis demonstrated how learning dynamics can reveal how reasoning capabilities emerge during finetuning. We introduced the concept of "pre-memorization train accuracy" – a metric capturing performance before rote memorization of training steps occurs – and demonstrated its effectiveness in predicting test generalization. This establishes a connection between observable training behavior and final test performance, offering a tool to guide improvements in the training process. We showcased its utility by demonstrating that data curation guided by this metric can significantly enhance sample efficiency for learning reasoning tasks.

Collectively, these studies contribute to a deeper understanding of generalization in deep learning. While a model's generalization may often seem random and unpredictable, one takeaway of this thesis is that models often exhibit systematic patterns in their behavior that can be traced back to the training process. We demonstrated specific instances of this connection, linking generalization outcomes to characteristics of the training data, the choice of loss function, and the learning dynamics of training. However, these studies primarily focus on a limited set of settings, e.g., LLM finetuning where train and test examples are drawn from the same distribution. Broadening this analysis to encompass the diverse regimes of modern LLM pipelines presents very interesting directions for future work; for instance, studying the interaction between pretraining and RL finetuning, the compositional effects of multitask and multimodal training, and the change in generalization behavior as models and datasets scale to ever increasing sizes. Overall, we believe that tracing generalization behavior back to aspects of the training recipe will unlock insights that allow us to more effectively steer how models generalize, ultimately leading to more reliable and interpretable AI systems. This thesis represents a step towards that goal.

Bibliography

Ayush Agrawal, Lester Mackey, and Adam Tauman Kalai. Do language models know when they're hallucinating references? *arXiv preprint arXiv:2305.18248*, 2023.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. Implicit regularization in deep matrix factorization. *Advances in Neural Information Processing Systems*, 32, 2019.

Devansh Arpit, Stanisław Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *International conference on machine learning*, pages 233–242. PMLR, 2017.

Amos Azaria and Tom Mitchell. The internal state of an LLM knows when its lying. *arXiv preprint arXiv:2304.13734*, 2023.

Christina Baek, Yiding Jiang, Aditi Raghunathan, and J Zico Kolter. Agreement-on-the-line: Predicting the performance of neural networks under distribution shift. *Advances in Neural Information Processing Systems*, 35:19274–19289, 2022.

Randall Balestriero, Jerome Pesenti, and Yann LeCun. Learning in high dimension always amounts to extrapolation. *arXiv preprint arXiv:2110.09485*, 2021.

Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. *Advances in neural information processing systems*, 30, 2017.

Peter L Bartlett, Nick Harvey, Christopher Liaw, and Abbas Mehrabian. Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks. *Journal of Machine Learning Research*, 20(63):1–17, 2019.

Yonatan Belinkov and James Glass. Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72, 2019.

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

Jon Bell. Wikiplots, 2017. URL `https://github.com/markriedl/WikiPlots`.

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19, 2006.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79:151–175, 2010.

Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth. Learnability and the vapnik-chervonenkis dimension. *Journal of the ACM (JACM)*, 36(4):929–965, 1989.

Olivier Bousquet and André Elisseeff. Algorithmic stability and generalization performance. *Advances in neural information processing systems*, 13, 2000.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with GPT-4. *arXiv preprint arXiv:2303.12712*, 2023.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models. In *USENIX Security Symposium*, 2021.

Nontawat Charoenphakdee, Zhenghang Cui, Yivan Zhang, and Masashi Sugiyama. Classification with rejection based on cost-sensitive classification. In *International Conference on Machine Learning*, pages 1507–1517. PMLR, 2021.

Li-Pang Chen. Mehryar mohri, afshin rostamizadeh, and ameet talwalkar: Foundations of machine learning: The mit press, cambridge, ma, 2018, 504 pp., cdn 96.53($hardback$), $isbn$9780262039406, 2019.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, et al. Alpagasus: Training a better alpaca with fewer data. *arXiv preprint arXiv:2307.08701*, 2023.

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James Glass, and Pengcheng He. DoLa: Decoding by contrasting layers improves factuality in large language models. *arXiv preprint arXiv:2309.03883*, 2023.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Edo Cohen-Karlik, Avichai Ben David, Nadav Cohen, and Amir Globerson. On the implicit bias of gradient descent for temporal extrapolation. In *International Conference on Artificial Intelligence and Statistics*, pages 10966–10981. PMLR, 2022.

Corinna Cortes, Giulia DeSalvo, and Mehryar Mohri. Learning with rejection. In *Algorithmic Learning Theory: 27th International Conference, ALT 2016, Bari, Italy, October 19-21, 2016, Proceedings 27*, pages 67–82. Springer, 2016.

Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 113–123, 2019.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Thomas G Dietterich and Alex Guyer. The familiarity hypothesis: Explaining the behavior of deep open set methods. *Pattern Recognition*, 132:108931, 2022.

Simon S Du, Wei Hu, and Jason D Lee. Algorithmic regularization in learning deep homogeneous models: Layers are automatically balanced. *Advances in neural information processing systems*, 31, 2018.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Logan Engstrom, Axel Feldmann, and Aleksander Madry. Dsdm: Model-aware dataset selection with datamodels. *arXiv preprint arXiv:2401.12926*, 2024.

Vitaly Feldman. Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–959, 2020.

Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. *Advances in Neural Information Processing Systems*, 33:2881–2891, 2020.

Leo Feng, Mohamed Osama Ahmed, Hossein Hajimirsadeghi, and Amir H Abdi. Towards better selective classification. In *The Eleventh International Conference on Learning Representations*, 2011.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

Tomer Galanti, Zachary S Siegel, Aparna Gupte, and Tomaso Poggio. Sgd and weight decay provably induce a low-rank bias in neural networks. *arxiv*, 2022.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The Pile: An 800GB dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Luyu Gao, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Arun Tejasvi Chaganty, Yicheng Fan, Vincent Zhao, Ni Lao, Hongrae Lee, Da-Cheng Juan, et al. RARR: Researching and revising what language models say, using language models. In *ACL*, 2023.

Saurabh Garg, Sivaraman Balakrishnan, Zachary C Lipton, Behnam Neyshabur, and Hanie Sedghi. Leveraging unlabeled data to predict out-of-distribution performance. *arXiv preprint arXiv:2201.04234*, 2022.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. *Advances in neural information processing systems*, 30, 2017.

Yoav Goldberg. Reinforcement learning for language models, 2023. URL `https://gist.github.com/yoavg/6bff0fecd65950898eba1bb321cfbd81`.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Arthur Gretton, Alex Smola, Jiayuan Huang, Marcel Schmittfull, Karsten Borgwardt, and Bernhard Schölkopf. Covariate shift by kernel mean matching. *Dataset shift in machine learning*, 3(4):5, 2009.

Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, et al. Studying large language model generalization with influence functions. *arXiv preprint arXiv:2308.03296*, 2023.

Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. *arXiv preprint arXiv:2007.01434*, 2020.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017.

Fredrik K Gustafsson, Martin Danelljan, and Thomas B Schön. How reliable is your regression model's uncertainty under real-world distribution shifts? *arXiv preprint arXiv:2302.03679*, 2023.

Abhimanyu Hans, Yuxin Wen, Neel Jain, John Kirchenbauer, Hamid Kazemi, Prajwal Singhania, Siddharth Singh, Gowthami Somepalli, Jonas Geiping, Abhinav Bhatele, et al. Be like a goldfish, don't memorize! mitigating memorization in generative llms. *arXiv preprint arXiv:2406.10209*, 2024.

Moritz Hardt, Ben Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *International conference on machine learning*, pages 1225–1234. PMLR, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 41–50, 2019.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. *arXiv preprint arXiv:1812.04606*, 2018.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. *ICCV*, 2021a.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021b.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Haiwen Huang, Zhihan Li, Lulu Wang, Sishuo Chen, Bin Dong, and Xinyu Zhou. Feature space singularity for out-of-distribution detection. *arXiv preprint arXiv:2011.14654*, 2020.

Minyoung Huh, Hossein Mobahi, Richard Zhang, Brian Cheung, Pulkit Agrawal, and Phillip Isola. The low-rank simplicity bias in deep networks. *arXiv preprint arXiv:2103.10427*, 2021.

Daksh Idnani, Vivek Madan, Naman Goyal, David J Schwab, and Shanmukha Ramakrishna Vedantam. Don't forget the nullspace! nullspace occupancy as a mechanism for out of distribution failure. In *The Eleventh International Conference on Learning Representations*, 2022.

Huseyin A Inan, Osman Ramadan, Lukas Wutschitz, Daniel Jones, Victor Rühle, James Withers, and Robert Sim. Training data leakage analysis in language models. *arXiv preprint arXiv:2101.05405*, 2021.

Ziwei Ji and Matus Telgarsky. Directional convergence and alignment in deep learning. *Advances in Neural Information Processing Systems*, 2020.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12):1–38, 2023.

Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. *arXiv preprint arXiv:1912.02178*, 2019.

Yiding Jiang, Vaishnavh Nagarajan, Christina Baek, and J Zico Kolter. Assessing generalization of sgd via disagreement. *arXiv preprint arXiv:2106.13799*, 2021.

Liqiang Jing, Ruosen Li, Yunmo Chen, Mengzhao Jia, and Xinya Du. FAITHSCORE: Evaluating hallucinations in large vision-language models. *arXiv preprint arXiv:2311.01477*, 2023.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Adam Tauman Kalai and Santosh S Vempala. Calibrated language models must hallucinate. *arXiv preprint arXiv:2311.14648*, 2023.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. Large language models struggle to learn long-tail knowledge. In *International Conference on Machine Learning*, 2023.

Katie Kang, Amrith Setlur, Claire Tomlin, and Sergey Levine. Deep neural networks tend to extrapolate predictably. *arXiv preprint arXiv:2310.00873*, 2023.

Katie Kang, Amrith Setlur, Dibya Ghosh, Jacob Steinhardt, Claire Tomlin, Sergey Levine, and Aviral Kumar. What do learning dynamics reveal about generalization in llm reasoning? *arXiv preprint arXiv:2411.07681*, 2024a.

Katie Kang, Eric Wallace, Claire Tomlin, Aviral Kumar, and Sergey Levine. Unfamiliar finetuning examples control how language models hallucinate. *arXiv preprint arXiv:2403.05612*, 2024b.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, pages 5637–5664. PMLR, 2021.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *arxiv*, 2009.

Anders Krogh and John Hertz. A simple weight decay can improve generalization. *Advances in neural information processing systems*, 4, 1991.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. *arXiv preprint arXiv:2302.09664*, 2023.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

Nayeon Lee, Wei Ping, Peng Xu, Mostofa Patwary, Pascale N Fung, Mohammad Shoeybi, and Bryan Catanzaro. Factuality enhanced language models for open-ended text generation. *Advances in Neural Information Processing Systems*, 2022.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. *arXiv preprint arXiv:2306.03341*, 2023a.

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. From quantity to quality: Boosting llm performance with self-guided data selection for instruction tuning. *arXiv preprint arXiv:2308.12032*, 2023b.

Stephanie Lin, Jacob Hilton, and Owain Evans. Teaching models to express their uncertainty in words. *arXiv preprint arXiv:2205.14334*, 2022.

Kevin Liu, Stephen Casper, Dylan Hadfield-Menell, and Jacob Andreas. Cognitive dissonance: Why do language model outputs disagree with internal representations of truthfulness? *arXiv preprint arXiv:2312.03729*, 2023.

Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning, 2024. URL `https://arxiv.org/abs/2312.15685`.

Keming Lu, Hongyi Yuan, Zheng Yuan, Runji Lin, Junyang Lin, Chuanqi Tan, Chang Zhou, and Jingren Zhou. # instag: Instruction tagging for analyzing supervised fine-tuning of large language models. In *The Twelfth International Conference on Learning Representations*, 2023.

Kaifeng Lyu and Jian Li. Gradient descent maximizes the margin of homogeneous neural networks. In *International Conference on Learning Representations*, 2019.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *ACL*, 2023.

Potsawee Manakul, Adian Liusie, and Mark JF Gales. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. *arXiv preprint arXiv:2303.08896*, 2023.

Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*, 2018.

Dheeraj Mekala, Alex Nguyen, and Jingbo Shang. Smaller language models are capable of selecting instruction-tuning training data for larger language models. *arXiv preprint arXiv:2402.10430*, 2024.

Mohsen Mesgar, Edwin Simpson, and Iryna Gurevych. Improving factual consistency between a response and persona facts. *arXiv preprint arXiv:2005.00036*, 2020.

John P Miller, Rohan Taori, Aditi Raghunathan, Shiori Sagawa, Pang Wei Koh, Vaishaal Shankar, Percy Liang, Yair Carmon, and Ludwig Schmidt. Accuracy on the line: on the strong correlation between out-of-distribution and in-distribution generalization. In *International Conference on Machine Learning*, pages 7721–7735. PMLR, 2021.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. *arXiv preprint arXiv:2305.14251*, 2023.

Niels Mündler, Jingxuan He, Slobodan Jenko, and Martin Vechev. Self-contradictory hallucinations of large language models: Evaluation, detection and mitigation. *arXiv preprint arXiv:2305.15852*, 2023.

Vaishnavh Nagarajan and J Zico Kolter. Generalization in deep networks: The role of distance from initialization. *arXiv preprint arXiv:1901.01672*, 2019a.

Vaishnavh Nagarajan and J Zico Kolter. Uniform convergence may be unable to explain generalization in deep learning. *Advances in Neural Information Processing Systems*, 32, 2019b.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? *arXiv preprint arXiv:1810.09136*, 2018.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. *arXiv preprint arXiv:1412.6614*, 2014.

Behnam Neyshabur, Russ R Salakhutdinov, and Nati Srebro. Path-sgd: Path-normalized optimization in deep neural networks. *Advances in neural information processing systems*, 28, 2015a.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In *Conference on learning theory*, pages 1376–1401. PMLR, 2015b.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. *Advances in neural information processing systems*, 30, 2017.

Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436, 2015.

Chenri Ni, Nontawat Charoenphakdee, Junya Honda, and Masashi Sugiyama. On the calibration of multiclass classification with rejection. *Advances in Neural Information Processing Systems*, 32, 2019.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems*, 32, 2019.

Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

Tim Pearce, Alexandra Brintrup, and Jun Zhu. Understanding softmax confidence and uncertainty. *arXiv preprint arXiv:2106.04972*, 2021.

Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, et al. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *arXiv preprint arXiv:2302.12813*, 2023.

Emmanouil Antonios Platanios, Avinava Dubey, and Tom Mitchell. Estimating accuracy from unlabeled data: A bayesian approach. In *International Conference on Machine Learning*, pages 1416–1425. PMLR, 2016.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International conference on machine learning*, pages 5389–5400. PMLR, 2019.

Paul Roit, Johan Ferret, Lior Shani, Roee Aharoni, Geoffrey Cideron, Robert Dadashi, Matthieu Geist, Sertan Girgin, Léonard Hussenot, Orgad Keller, et al. Factually consistent summarization via reinforcement learning with textual entailment feedback. *arXiv preprint arXiv:2306.00186*, 2023.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Shibani Santurkar, Dimitris Tsipras, and Aleksander Madry. Breeds: Benchmarks for subpopulation shift. *arXiv preprint arXiv:2008.04859*, 2020.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Amrith Setlur, Saurabh Garg, Xinyang Geng, Naman Garg, Virginia Smith, and Aviral Kumar. Rl on incorrect synthetic data scales the efficiency of llm math reasoning by eight-fold. *arXiv preprint arXiv:2406.14532*, 2024.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, YK Li, Yu Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

John Shulman. Reinforcement learning from human feedback: Progress and challenges, 2023. URL `https://www.youtube.com/watch?v=hhiLw5Q_UFg`.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. Retrieval augmentation reduces hallucination in conversation. *arXiv preprint arXiv:2104.07567*, 2021.

Chenglei Si, Zhe Gan, Zhengyuan Yang, Shuohang Wang, Jianfeng Wang, Jordan Boyd-Graber, and Lijuan Wang. Prompting GPT-3 to be reliable. *arXiv preprint arXiv:2210.09150*, 2022.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *Journal of Machine Learning Research*, 19(70):1–57, 2018.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Marco Antonio Stranisci, Rossana Damiano, Enrico Mensa, Viviana Patti, Daniele Radicioni, and Tommaso Caselli. Wikibio: a semantic resource for the intersectional analysis of biographical events. *arXiv preprint arXiv:2306.09505*, 2023.

Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert Müller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8(5), 2007.

Yiyou Sun, Yifei Ming, Xiaojin Zhu, and Yixuan Li. Out-of-distribution detection with deep nearest neighbors. In *International Conference on Machine Learning*, pages 20827–20840. PMLR, 2022.

Zhiqing Sun, Sheng Shen, Shengcao Cao, Haotian Liu, Chunyuan Li, Yikang Shen, Chuang Gan, Liang-Yan Gui, Yu-Xiong Wang, Yiming Yang, et al. Aligning large multimodal models with factually augmented RLHF. *arXiv preprint arXiv:2309.14525*, 2023.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Jihoon Tack, Sangwoo Mo, Jongheon Jeong, and Jinwoo Shin. Csi: Novelty detection via contrastive learning on distributionally shifted instances. *Advances in neural information processing systems*, 33:11839–11852, 2020.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. Bert rediscovers the classical nlp pipeline. *arXiv preprint arXiv:1905.05950*, 2019.

Katherine Tian, Eric Mitchell, Huaxiu Yao, Christopher D Manning, and Chelsea Finn. Fine-tuning language models for factuality. *arXiv preprint arXiv:2311.08401*, 2023a.

Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. *arXiv preprint arXiv:2305.14975*, 2023b.

Nadav Timor, Gal Vardi, and Ohad Shamir. Implicit regularization towards rank minimization in relu networks. In *International Conference on Algorithmic Learning Theory*, pages 1429–1459. PMLR, 2023.

Kushal Tirumala, Aram Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. Memorization without overfitting: Analyzing the training dynamics of large language models. *Advances in Neural Information Processing Systems*, 35:38274–38290, 2022.

Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *CVPR 2011*, pages 1521–1528. IEEE, 2011.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Logesh Kumar Umapathi, Ankit Pal, and Malaikannan Sankarasubbu. Med-HALT: Medical domain hallucination test for large language models. *arXiv preprint arXiv:2307.15343*, 2023.

Vladimir N Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of complexity: festschrift for alexey chervonenkis*, pages 11–30. Springer, 2015.

Neeraj Varshney, Wenlin Yao, Hongming Zhang, Jianshu Chen, and Dong Yu. A stitch in time saves nine: Detecting and mitigating hallucinations of LLMs by validating low-confidence generation. *arXiv preprint arXiv:2307.03987*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. Learning robust global representations by penalizing local predictive power. In *Advances in Neural Information Processing Systems*, pages 10506–10518, 2019.

Taylor Webb, Zachary Dulberg, Steven Frankland, Alexander Petrov, Randall O'Reilly, and Jonathan Cohen. Learning representations that support extrapolation. In *International conference on machine learning*, pages 10136–10146. PMLR, 2020.

Yongtao Wu, Zhenyu Zhu, Fanghui Liu, Grigorios Chrysos, and Volkan Cevher. Extrapolation and spectral bias of neural nets with hadamard product: a polynomial net study. *Advances in neural information processing systems*, 35:26980–26993, 2022.

Guoxuan Xia and Christos-Savvas Bouganis. Augmenting softmax information for selective classification with out-of-distribution data. In *Proceedings of the Asian Conference on Computer Vision*, pages 1995–2012, 2022.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*, 2021.

Keyulu Xu, Mozhi Zhang, Jingling Li, Simon S Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. How neural networks extrapolate: From feedforward to graph neural networks. *arXiv preprint arXiv:2009.11848*, 2020.

Weijia Xu, Sweta Agrawal, Eleftheria Briakou, Marianna J Martindale, and Marine Carpuat. Understanding and detecting hallucinations in neural machine translation via model introspection. *TACL*, 2023.

Yuqing Yang, Ethan Chern, Xipeng Qiu, Graham Neubig, and Pengfei Liu. Alignment for honesty. *arXiv preprint arXiv:2312.07000*, 2023.

Sina J Semnani Violet Z Yao, Heidi C Zhang, and Monica S Lam. WikiChat: Combating hallucination of large language models by few-shot grounding on wikipedia. *arXiv preprint arXiv:2305.14292*, 2023.

Gilad Yehudai, Ethan Fetaya, Eli Meirom, Gal Chechik, and Haggai Maron. From local structures to size generalization in graph neural networks. In *International Conference on Machine Learning*, pages 11975–11986. PMLR, 2021.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64 (3):107–115, 2021.

Hanning Zhang, Shizhe Diao, Yong Lin, Yi R Fung, Qing Lian, Xingyao Wang, Yangyi Chen, Heng Ji, and Tong Zhang. R-tuning: Teaching large language models to refuse unknown questions. *arXiv preprint arXiv:2311.09677*, 2023a.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017a.

Yue Zhang, Leyang Cui, Wei Bi, and Shuming Shi. Alleviating hallucinations of large language models through induced hallucinations. *arXiv preprint arXiv:2312.15710*, 2023b.

Zhifei Zhang, Yang Song, and Hairong Qi. Age progression/regression by conditional adversarial autoencoder. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5810–5818, 2017b.

Yingxiu Zhao, Bowen Yu, Binyuan Hui, Haiyang Yu, Fei Huang, Yongbin Li, and Nevin L Zhang. A preliminary study of the intrinsic relationship between complexity and alignment. *arXiv preprint arXiv:2308.05696*, 2023.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, 2021.

Appendix A

Appendices for Deep Neural Network Extrapolation

A.1 Instances Where "Reversion to the OCS" Does Not Hold

In this section, we will discuss some instances where "reversion to the OCS" does not hold. The first example is evaluating an MNIST classifier (trained via cross entropy) on an adversarially generated dataset using the Fast Gradient Sign Method (FSGM) [Goodfellow et al., 2014]. In Fig. A.1, we show our findings. On the left plot, we can see that the outputs corresponding to the adversarial dataset were farther from the OCS than the predictions corresponding to the training distribution, even though the adversarial dataset is more OOD. On the right plot, we show the normalized representation magnitude of the original and adversarial distributions throughout different layers of the network. We can see that the representations corresponding to adversarial inputs have larger magnitudes compared to those corresponding to the original inputs. This is a departure from the leftmost plots in Fig. 2.4, where the norm of the representations in later layers decreased as the inputs became more OOD. One potential reason for this behavior is that the adversarial optimization pushes the adversarial inputs to yield representations which align closer with the network weights, leading to higher magnitude representations which push outputs father from the OCS.

Our second example is Gaussian NLL models trained on UTKFace, and evaluated on inputs with impulse noise. Previously, in Fig. 2.3, we had shown that "reversion to the OCS" holds for UTKFace with gaussian blur, but we found this to not necessarily be the case for all corruptions. In Fig. A.2, we show the behavior of the models evaluated on inputs with increasing amounts of impulse noise. In the middle plot, we can see that as the OOD score increases (greater noising), the distance to the OCS increases, contradicting "reversion to the OCS". In the right plot, we show the magnitude of the representations in an internal layer for inputs with both gaussian blur and impulse noise. We can see that while

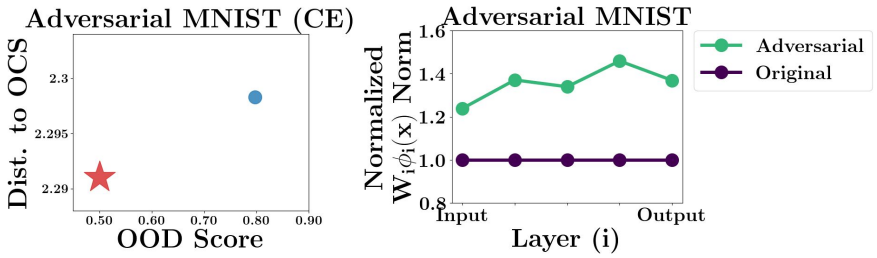

Figure A.1: On the left, we evaluate the distance between network predictions and the OCS as the input distribution becomes more OOD for an MNIST classifier. The red star represents the (holdout) training distribution, and the blue circle represents an adversarially generated evaluation distribution. Even though the adversarial distribution is more OOD, its predictions were farther from the OCS. On the right, we plot the normalized representation magnitude across different layers of the network. The representations corresponding to adversarial inputs had greater magnitude throughout all the layers.

the representation norm decreases with greater noise for gaussian blur, the representation norm actually increases for impulse noise. We are not sure why the model follows "reversion to the OCS" for gaussian blur but not impulse noise. We hypothesize that one potential reason could be that, because the model was trained to predict age, the model learned to identify uneven texture as a proxy for wrinkles. Indeed, we found that the model predicted higher ages for inputs with greater levels of impulse noise, which caused the predictions to move farther from the OCS.

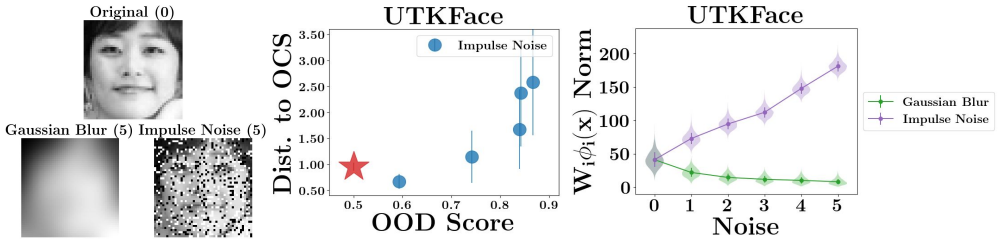

Figure A.2: On the left, we visualize an example of UTKFace inputs in its original form, with gaussian blur, and with impulse noise. In the middle, we evaluate the distance between network predictions and the OCS as the input distribution becomes more OOD. Each point represents a different evaluation dataset, with the red star representing the (holdout) training distribution, and circles representing OOD datasets with increasing levels of impulse noise. The vertical line associated with each point represents the standard deviation over 5 training runs. As the OOD score of the evaluation dataset increases, the model predictions here tended to move farther from the OCS. On the right, we plot the magnitude of the representation in a specific layer of the model corresponding to inputs with different levels of gaussian blur and impulse noise. The representation magnitude for inputs with greater gaussian blur tend to decrease, while the representation magnitude with greater impulse noise tend to increase.

A.2 Experiment Details

Datasets

The datasets with discrete labels include CIFAR10 [Krizhevsky et al., 2009], ImageNet [Deng et al., 2009] (subsampled to 200 classes to match ImageNet-R(rendition) [Hendrycks et al., 2021a]), DomainBed OfficeHome [Gulrajani and Lopez-Paz, 2020], BREEDS LIVING-17 and NON-LIVING-26 [Santurkar et al., 2020], and Wilds Amazon [Koh et al., 2021], and the datasets with continuous labels include SkinLesionPixels [Gustafsson et al., 2023] and UTKFace [Zhang et al., 2017b]. We evaluate CIFAR10 models on CIFAR10-C [Hendrycks and Dietterich, 2019], which includes synthetic corruptions at varying levels of intensity. We evaluate ImageNet models on ImageNet-R and ImageNet-Sketch [Wang et al., 2019], which include renditions of ImageNet classes in novel styles and in sketch form. OfficeHome includes images of furniture in the style of product, photo, clipart, and art. We train on product images and evaluate on the other styles. BREEDS datasets consist of training and OOD test datasets consisting of the same classes but distinct subclasses. Wilds Amazon consists of training and OOD test datasets of Amazon reviews from different sets of users. To amplify the distribution shift, we subsample the OOD dataset to only include incorrectly classified points. SkinLesionPixels consists of dermatoscopic images, where the training and OOD test datasets were collected from patients from different countries. UTKFace consists

of images of faces, and we construct OOD test datasets by adding different levels of Gaussian blur to the input.

Training Parameters

First, we will discuss the parameters we used to train our models.

Task	Network Architecture
MNIST	2 convolution layers followed by 2 fully connected layers ReLU nonlinearities
CIFAR10	ResNet20
ImageNet	ResNet50
OfficeHome	ResNet50
BREEDS	ResNet18
Amazon	DistilBERT
SkinLesionPixels	ResNet34
UTKFace	Custom VGG style architecture

Task	Optimizer	Learning Rate	Learning Rate Scheduler	Weight Decay	Momentum
MNIST	Adam	0.001	Step; $\gamma = 0.7$	0.01	-
CIFAR10	SGD	0.1	Multi step milestones=[100, 150]	0.0001	0.9
ImageNet	SGD	0.1	Step; $\gamma = 0.1$ step size = 30	0.0001	0.9
OfficeHome	Adam	0.00005	-	0	-
BREEDS	SGD	0.2	Linear with warm up (warm up frac=0.05)	0.00005	0.9
Amazon	AdamW	0.00001	Linear with warm up (warm up frac=0)	0.01	-
SkinLesionPixels	Adam	0.001	-	0	-
UTKFace	Adam	0.001	-	0	-

Task	Data Preprocessing
MNIST	Normalization
CIFAR10	Random horizontal flip, random crop, normalization
ImageNet	Random resized crop, random horizontal flip, normalization
OfficeHome	Random resized crop, random horizontal flip, color jitter, random grayscale, normalization
BREEDS	Random horizontal flip, random resized crop, randaugment
Amazon	DistilBERT Tokenizer
SkinLesionPixels	Normalization
UTKFace	Normalization

Evaluation Metrics

Next, we will describe the details of our OOD score calculation. For image datasets, we pass the image inputs through a pretrained ResNet18 ImageNet featurizer to get feature representations, and train a linear classifier to classify whether the feature representations are from the training distribution or the evaluation distribution. We balance the training data of the classifier such that each distribution makes up 50 percent. We then evaluate the linear classifier on the evaluation distribution, and calculate the average predicted likelihood that the batch of inputs are sampled from the evaluation distribution, which we use as the OOD score. For text datasets, we use a similar approach, but use a DistilBERT classifier and Tokenizer instead of the linear classifier and ImageNet featurizer.

There are some limitations to the OOD score. Ideally, we would like the OOD score to be a measure of how well model trained on the training distribution will generalize to the evaluation distribution. This is often the case, such as the datasets in our experiments, but not always. Consider a scenario where the evaluation dataset is a subset of the training dataset with a particular type of feature. Here, a neural network model trained on the training dataset will likely generalize well to the evaluation dataset in terms of task performance. However, the evaluation dataset will likely receive a high OOD score, because the evaluation inputs will be distinguishable from the training inputs, since the evaluation dataset has a high concentration of a particular type of feature. In this case, the OOD score is not a good measure of the distribution shift of the evaluation dataset.

Additionally, with regards to our measure of distance between model predictions and the OCS, we note that this measure is only informative if the different datasets being evaluated have around the same distribution of labels. This is because both the MSE and the KL metrics are being averaged over the evaluation dataset.

Characterizing the OCS for Common Loss Functions

In this section, we will precisely characterize the OCS for a few of the most common loss functions: cross entropy, mean squared error (MSE), and Gaussian negative log likelihoog (Gaussian NLL).

Cross entropy. With a cross entropy loss, the neural network outputs a vector where each entry is associated with a class, which we denote as $f_\theta(x)_i$. This vector parameterizes a categorical distribution: $P_\theta(y_i|x) = \frac{e^{f_\theta(x)_i}}{\sum_{1\leq j\leq m} e^{f_\theta(x)_j}}$. The loss function minimizes the divergence between $P_\theta(y|x)$ and $P(y|x)$, given by

$$\mathcal{L}(f_\theta(x), y) = \sum_{i=1}^{m} \mathbb{1}[y = y_i] \log\Big(\frac{e^{f_\theta(x)_i}}{\sum_{j=1}^{m} e^{f_\theta(x)_j}}\Big).$$

While there can exist multiple optimal constant solutions for the cross entropy loss, they all map to the same distribution which matches the marginal empirical distribution of the training labels, $P_{f^*_{\text{constant}}}(y_i) = \frac{e^{f^*_{\text{constant},i}}}{\sum_{1\leq j\leq m} e^{f^*_{\text{constant},j}}} = \frac{1}{N}\sum_{1\leq i\leq N} \mathbb{1}[y = y_i]$.

For the cross entropy loss, the uncertainty of the neural network prediction can be captured by the entropy of the output distribution. Because $P_{f^*_{\text{constant}}}(y)$ usually has much higher entropy than $P_\theta(y|x)$ evaluated on the training distribution, $P_\theta(y|x)$ on OOD inputs will tend to have higher entropy than on in-distribution inputs.

Gaussian NLL. With this loss function, the output of the neural network parameterizes the mean and standard deviation of a Gaussian distribution, which we denote as $f_\theta(x) = [\mu_\theta(x), \sigma_\theta(x)]$. The objective is to minimize the negative log likelihood of the training labels under the predicted distribution, $P_\theta(y|x) \sim \mathcal{N}(\mu_\theta(x), \sigma_\theta(x))$:

$$\mathcal{L}(f_\theta(x), y) = \log(\sigma_\theta(x)^2) + \frac{(y - \mu_\theta(x))^2}{\sigma_\theta(x)^2}.$$

Let us similarly denote $f^*_{\text{constant}} = [\mu^*_{\text{constant}}, \sigma^*_{\text{constant}}]$. In this case, $\mu^*_{\text{constant}} = \frac{1}{N}\sum_{1\leq i\leq N} y_i$, and $\sigma^*_{\text{constant}} = \frac{1}{N}\sum_{1\leq i\leq N}(y_i - \mu^*_{\text{constant}})^2$. Here, $\sigma^*_{\text{constant}}$ is usually much higher than the standard deviation of $P(y|x)$ for any given x. Thus, our observation suggests that neural networks should predict higher standard deviations for OOD inputs than training inputs.

MSE. Mean squared error can be seen as a special case of Gaussian NLL in which the network only predicts the mean of the Gaussian distribution, while the standard deviation is held constant, i.e. $P_\theta(y|x) \sim \mathcal{N}(f_\theta(x), 1)$. The specific loss function is given by:

$$\mathcal{L}(f_\theta(x), y) = (y - f_\theta(x))^2.$$

Here, $f^*_{\text{constant}} = \frac{1}{N}\sum_{1\leq i\leq N} y_i$. Unlike the previous two examples, in which OOD predictions exhibited greater uncertainty, predictions from an MSE loss do not capture any explicit notions of uncertainty. However, our observation suggests that the model's predicted mean will still move closer to the average label value as the test-time inputs become more OOD.

A.3 Empirical Analysis

Additional Experiments

In this section, we will provide additional experimental analysis to support the hypothesis that we put forth in Section 2.4. First, in order to understand whether the trends that we observe for CIFAR10 and MNIST would scale to larger models and datasets, we perform the same analysis as the ones presented in Figure 2.4 on a ResNet50 model trained on ImageNet, and evaluated on ImageNet-Sketch and ImageNet-R(endition). Our findings are presented in

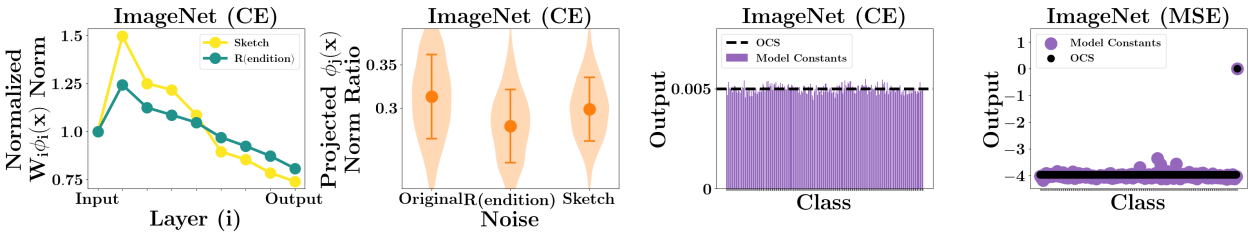

Figure A.3: Analysis of the interaction between representations and weights for as distribution shift increases, for a model trained on ImageNet and evaluated on ImageNet-Sketch and ImageNet-R(enditions).

Figure A.3. Here, we can see that the same trends from the CIFAR10 and MNIST analysis seem to transfer to the ImageNet models.

Next, we aim to better understand the effects of normalization layers in the network on the behavior of the model representations. We trained models with no normalization (NN), batch normalization (BN), and layer normalization (LN) on the MNIST and CIFAR10 datasets, and evaluated them on OOD test sets with increasing levels of rotation and noise. Note that the model architecture and all other training details are held fixed across these models (for each datasets) with the exception of the type of normalization layer used (or lack thereof). We perform the same analysis as the ones presented in Figure 2.4, and present our findings in Figure A.4 and A.5. We found similar trends across the different models which are consistent with the ones we presented in Figure 2.4

Analysis Details

In this section, we will provide details on the specific neural network layers that we used in our analysis in Sections 2.4 and A.3. We will illustrate diagrams for the neural network architectures that we used for each of our experiments, along with labels of the layers associated with each quantity we measure. In the first column of each analysis figure, we measure quantities at different layers of of the network; we denote these by $i_0, ..., i_n$, where each i represents one tick in the X-axis from left to right. We use j to denote the layer used in the plots in second column of each figure. We k_{CE} (and k_{MSE})to denote the layer used in the plots in the third (and fourth) columns of each figure, respectively. We illustrate the networks used in Figure 2.4 in Figure A.6, the one used in Figure A.3 in Figure A.7, the ones used in Figure A.4 in Figure A.8, and the ones used in Figure A.5 in Figure A.9.

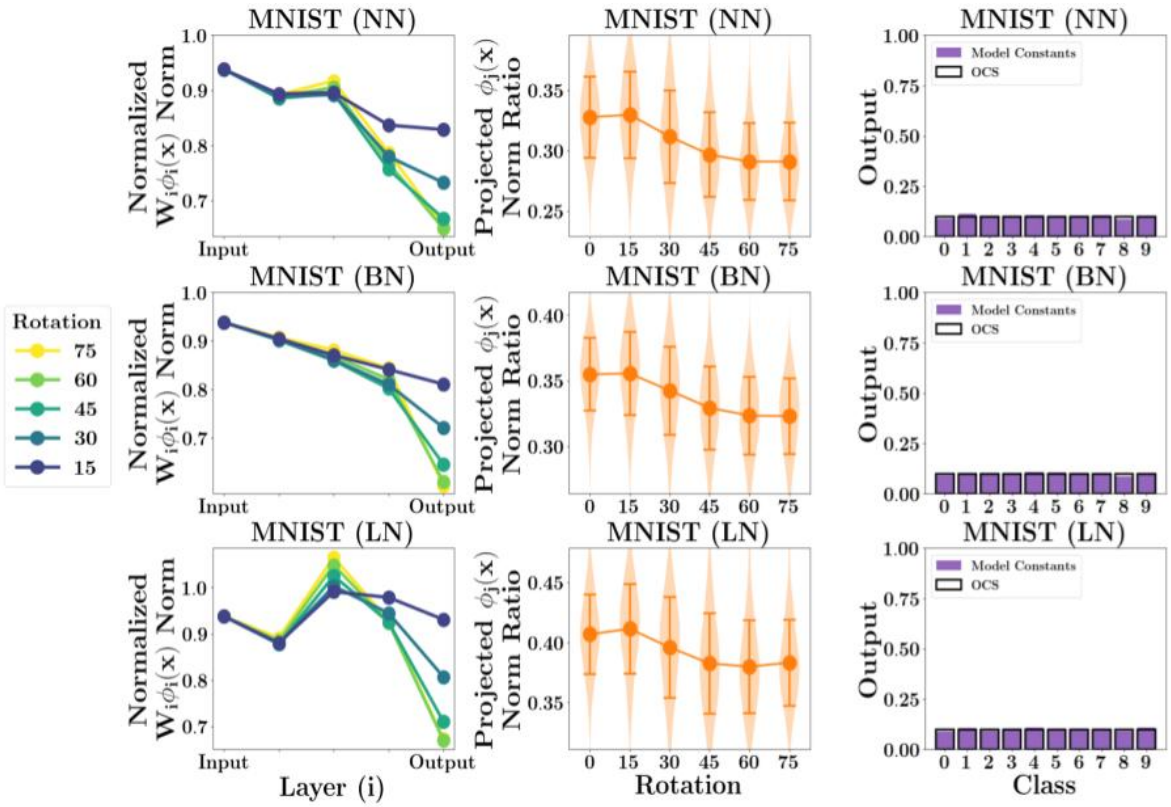

Figure A.4: Analysis of the interaction between representations and weights as distribution shift increases, for a model trained on MNIST and evaluated on increasing levels of rotation. The models being considered includes a four layer neural network with no normalization (top), batch normalization (middle), and layer normalization (bottom).

A.4 Proofs from Section 2.4

In the first subsection, we describe the setup, and gradient flow with some results from prior works that we rely upon in proving our claims on the in-distribution and out-of-distribution activation magnitudes. In the following subsections we prove our main claims from Section 2.4.

Setup. We are learning over a class of homogeneous neural networks $\mathcal{F} := \{f(W; x) : w \in \mathcal{W}\}$, with L layers, and element wise activation $\sigma(x) = x\mathbb{1}(x \geq 0)$ (ReLU function), taking the functional form:

$$f(W; x) = W_L\sigma(W_{L-1} \ldots \sigma(W_2\sigma(W_1x)) \ldots),$$

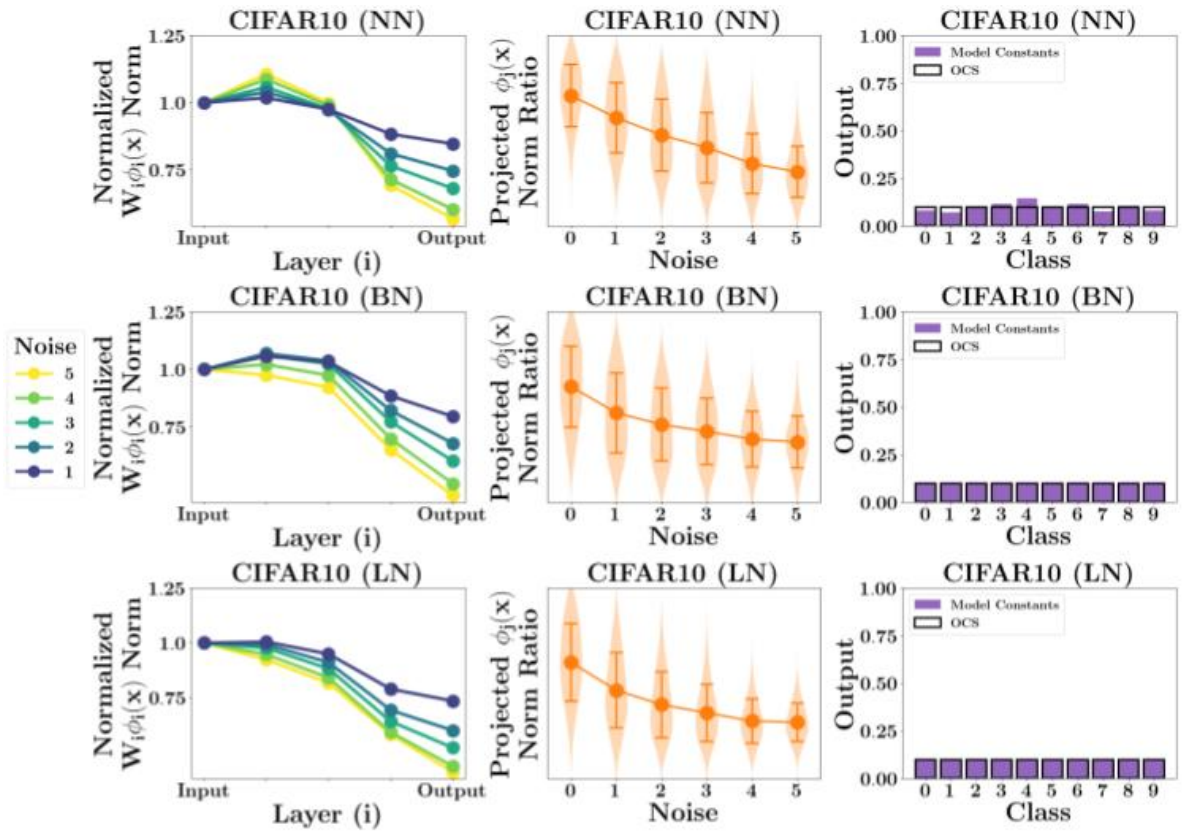

Figure A.5: Analysis of the interaction between representations and weights as distribution shift increases, for a model trained on CIFAR10 and evaluated on increasing levels of noise. The models being considered includes AlexNet with no normalization (top), batch normalization (middle), and layer normalization (bottom).

where $W_i \in \mathbb{R}^{m\times m}, \forall i \in \{2,\ldots,L-1\}$, $W_1 \in \mathbb{R}^{m\times 1}$ and output dimension is set to 1, i.e., $W_L \in \mathbb{R}^{1\times m}$. We say that class $\mathcal{F}$ is homogeneous, if there exists a constant C such that, for all $w \in \mathcal{W}$, we have:

$$f(\alpha \cdot W; x) = \alpha^C \cdot f(W; x).$$

Our focus is on a binary classification problem where we have a joint distribution over inputs and labels: $\mathcal{X} \times \mathcal{Y}$. Here, the inputs are from set $\mathcal{X} := \{x \in \mathbb{R}^d : \|x\|_2 \leq B\}$, and labels are binary $\mathcal{Y} := \{-1, +1\}$. We have an IID sampled training dataset $\mathcal{D} := \{(x_i, y_i)\}_{i=1}^n$ containing pairs of data points x_i and their corresponding labels y_i.

For a loss function $\ell : \mathbb{R} \to \mathbb{R}$, the empirical loss of $f(W; x)$ on the dataset $\mathcal{D}$ is

$$L(w; \mathcal{D}) := \sum_{i=1}^{n} \ell\left(y_i f(W; x_i)\right). \tag{A.1}$$

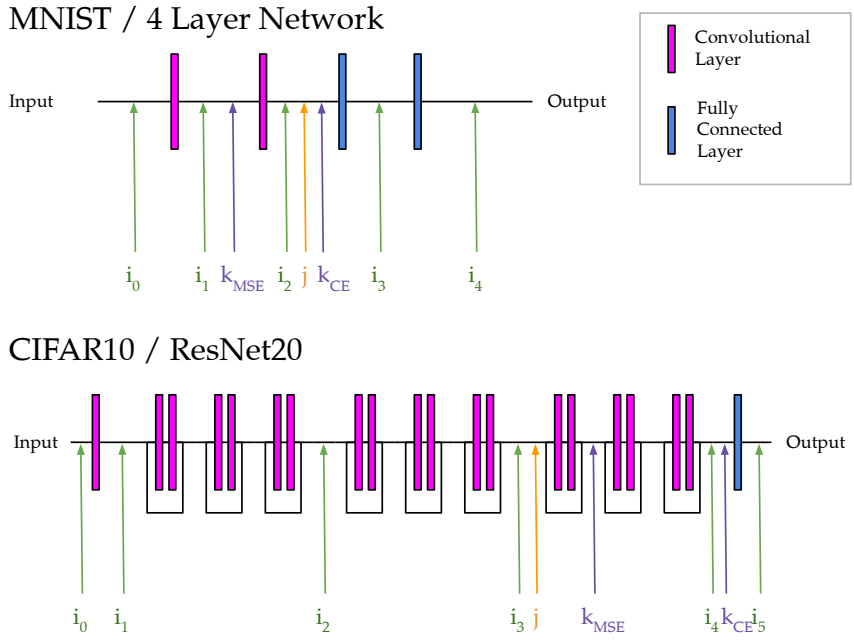

Figure A.6: Diagram of neural network models used in our experimental analysis in Figure 2.4, along with labels of the specific layers we used in our analysis.

Here, the loss ℓ can be the exponential loss $\ell(q) = e^{-q}$ and the logistic loss $\ell(q) = \log(1+e^{-q})$. To refer to the output (post activations if applicable) of layer j in the network $f(W;\cdot)$, for the input x, we use the notation: $f_j(W;x)$.

Gradient flow (GF). We optimize the objective in Equation equation A.1 using gradient flow. Gradient flow captures the behavior of gradient descent with an infinitesimally small step size [Arora et al., 2019, Huh et al., 2021, Galanti et al., 2022, Timor et al., 2023]. Let $W(t)$ be the trajectory of gradient flow, where we start from some initial point $W(0)$ of the weights, and the dynamics of $W(t)$ is given by the differential equation:

$$\frac{dW(t)}{dt} = -\nabla_{W=W(t)} L(w;\mathcal{D}). \tag{A.2}$$

Note that the ReLU function is not differentiable at 0. Practical implementations of gradient methods define the derivative $\sigma'(0)$ to be some constant in $[0,1]$. Following prior works [Timor et al., 2023], in this work we assume for convenience that $\sigma'(0) = 0$. We say that gradient flow converges if the followling limit exists:

$$\lim_{t\to\infty} W(t).$$

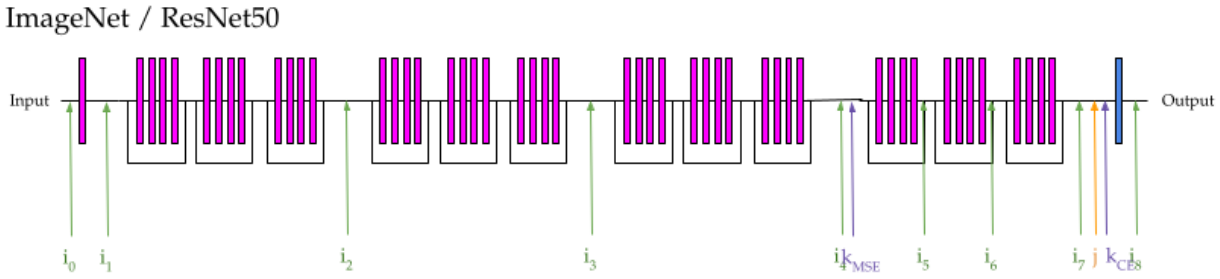

Figure A.7: Diagram of neural network models used in our experimental analysis in Figure A.3, along with labels of the specific layers we used in our analysis.

In this case, we denote $W(\infty) := \lim_{t\to\infty} W(t)$. We say that the gradient flow converges in direction if the following limit exists:

$$\lim_{t\to\infty} W(t)/\|W(t)\|_2.$$

Whenever the limit point $\lim_{t\to\infty} W(t)$ exists, we refer to the limit point as the ERM solution by running gradient flow, and denote it as $\hat{W}$.

Gradient flow convergence for interpolating homogeneous networks. Now, we use a result from prior works that states the implicit bias of gradient flow towards max-margin solutions when sufficiently deep and wide homogeneous networks are trained with small learning rates and exponential tail classification losses. As the loss converges to zero, the solution approaches a KKT point of an optimization problem that finds the minimum l_2 norm neural network with a margin of at least 1 on each point in the training set. This is formally presented in the following Lemma adapted from [Ji and Telgarsky, 2020] and [Lyu and Li, 2019].

Lemma A.4.1 (Gradient flow is implicitly biased towards minimum $\|\cdot\|_2$) *Consider minimizing the average of either the exponential or the logistic loss (in equation A.1) over a binary classification dataset $\mathcal{D}$ using gradient flow (in equation A.2) over the class of homogeneous neural networks $\mathcal{F}$ with ReLU activations. If the average loss on $\mathcal{D}$ converges to zero as $t \to \infty$, then gradient flow converges in direction to a first order stationary point (KKT point) of the following maximum margin problem in the parameter space of $\mathcal{W}$:*

$$\min_{f(W;x)\in\mathcal{F}} \frac{1}{2}\sum_{j\in[L]} \|W_i\|_2^2 \quad s.t. \quad \forall i \in [n] \;\; y_i f(W; x_i) \geq 1. \tag{A.3}$$

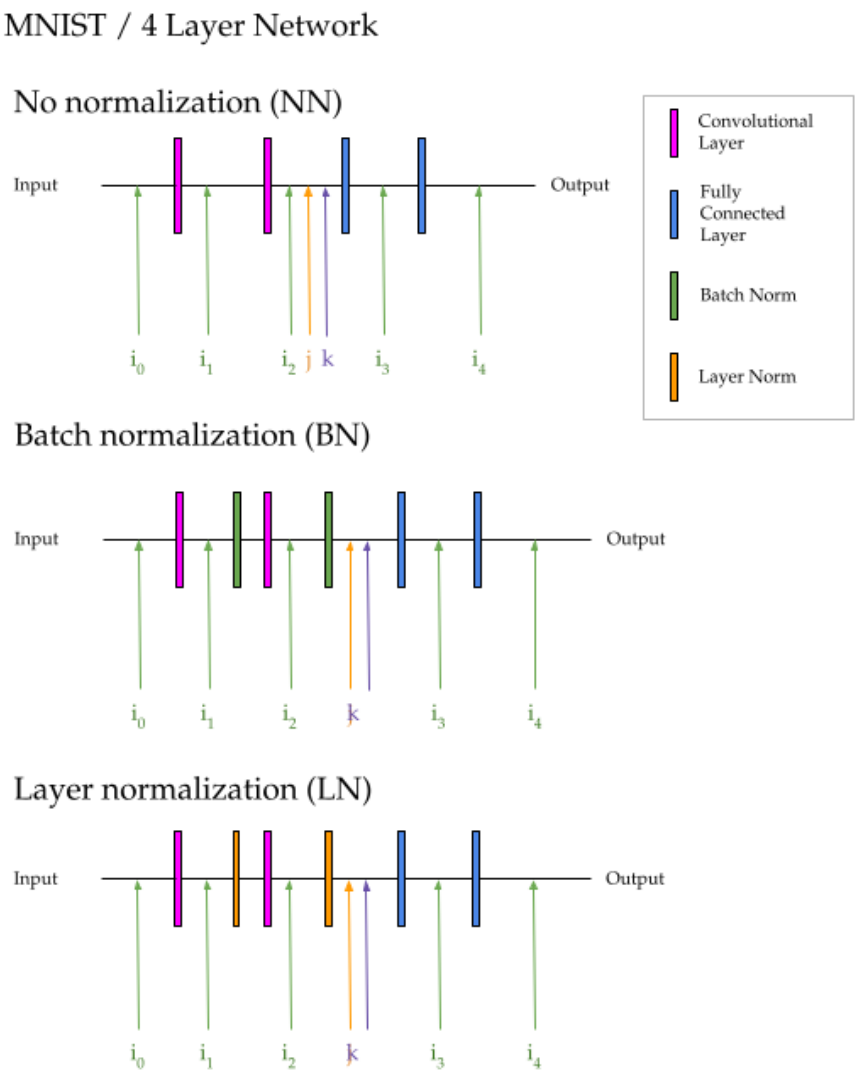

Figure A.8: Diagram of neural network models used in our experimental analysis in Figure A.4, along with labels of the specific layers we used in our analysis.

Spectrally normalized margin based generalization bounds [Bartlett et al., 2017]. Prior work on Rademacher complexity based generalization bounds provides excess risk bounds based on spectrally normalized margins, which scale with the Lipschitz constant (product of spectral norms of weight matrices) divided by the margin.

Lemma A.4.2 (Adaptation of Theorem 1.1 from [Bartlett et al., 2017]) *For the class*

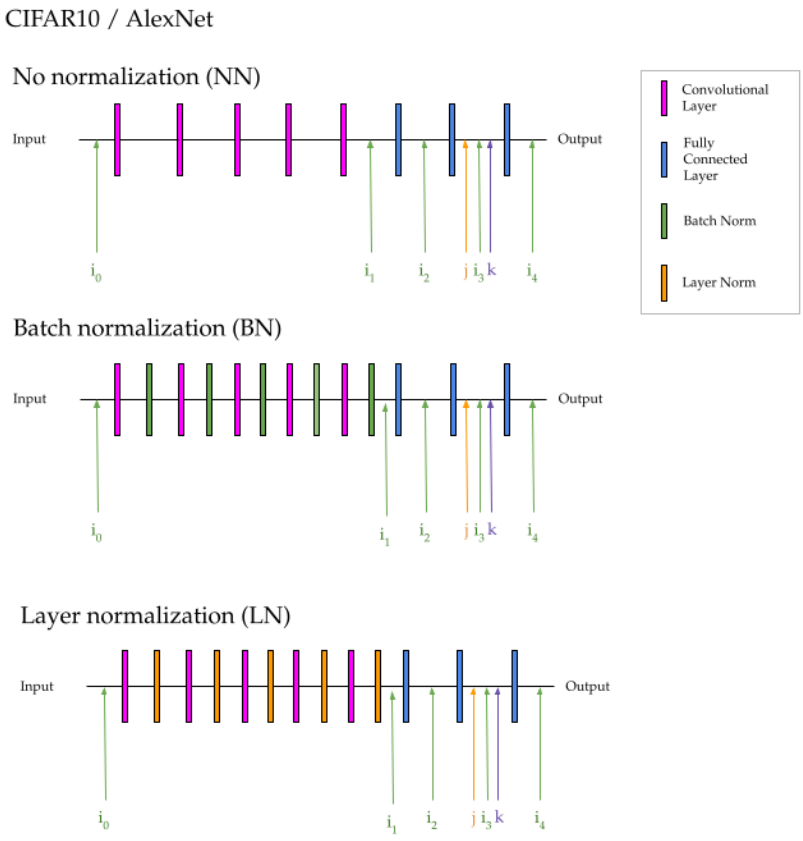

Figure A.9: Diagram of neural network models used in our experimental analysis in Figure A.5, along with labels of the specific layers we used in our analysis.

homogeneous of ReLU networks in $\mathcal{F}$ *with reference matrices* $(A_1, \ldots, A_L)$*, and all distributions* P *inducing binary classification problems, i.e., distributions over* $\mathbb{R}^d \times \{-1, +1\}$*, with probability* $1-\delta$ *over the IID sampled dataset* $\mathcal{D}$*, and margin* $\gamma > 0$*, the network* $f(w; x)$ *has expected margin loss upper bounded as:*

$$\mathbb{E}_{(x,y)\sim P} \mathbb{1}(yf(w;x) \geq \gamma) \;\leq\; \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}(y_i f(w;x_i) \geq \gamma) + \tilde{\mathcal{O}}\left(\frac{\mathcal{R}_{W,A}}{\gamma n}\log(m) + \sqrt{\frac{\log(1/\delta)}{n}}\right),$$

where the covering number bound determines $\mathcal{R}_{W,A} := \left(\prod_{i=1}^{L} \|W_i\|_{\text{op}}\right)\left(\sum_{i=1}^{L} \frac{\|W_i^\top - A_i^\top\|_{2,1}}{\|W_i\|_{\text{op}}^{2/3}}\right)$.

Lower bound for activation magnitude on in-distribution data

Proposition A.4.1 (P_{train} **observes high norm features)** *When* $f(\hat{W}; x)$ *fits* $\mathcal{D}$*, i.e.,* $y_i f(\hat{W}; x_i) \geq \gamma$*,* $\forall i \in [N]$*, then w.h.p* $1-\delta$ *over* D*, layer* j *representations* $f_j(\hat{W}; x)$ *satisfy*

$\mathbb{E}_{P_{\text{train}}}[\|f_j(\hat{W};x)\|_2] \geq (1/C_0)(\gamma - \tilde{\mathcal{O}}(\sqrt{\log(1/\delta)/N} + C_1^2 \log m/N\gamma))$, *if* $\exists$ *constants* C_0, C_1 *s.t.* $\|\hat{W}_j\|_2 \leq C_0^{j+1-L}$, $C_1 \geq C_0^L$.

Proof.

Here, we lower bound the expected magnitude of the in-distribution activation norms at a fixed layer j in terms of the expected activation norm at the last layer, by repeatedly applying Cauchy-Schwartz inequality and the property of ReLU activations: $\|\sigma(x)\|_2 \leq \|x\|_2$, alternatively.

$$\begin{aligned}
&\mathbb{E}_{P_{\text{train}}}|f(\hat{W};x)| \\
&= \quad \mathbb{E}_{P_{\text{train}}}\left[\|\hat{W}_L\sigma(\hat{W}_{L-1}\ldots\sigma(\hat{W}_2\sigma(\hat{W}_1x))\ldots)\|_2\right] \\
&\leq \quad \|\hat{W}_L\|_{\text{op}}\mathbb{E}_{P_{\text{train}}}\left[\|\sigma(\hat{W}_{L-1}\ldots\sigma(\hat{W}_2\sigma(\hat{W}_1x))\ldots)\|_2\right] \quad \text{(Cauchy-Schwartz)} \\
&\leq \quad \|\hat{W}_L\|_{\text{op}}\mathbb{E}_{P_{\text{train}}}\left[\|\hat{W}_{L-1}\ldots\sigma(\hat{W}_2\sigma(\hat{W}_1x))\|_2\right] \quad \text{(ReLU activation property)}
\end{aligned}$$

Doing the above repeatedly gives us the following bound:

$$\begin{aligned}
\mathbb{E}_{P_{\text{train}}}|f(\hat{w};x)| &\leq \left(\prod_{k=j+1}^{L}\|\hat{W}_k\|_{\text{op}}\right)\cdot\mathbb{E}_{P_{\text{train}}}\left[\|f_j(\hat{W};x)\|_2\right] \\
&\leq C_0^{L-(j+1)/L}\left[\|f_j(\hat{W};x)\|_2\right] \leq C_0\left[\|f_j(\hat{W};x)\|_2\right]
\end{aligned}$$

Next, we use a generalization bound on the margin loss to further lower bound the expected norm of the last layer activations. Recall, that we use gradient flow to converge to globally optimal solution of the objective in equation A.1 such that the training loss converges to 0 as $t \to \infty$. Now, we use the spectrally normalized generalization bound from Lemma A.4.2 to get:

$$\mathbb{E}_{(x,y)\sim P}\mathbb{1}(yf(\hat{W};x) \geq \gamma) \lesssim \tilde{\mathcal{O}}\left(\frac{\log m}{\gamma N}\left(\prod_{i=1}^{L}\|\hat{W}_i\|_{\text{op}}\right)\left(\sum_{i=1}^{L}\frac{\|\hat{W}_i^\top - A_i^\top\|_{2,1}}{\|\hat{W}_i\|_{\text{op}}^{2/3}}\right) + \sqrt{\frac{\log(1/\delta)}{N}}\right)$$

This implies that with probability $1-\delta$ over the training set $\mathcal{D}$, on at least $\mathcal{O}(\sqrt{\log(1/\delta)/n})$ fraction of the test set, the margin is at least γ, i.e., if we characterize the set of correctly classified test points as $\mathcal{C}_{\hat{W}}$, then:

$$\begin{aligned}
\mathbb{E}[|f(\hat{W};x)| \mid (x,y) \in \mathcal{C}_{\hat{W}}] &= \mathbb{E}[|y\cdot f(\hat{W};x)| \mid (x,y) \in \mathcal{C}_{\hat{W}}] \\
&\geq \mathbb{E}[y\cdot f(\hat{W};x) \mid (x,y) \in \mathcal{C}_{\hat{W}}] \geq \gamma
\end{aligned}$$

We are left with lower bounding: $\mathbb{E}[|f(\hat{W};x)| \mid (x,y) \notin \mathcal{C}_{\hat{W}}]$ which is trivially ≥ 0. Now, from the generalization guarantee we know:

$$\begin{aligned}\mathbb{E}(\mathbb{1}((x,y) \in C_{\hat{W}})) &\lesssim \tilde{\mathcal{O}}\left(\frac{\log m}{\gamma N}\left(\prod_{i=1}^{L} C_0^{1/L}\right)\left(\sum_{i=1}^{L}\frac{\|\hat{W}_i^\top - A_i^\top\|_{2,1}}{\|\hat{W}_i\|_{\text{op}}^{2/3}}\right) + \sqrt{\frac{\log(1/\delta)}{N}}\right) \\ &\lesssim \tilde{\mathcal{O}}\left(\frac{\log m}{\gamma N} C_0\left(\sum_{i=1}^{L}\frac{\|\hat{W}_i^\top - A_i^\top\|_{2,1}}{\|\hat{W}_i\|_{\text{op}}^{2/3}}\right) + \sqrt{\frac{\log(1/\delta)}{N}}\right) \\ &\lesssim \tilde{\mathcal{O}}\left(\frac{\log m}{\gamma N} C_1\left(\sum_{i=1}^{L}\frac{\|\hat{W}_i^\top - A_i^\top\|_{2,1}}{\|\hat{W}_i\|_{\text{op}}^{2/3}}\right) + \sqrt{\frac{\log(1/\delta)}{N}}\right),\end{aligned}$$

where the final inequality uses

$$\frac{1}{L}\sum_i \|W_i\|_2^{2/3} \geq \left(\prod_{i=1}^{L}\|W_i\|_2^{2/3}\right)^{1/L},$$

which is the typical AM-GM inequality. We also use the inequality: $C_1 \geq C_0^{3L/2}$. This bound tells us that:

$$\mathbb{E}(\mathbb{1}((x,y) \in C_{\hat{W}})) \geq 1 - \tilde{\mathcal{O}}\left(\frac{\log m}{\gamma N} C_1 + \sqrt{\frac{\log(1/\delta)}{N}}\right).$$

Plugging the above into the lower bound we derived completes the proof of Proposition A.4.1.

Upper bound for activation magnitude on out-of-distribution data

Theorem A.4.3 (Feature norms can drop easily on P_{OOD}) *If $\exists$ a shallow network $f'(W;x)$ with L' layers and m' neurons satisfying conditions in Proposition 2.4.1 (γ=1), then optimizing the training objective with gradient flow over a class of deeper and wider homogeneous network $\mathcal{F}$ with $L > L', m > m'$ would converge directionally to a solution $f(\hat{W};x)$, for which the following is true: $\exists$ a set of rank 1 projection matrices $\{A_i\}_{i=1}^{L}$, such that if representations for any layer j satisfy $\mathbb{E}_{P_{\text{OOD}}}\|A_j f_j(\hat{W};x)\|_2 \leq \epsilon$, then $\exists C_2$ for which $\mathbb{E}_{P_{\text{OOD}}}[|f(\hat{W};x)|] \lesssim C_0(\epsilon + C_2^{-1/L}\sqrt{L+1/L})$.*

Proof.

Here, we show that there will almost rank one subspaces for each layer of the neural network, such that if the OOD representations deviate even slightly from the low rank subspace at any given layer, the last layer magnitude will collapse. This phenomenon is exacerbated in deep and wide networks, since gradient descent on deep homogeneous networks is biased towards KKT points of a minimum norm, max-margin solution [Lyu and Li, 2019], which consequently leads gradient flow on sufficiently deep and wide networks towards weight matrices that are low rank (as low as rank 1).

We can then show by construction that there will always exist these low rank subspaces which the OOD representations must not deviate from for the last layer magnitudes to not drop. Before we prove the main result, we adapt some results from [Timor et al., 2023] to show that gradient flow is biased towards low rank solutions in our setting. This is formally presented in Lemma A.4.4.

Lemma A.4.4 (GF on deep and wide nets is learns low rank $W_1, \ldots, W_L$) *We are given the IID sampled dataset for the binary classification task defined by distribution P_{train}, i.e., $\mathcal{D} := \{(x_i, y_i)\}_{i=1}^n \subseteq \mathbb{R}^d \times \{-1, 1\}$. Here, $\|x_i\|_2 \leq 1$, with probability 1. Let there exist a homogeneous network in class $\mathcal{F}'$, of depth $L' \geq 2$, and width $m' \geq 2$, if there exists a neural network $f(W'; x)$, such that: $\forall (x_i, y_i) \in \mathcal{D},\ f(W'; x_i) \cdot y_i \ \geq \ \gamma$, and the weight matrices $W_1', \ldots, W_{L'}'$ satisfy $\|W_i^{'}\|_F \leq C$, for some fixed constant $C > 0$. If the solution $f(W^\star, x)$ of gradient flow any class $\mathcal{F}$ of deeper $L > L'$ and wider $m > m'$ networks $f(W; x)$ converges to the global optimal of the optimization problem:*

$$\min_{f(W;x) \in \mathcal{F}} \frac{1}{2} \sum_{j \in [L]} \|W_i\|_2^2 \quad s.t. \quad \forall i \in [n] \ \ y_i f(W; x_i) \geq 1, \tag{A.4}$$

then for some universal constant C_1, the following is satisfied:

$$\max_{i \in [L]} {}^{\|W_i^\star\|_{\text{op}}}\!/_{\|W_i^\star\|_F} \ \geq \ \frac{1}{L} \sum_{i=1}^{L} \frac{\|W_i^\star\|_{\text{op}}}{\|W_i^\star\|_F} \ \geq \ C_1^{1/L} \cdot \sqrt{\frac{L}{L+1}} \ . \tag{A.5}$$

Proof.

We will prove this using the result from Lemma A.4.1, and major parts of the proof technique is a re-derivation of some of the results from [Timor et al., 2023], in our setting. From Lemma A.4.1 we know that gradient flow on $\mathcal{F}$ necessarily converges in direction to a KKT point of the optimization problem in Equation A.4. Furthermore, from [Lyu and Li, 2019] we know that this optimization problem satisfies the Mangasarian-Fromovitz Constraint Qualification (MFCQ) condition, which means that the KKT conditions are first-order necessary conditions for global optimality.

We will first construct a wide and deep network $f(W; x) \in \mathcal{F}$, using the network $f(W'; x)$ from the relatively shallower class $\mathcal{F}'$, and then argue about the Frobenius norm weights of the constructed network to be larger than the global optimal of problem in Equation A.4.

Recall that $f(W'; x) \in \mathcal{F}'$ satisfies the following:

$$\forall (x_i, y_i) \in \mathcal{D},\ f(W'; x_i) \cdot y_i \ \geq \ \gamma.$$

Now, we can begin the construction of $f(W; x) \in \mathcal{F}$. Set the scaling factor:

$$\alpha = \left({}^{\sqrt{2}}\!/_{C}\right)^{\frac{L-L'}{L}} .$$

Then, for any weight matrix W_i for $i \in 1, \ldots, L' - 1$, set the value for W_i to be:

$$W_i = \alpha \cdot W'_i = \left(\sqrt{2}/C\right)^{\frac{L-L'}{L}} \cdot W'_i.$$

Let v be the vector of the output layer L' in shallow $f(W'; x)$. Note that this is an m-dimensional vector, since this is the final layer for $f(W'; x)$. But in our construction, layer L' is a layer that includes a fully connected matrix $W_{L'} \in \mathbb{R}^{m \times m}$ matrix.

So for layer L', we set the new matrix to be:

$$W_{L'} = \alpha \cdot \begin{bmatrix} v^\top \\ -v^\top \\ \mathbf{0}_m \\ \vdots \\ \mathbf{0}_m \end{bmatrix},$$

where $\mathbf{0}_m$ is the m-dimensional vector of 0s.

This means that for the L'-th layer in $f(W; x)$ we have the following first two neurons: the neuron that has weights which match the corresponding layer from $f(W'x)$, and the neuron that has weights given by its negation. Note that since the weights in $f(W; x)$ are constructed directly from the weights of $f(W'; x)$, via scaling the weights through the scaling parameter α defined above, we can satisfy the following for every input x for the output of layer L' in $f(W; x)$:

$$f_{L'}(W; x) = \begin{bmatrix} \alpha^k \cdot f(W; x) \\ -\alpha^k \cdot f(W; x) \\ \mathbf{0}_m \\ \vdots \\ \mathbf{0}_m \end{bmatrix}.$$

Next, we define the weight matrices for the layers: $\{L' + 1, \ldots, L\}$. We set the weight matrices $i \in \{L' + 1, \ldots, L - 1\}$ to be:

$$W_i = \left(\frac{\sqrt{2}}{C}\right)^{-\frac{L'}{L}} \cdot \mathbf{I}_m,$$

where $\mathbf{I}_m$ is the $m \times m$ identity matrix. The last layer L in $f(W; x)$ is set to be $[\left(\frac{\sqrt{2}}{C}\right)^{-\frac{L'}{L}}, -\left(\frac{\sqrt{2}}{C}\right)^{-\frac{L'}{L}}, 0, \ldots, 0]^\top \in \mathbb{R}^m$. For this construction, we shall now prove that $f(W'; x) = f(W; x)$ for every input x.

For any input x, the output of layer L' in $f(W'x)$ is:

$$\begin{pmatrix} \text{ReLU}\left(\alpha^k \cdot f(W;x)\right) \\ \text{ReLU}\left(-\alpha^k \cdot f(W;x)\right) \\ \mathbf{0}_m \\ \vdots \\ \mathbf{0}_m \end{pmatrix}.$$

Given our construction for the layers that follow we get for the last but one layer:

$$\begin{pmatrix} \left(\left(\frac{\sqrt{2}}{C}\right)^{-\frac{L'}{L}}\right)^{L-L'-1} \cdot \text{ReLU}\left(\alpha^k \cdot f(W;x)\right) \\ \left(\left(\frac{\sqrt{2}}{C}\right)^{-\frac{L'}{L}}\right)^{L-L'-1} \cdot \text{ReLU}\left(-\alpha^k \cdot f(W;x)\right) \\ \mathbf{0}_m \\ \vdots \\ \mathbf{0}_d \end{pmatrix}.$$

Hitting the above with the last layer $[\left(\frac{\sqrt{2}}{C}\right)^{-\frac{L'}{L}}, -\left(\frac{\sqrt{2}}{C}\right)^{-\frac{L'}{L}}, 0, \ldots, 0]^\top$, we get:

$$\begin{aligned} &\left(\left(\frac{\sqrt{2}}{C}\right)^{-\frac{L'}{L}}\right)^{L-L'-1} \cdot \text{ReLU}\left(\alpha^k \cdot f(W;x)\right) \\ &\quad - \left(\left(\frac{\sqrt{2}}{C}\right)^{-\frac{L'}{L}}\right)^{L-L'-1} \cdot \text{ReLU}\left(-\alpha^k \cdot f(W;x)\right) \\ &\quad = \left(\frac{\sqrt{2}}{C}\right)^{-\frac{L'}{L}\cdot(L-L')} \cdot \left(\frac{\sqrt{2}}{C}\right)^{\frac{L-L'}{L}\cdot L'} \cdot f(W;x) = f(W;x). \end{aligned}$$

Thus, $f(W;x) = f(W';x)$, $\forall x$.

If $W = [W_1, \ldots, W_L]$ be the parameters of wider and deeper network $f(W;x)$ and if $fW^\star;x$ be the network with the parameters $W^\star$ achieving the global optimum of the constrained optimization problem in equation A.4.

Because we have that $f(W^\star;x)$ is of depth $L > L'$ and has m neurons where $m > m'$ and is optimum for the squared ℓ_2 norm minimization problem, we can conclude that $\|W^\star\|_2 \leq$

$\|W\|$. Therefore,

$$\begin{aligned}
\|W^\star\|^2 &\leq \|W'\|^2 \\
&= \left(\sum_{i=1}^{L'-1} \left(\left(\sqrt{2}/C\right)^{L-L'/L\cdot L'}\right)^2 \|W_i\|_F^2\right) \\
&+ \left(\left(\sqrt{2}/C\right)^{L-L'/L\cdot L'}\right)^2 \left(2\,\|W_k\|_F^2\right) \\
&+ (L-L'-1)\left(\left(\sqrt{2}/C\right)^{L'-L/L\cdot L'}\right)^2 \cdot 2 + \left(\left(\sqrt{2}/C\right)^{L'-L/L\cdot L'}\right)^2 \cdot 2 \\
&\leq \left(\left(\sqrt{2}/C\right)^{L-L'/L\cdot L'}\right)^2 C^2(L'-1) + \left(\left(\sqrt{2}/C\right)^{L-L'/L\cdot L'}\right)^2 \cdot 2C^2 \\
&+ (2(L-L'-1)+2)\left(\left(\sqrt{2}/C\right)^{L'-L/L\cdot L'}\right)^2 \\
&= C^2(L'+1)\left(\left(\sqrt{2}/C\right)^{L-L'/L\cdot L'}\right)^2 \left(\left(\sqrt{2}/C\right)^{L-L'/L\cdot L'}\right)^2 \\
&+ 2(L-L')\left(\left(\sqrt{2}/C\right)^{L'-L/L\cdot L'}\right)^2 \\
&= \left(2/C^2\right)^{L-L'/L} C^2(L'+1) + \left(2/C^2\right)^{-L'/L} \cdot 2(L-L') \\
&= 2\cdot\left(2/B^2\right)^{-L'/L}(L'+1) + \left(2/C^2\right)^{-L'/L}\cdot 2(L-L') = 2\cdot\left(2/C^2\right)^{-L'/L}(L+1).
\end{aligned}$$

Since $f^\star$ is a global optimum of equation A.4, we can also show that it satisfies:

$$\|W_i^\star\|_F = \left\|W_j^\star\right\|_F, \quad i<j, \quad i,j\in[L]$$

By the lemma, there is $C^\star > 0$ such that $C^\star = \|W_i^\star\|_F$ for all $i \in [L]$. To see why this is true, consider the net $f(\tilde{W};x)$ where $\tilde{W}_i = \eta W_i^\star$ and $\tilde{W}_j = 1/\eta W_j^\star$, for some $i<j$ and $i,j\in[L]$. By the property of homogeneous networks we have that for every input x, we get: $f(\tilde{W};x) = f(W^\star;x)$. We can see how the sum of the weight norm squares change with a small change in η:

$$\frac{d}{d\eta}(\eta^2\|W_i^\star\| + (1/\eta^2)\|W_j^\star\|_2^2) = 0$$

at $\eta = 1$, since $W^\star$ is the optimal solution. Taking the derivative we get: $\frac{d}{d\eta}(\eta^2\|W_i^\star\| + (1/\eta^2)\|W_j^\star\|_2^2) = 2\eta\|W_i^\star\|_F^2 - 2/\eta^3\|W_j^\star\|_F^2$. For this expression to be zero, we must have $W_i^\star = W_j^\star$, for any $i<j$ and $i,j\in[L]$.

Based on the above result we can go back to our derivation of $\|W^\star\|_F^2 \leq 2\frac{2}{C^2}^{-L'/L}(L+1)$. Next, we can see that for every $i\in[L]$ we have:

$$C^{\star 2} L \leq 2 \frac{2}{C^2}^{-L'/L} (L+1)$$
$$C^{\star 2} \leq 2 \frac{2}{C^2}^{-L'/L} \frac{(L+1)}{L}$$
$$1/C^\star \geq \frac{1}{\sqrt{2}} \left(\frac{C}{\sqrt{2}} \right)^{L'/L} \sqrt{L/L+1}$$

Now we use the fact that $\forall x \in \mathcal{X}$, the norm $\|x\|_2 \leq 1$:

$$1 \leq y_i f^\star(x_i) \leq |f^\star(x_i)| \leq \|x_i\| \prod_{i \in [L]} \|W_i^\star\|_{\text{op}} \leq \prod_{i \in [L]} \|W_i^\star\|_{\text{op}} \leq \left(\frac{1}{L} \sum_{i \in [L]} \|W_i^\star\|_{\text{op}} \right)^L ,$$

Thus: $\frac{1}{L} \sum_{i \in [L]} \|W_i^\star\|_{\text{op}} \geq 1$. Plugging this into the lower bound on $\frac{1}{C^\star}$:

$$\frac{1}{L} \sum_{i \in [L]} \frac{\|W_i^\star\|_{\text{op}}}{\|W_i^\star\|_F} = \frac{1}{C^\star} \cdot \frac{1}{L} \sum_{i \in [L]} \|W_i^\star\|_{\text{op}} \geq \frac{1}{\sqrt{2}} \left(\frac{C}{\sqrt{2}} \right)^{L'/L} \sqrt{L/L+1} \cdot 1 \tag{A.6}$$
$$= \frac{1}{\sqrt{2}} \cdot \left(\frac{\sqrt{2}}{C} \right)^{\frac{L'}{L}} \cdot \sqrt{\frac{L}{L+1}} \, . \tag{A.7}$$

This further implies that $\forall i \in [L]$:

$$\|W_i^\star\|_F \leq \|W_i^\star\|_{\text{op}} \sqrt{2} \cdot \frac{\sqrt{2}}{C}^{-L'/L} \cdot \sqrt{L+1/L}.$$

Setting $C' = \frac{1}{\sqrt{2}} \cdot \left(\frac{\sqrt{2}}{C} \right)^{L'}$ we get the final result:

$$\|W_i^\star\|_F \leq \|W_i^\star\|_{\text{op}} \cdot C_1^{1/L} \cdot \sqrt{\frac{L+1}{L}}, \quad \forall i \in [n].$$

From Lemma A.4.4 we know that at each layer the weight matrices are almost rank 1, i.e., for each layer $j \in [L]$, there exists a vector v_j, such that $\|v_j\|_2 = 1$ and $W_j \approx \sigma_j v_j v_j^\top$ for some $\sigma_j > 0$. More formally, we know that for any $L > L'$ and $m \geq m'$ satisfying the conditions in Lemma A.4.4, for every layer j we have:

$$\|(I - v_j v_j^\top) \hat{W}_j\|_2 = \sqrt{\|\hat{W}_j\|_F^2 - \sigma_j^2} \tag{A.8}$$

Next, we can substitute the previous bound that we derived: $\|\hat{W}_j\|_F \leq \sigma_j C'^{1/L} \cdot \sqrt{\frac{L+1}{L}}$ to get the following, when gradient flow converges to the globally optimal solution of equation A.4:

$$\|(I - v_j v_j^\top)\hat{W}_j\|_2 \leq \sigma_j \sqrt{C'^{2/L} \cdot {}^{L+1}/_{L} - 1} \leq \sigma_j C'^{1/L} \sqrt{\frac{L+1}{L}} \tag{A.9}$$

Now we are ready to derive the final bound on the last layer activations:

$$\begin{aligned}
&\mathbb{E}_{x\sim P_{\text{OOD}}}|f(\hat{W};x)|_2 = \mathbb{E}_{P_{\text{OOD}}}|W_L\sigma(W_{L-1}\sigma(W_{L-2}\dots\sigma(W_2\sigma(W_1x))\dots))| \\
&= \mathbb{E}_{P_{\text{OOD}}}|W_L\sigma(W_{L-1}\sigma(W_{L-2}\dots\|f_j(\hat{W})\|_2 \cdot \frac{f_j(\hat{W}x)}{\|f_j(\hat{W};x)\|}\dots))| \\
&= \|f_j(\hat{W})\|_2 \cdot \mathbb{E}_{P_{\text{OOD}}}|W_L\sigma(W_{L-1}\sigma(W_{L-2}\dots\cdot \frac{f_j(\hat{W}x)}{\|f_j(\hat{W};x)\|}\dots))| \\
&\leq \mathbb{E}_{P_{\text{OOD}}}\|f_j(\hat{W})\|_2 \cdot \prod_{z=j+1}^{L} C_0^{1/L} \leq \mathbb{E}_{P_{\text{OOD}}}\|f_j(\hat{W})\|_2 \cdot C_0
\end{aligned}$$

where the final inequality repeatedly applies Cauchy-Schwartz on a norm one vector: $f_j(\hat{W};x)/\|f_j(\hat{W};x)\|_2$, along with another property of ReLU activations: $\|\sigma(v)\|_2 \leq \|v\|_2$.

$$\begin{aligned}
\mathbb{E}_{P_{\text{OOD}}}\|f_j(\hat{W})\|_2 &\leq \sqrt{\mathbb{E}_{P_{\text{OOD}}}\|f_j(\hat{W})\|_2^2} \\
&\leq \sqrt{\mathbb{E}_{P_{\text{OOD}}}(\sigma_j^2\|v_j v_j^\top f(\hat{W};x)\|_2^2 + \|(I - v_j v_j^\top)\hat{W}_j\|_2^2 C_0^2)} \\
&\leq \sqrt{\sigma_j^2\epsilon^2 + \|(I - v_j v_j^\top)\hat{W}_j\|_2^2 C_0^2}
\end{aligned}$$

Since, $\mathbb{E}_{P_{\text{OOD}}} \leq \|v_j v_j^\top \hat{W}_j\|_2$ and

$$\begin{aligned}
\mathbb{E}_{P_{\text{OOD}}}\|f_j(\hat{W})\|_2 &\leq \sigma_j \sqrt{(\epsilon^2 + \|(I - v_j v_j^\top)f_j(\hat{W})\|_2)} \\
&\leq \sigma_j \sqrt{(\epsilon^2 + C'^{2/L} \cdot {}^{L+1}/_{L})} \\
&\leq \sigma_j(\epsilon + C'^{1/L} \cdot \sqrt{{}^{L+1}/_{L}})
\end{aligned}$$

Recall that $\sigma_j \leq C_0^{1/L}$. From the above, we get the following result:

$$\mathbb{E}_{x\sim P_{\text{OOD}}}|f(\hat{W};x)|_2 \leq C_0\mathbb{E}_{P_{\text{OOD}}}\|f_j(\hat{W})\|_2 \leq C_0(\epsilon + C'^{1/L} \cdot \sqrt{{}^{L+1}/_{L}}),$$

which completes the proof of Theorem A.4.3.

Bias learnt for nearly homogeneous nets

In this subsection, we analyze a slightly modified form of typical deep homogeneous networks with ReLU activations. To study the accumulation of model constants, we analyze the class of functions $\tilde{\mathcal{F}} = \{f(W;\cdot) + b : b \in \mathbb{R}, f(W;\cdot) \in \mathcal{F}\}$, which consists of deep homogeneous networks with a bias term in the final layer. In Proposition 2.4.2, we show that there exists a set of margin points (analogous to support vectors in the linear setting) which solely determines the model's bias $\hat{b}$.

Proposition A.4.2 (Analyzing network bias) *If gradient flow on $\tilde{\mathcal{F}}$ converges directionally to $\hat{W}, \hat{b}$, then $\hat{b} \propto \sum_k y_k$ for margin points $\{(x_k, y_k) : y_k \cdot f(\hat{W}; x_k) = \arg\min_{j\in[N]} y_j \cdot f(\hat{W}; x_j)\}$.*

Proof.

Lemmas C.8, C.9 from [Lyu and Li, 2019] can be proven for gradient flow over $\tilde{F}$ as well since all we need to do is construct $h_1, \ldots, h_N$ such that h_i satisfoes forst order stationarity for the i^{th} constraint in the following optimization problem, that is only lightly modified version of problem instance P in [Lyu and Li, 2019]:

$$\min_{W,b} L(W, b; \mathcal{D}) := \sum_{i=1}^{L} \|W_i\|_F^2$$
$$\text{s.t.} \quad y_i f(W; x_i) \geq 1 - b, \quad \forall i \in [N]$$

Note that the above problem instance also satisfies MFCQ (Mangasarian-Fromovitz Constraint Qualification), which can also be shown directly using Lemma C.7 from [Lyu and Li, 2019].

As a consequence of the above, we can show that using gradient flow to optimize the objective:

$$\min_{W,b} \frac{1}{N} \sum_{(x,y)\in\mathcal{D}} \exp\left(-y \cdot (f(W; x) + b)\right),$$

also converges in direction to the KKT point of the above optimization problem with the nearly homogeneous networks $\mathcal{F}$. This result is also in line with the result for linear non-homogeneous networks derived in [Soudry et al., 2018].

Finally, at directional convergence, the gradient of the loss $\frac{\partial L(W;\mathcal{D})}{\partial W}$ converges, as a direct consequence of the asymptotic analysis in [Lyu and Li, 2019].

Let us denote the set of margin points at convergence as $\mathcal{M} = \{(x_k, y_k) : y_k f(W; x_k) = \min_{j\in[N]} y_j f(W; x_j)\}$. These, are precisely the set of points for which the constraint in the above optimization problem is tight, and the gradients of their objectives are the only contributors in the construction of $h_1, \ldots, h_N$ for Lemma C.8 in [Lyu and Li, 2019]. Thus, it is easy to see that at convergence the gradient directionally converges to the following value, which is purely determined only by the margin points in $\mathcal{M}$.

$$\lim_{t\to\infty} \frac{\frac{\partial}{\partial W} L(W, b; \mathcal{D})}{\|\frac{\partial}{\partial W} L(W, b; \mathcal{D})\|_2} = -\frac{\sum_{k\in\mathcal{M}} y_k \cdot \nabla_W f(W; x_k)}{\|\sum_{k\in\mathcal{M}} y_k \cdot \nabla_W f(W; x_k)\|_2}$$

Similarly we can take the derivative of the objective with respect to the bias b, and verify its direction. For that, we can note that: $\frac{\partial \exp(-y(f(W;x)+b))}{\partial b} = -y \cdot \exp(-y(f(W; x) + b))$, but more importantly, $\exp(-y(f(W; x) + b)$ evaluates to the same value for all margin points in $\mathcal{M}$. Hence,

$$\lim_{t\to\infty} \frac{\frac{\partial}{\partial b} L(W, b; \mathcal{D})}{\|\frac{\partial}{\partial b} L(W, b; \mathcal{D})\|_2} = -\frac{\sum_{k\in\mathcal{M}} y_k}{|\sum_{k\in\mathcal{M}} y_k|}$$

While both $\hat{b}$ and $\hat{W}$ have converged directionally, their norms keep increasing, similar to analysis in other works [Soudry et al., 2018, Huh et al., 2021, Galanti et al., 2022, Lyu and Li, 2019, Timor et al., 2023]. Thus, from the above result it is easy to see that bias keeps increasing along the direction that is just given by the sum of the labels of the margin points in the binary classification task. This direction also matches the OCS solution direction (that only depends on the label marginal) if the label marginal distribution matches the distribution of the targets y on the support points. This completes the proof of Proposition A.4.2.

Appendix B

Appendices for Large Language Model Hallucinations

B.1 Unfamiliarity Metrics

In this section, we provide more details on the metrics we used in our experiments to measure the unfamiliarity of a query or example with respect to a model.

Number of Mentions in Pretraining Corpus

One metric we used is the number of times concepts from the query are mentioned in the model's pretraining corpus. To measure this quantity, we made use of the work of Kandpal et al. [2023], which entity linked pretraining datasets for a variety of models, providing dictionaries which linked entity names to documents which mention the entity in the pretraining dataset. Because we used Llama2-7B as the pretrained model in all of our experiments, we used the dictionary associated with The Pile dataset [Gao et al., 2020], which is representative of the data used to pretrain Llama2-7B models.

We used this metric to evaluate the behavior of our models for TriviaQA, and biography and bio generation. To calculate the number of times a query is mentioned in the pretraining corpus, we first mapped the query to relevant entities, then, using the provided dictionary, we measured the number of intersecting documents that are associated with all relevant entities. For TriviaQA, Kandpal et al. [2023] additionally provided a mapping from questions to relevant entities, which we directly used. For biography and bio generation, we used the name of the person or title of the book/movie as the entity for measuring the unfamiliarity of the query.

Pretrained Model Prediction Uncertainty

To measure the unfamiliarity of queries for our MMLU experiments, we used the uncertainty of the pretrained model's predictions. More specifically, we prompt the pretrained model

with 5 examples of queries and responses, and then with our target query. We then normalize the model's output across the answer choices (A-D), and used the negative log likelihhood of the models top prediction answer as a measure of the query's unfamiliarity. We used this metric to measure both the unfamiliarity of input queries at test time and the unfamiliarity of examples during finetuning.

SFT Model Prediction Uncertainty

To measure the unfamiliarity of finetuning examples for TriviaQA, we used the uncertainty of predictions from a model that has been finetuned on the task. More specifically, we first finetune a model on the TriviaQA training set. Then, we pass all training examples through the finetuned model, and measured the negative log likelihood (averaged over tokens) associated with the model's top predicted answer. Unfamiliar finetuning examples tend to be associated with higher NLL predictions,

We use this metric to determine the unfamiliarity of finetuning examples (rather than the number of mentions in the pretraining corpus, which we used for evaluation) because this metric is more general and straightforward to acquire, making it more generalizable to other tasks. While the number of mentions in the pretraining corpus is a more direct measurment of an example's unfamiliarity to a model, entity linking a model's pretraining corpus is a significant undertaking, and might not be possible for all tasks or models. Because we want the metric used in our finetuning approach to be broadly applicable to different tasks and models, we instead make use of this easier-to-acquire metric in our finetuning procedure.

B.2 MMLU Training Details

In this section, we provide more details on our training and evaluation procedure for our MMLU experiments. For all experiments, we finetuned on the evaluation split of MMLU, and evaluated on the validation split. This is because MMLU does not have a training split.

SFT Models

We use the metric described in Appendix Sec. B.1 to determine whether the familiarity of a finetuning example. We classify examples with NLL greater than 0.7 as unfamiliar, and the rest as familiar. During finetuning, we rebalance the dataset such that 50% of finetuning examples are familiar and 50% are unfamiliar.

RL Models

We initialize all RL finetuning with a model that has already be supervised finetuned to produce responses that consist of answer choices. The SFT model we used for initialization

is trained predict the E option 50% of the time, and to produce the correct answer to the query 50% of the time.

B.3 TriviaQA Training Details

To determine the unfamiliarity of a finetuning example, we use the metric described in Appendix Sec. B.1. We classify an example as being unfamiliar if the example's NLL is greater than 0.045, and familiar otherwise. We relabel the responses associated with all unfamiliar finetuning examples to be "I don't know". To determine the unfamiliarity of an evaluation query, we use the metric described in Appendix Sec. B.1.

B.4 Long-form Tasks Training Details

In this section, we provide training and evaluation details for our long-form factuality finetuning experiments.

Data

We construct finetuning and evaluation datasets using WikiBios and WikiPlots, both of which consist of wikipedia entries attached to people and books/movies. We make use of the first sentence in the wikipedia entry for both tasks as the target response in our SFT finetuning datasets. The prompts we use for finetuning are "Write a biography for [name]." and "What is the promise of [title]?". For the biography task, our finetuning dataset includes 104539 examples, and our evaluation dataset includes 5000 examples. For the plot generation task, our finetuning dataset includes 10000 examples, and our evaluation dataset includes 4795 examples.

Reward Model Learning

We take a two-staged approach to learning a reward model. First, we trained a model to break down a response into individual atomic facts. Next, we trained a separate model to predict the factuality of each atomic fact. We then use the predicted factuality of each fact to calculate the overall reward associated with each response. The supervision for both models are collected by querying FActScore, which is a automated pipeline that queries GPT-3.5 to decompose a response into atomic facts and produces the factuality of each atomic fact. We use 10000 labeled examples to train the conservative reward model and the standard reward models each for both tasks. Note that while we use a two-staged strategy for learning reward models in our implementation, our general approach for learning conservative reward model should apply to other reward model learning strategies as well, such as directly predicting the reward associated with a response.

Policy Learning

We initialize all RL finetuning with the SFT model, and use the reward predicted by the reward model described above as supervision.

Appendix C

Appendices for Large Language Model Reasoning Generalization

C.1 Selection of Memorization Threshold

We find the threshold p by sweeping across a range of values, calculating the pre-memorization train accuracy across different training runs, and selecting the value which yields the strongest predictor of test accuracy. In Fig. C.1, we illustrate how the value of p influences the R^2 for predicting average test accuracy. We can see that R^2 degrades smoothly with respect to p, which makes it is relatively easy to find a good value of p by sweeping a range of values.

This calibration process only requires a small number of training runs (e.g. 1-3) to arrive at a robust value of p which can generalize to new training runs on the same model and finetuning dataset, illustrated in Fig. C.2. However, it is important the the training

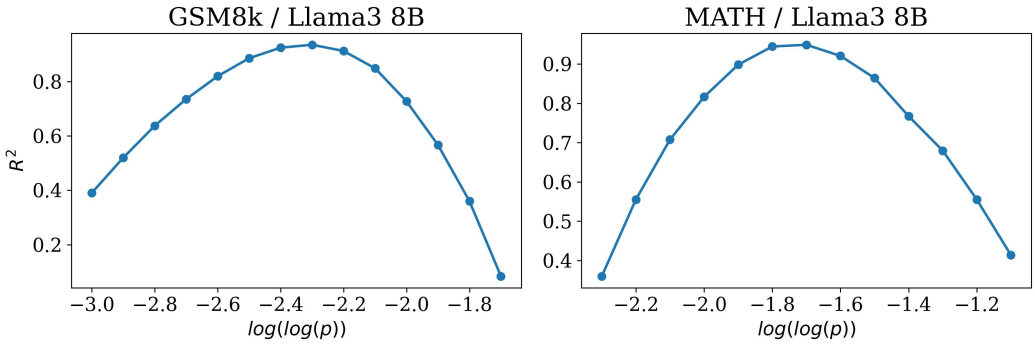

Figure C.1: Relationship between the value of p and the coefficient of determination (R^2) with respect to pre-memorization train accuracy and test accuracy. The R^2 is taken in aggregate of all the corresponding training runs in Fig. 4.4.

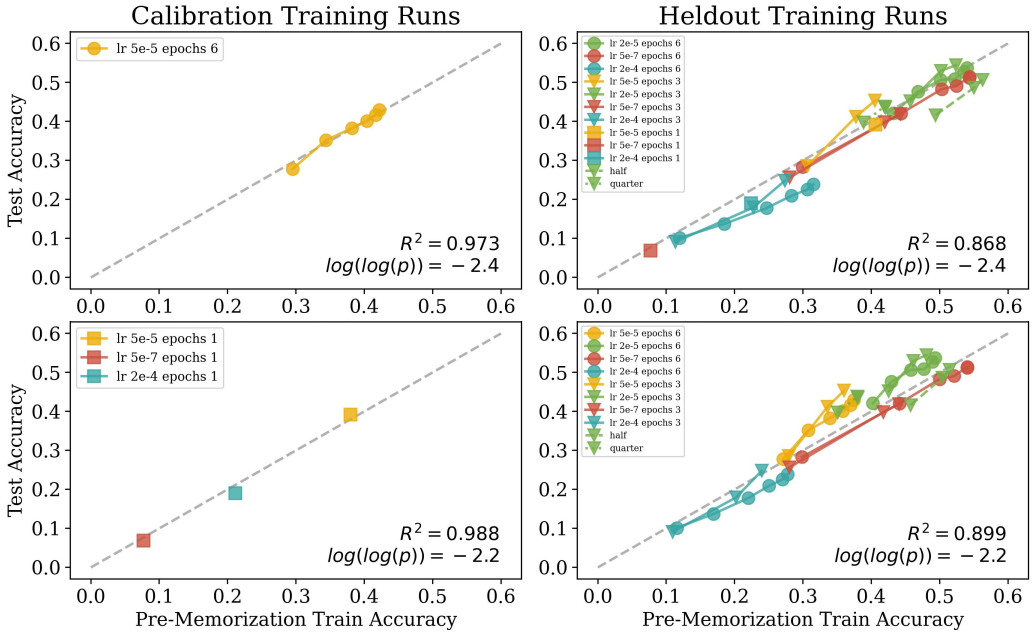

Figure C.2: Calibrating p on a subset of training runs, and evaluating R^2 on heldout training runs using GSM8k and Llama3 8B. We can see that calibrating on just 1-3 training runs was able to yield a robust value of p which leads to high R^2 on heldout training runs.

runs used for calibration exhibit some spread over test accuracies, and memorization during training.

Finally, we also show that the calibration process generalizes to new test examples. We divide the test set into two halves: a calibration test set, and a heldout test set. We calibrate p on the calibration test set, and evaluate the coefficient of determination of the heldout test set. In Fig. C.3, we can see that the value of p is able to generalize robustly to new examples on which it had not been calibrated, achieving high coefficient of determination.

C.2 Section 4.4 Training Runs Details

In this section, we will enumerate all training runs shown in Fig. 4.4 and their training details. For our half and quarter training runs, we fix the total number of training steps to be equivalent to training for 3 epochs on the full dataset.

GSM8k LLama3 8B

For all training runs with GSM8k and Llama3 8B, we use the AdamW optimizer, with a linear decay learning rate scheduler with 20 warmup steps, a batch size of 128, and a max

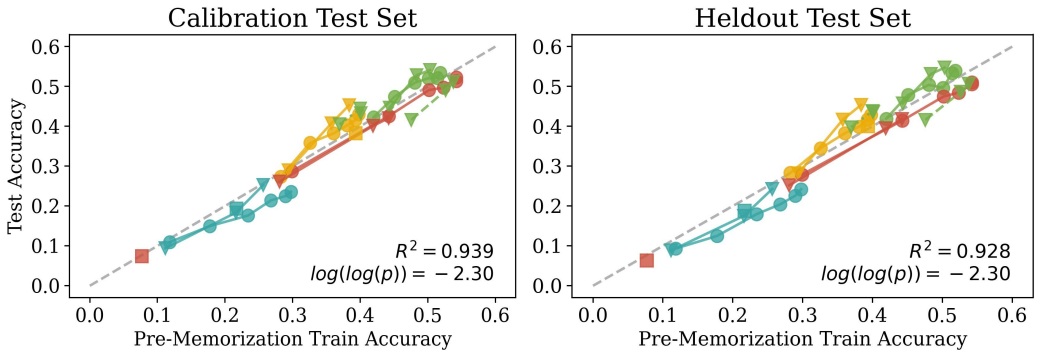

Figure C.3: Calibrating p using a subset of of the test set (calibration test set), and evaluating R^2 on a heldout test set using GSM8k and Llama3 8B. We can see that calibrating on just the calibration test set was able to yield a robust value of p which leads to high R^2 on the heldout test set.

gradient norm of 2.

Learning Rate	Epochs	Dataset Size
5e-5	6	full
2e-5	6	full
5e-7	6	full
2e-4	6	full
5e-5	3	full
2e-5	3	full
5e-7	3	full
2e-4	3	full
5e-5	1	full
5e-7	1	full
2e-4	1	full
2e-5	6	half
2e-5	12	quarter

MATH LLama3 8B

For all training runs with MATH and Llama3 8B, we use the AdamW optimizer, with a linear decay learning rate scheduler with 20 warmup steps, a batch size of 24, and a max gradient norm of 2.

Learning Rate	Epochs	Dataset Size
5e-5	6	full
5e-7	6	full
2e-4	6	full
5e-5	3	full
5e-7	3	full
2e-4	3	full
5e-5	1	full
5e-7	1	full
2e-4	1	full
2e-5	6	half
2e-5	12	quarter

GSM8k Gemma2 9B

For all training runs with GSM8k and Gemma2 9B, we use the Adam optimizer, with a cosine decay learning rate scheduler with (0.1*total steps) warmup steps, a batch size of 32, and a max gradient norm of 1.

Learning Rate	Epochs	Dataset Size
5e-4	6	full
5e-5	6	full
5e-6	6	full
5e-7	6	full
5e-4	3	full
5e-5	3	full
5e-6	3	full
5e-7	3	full
5e-4	1	full
5e-5	1	full
5e-6	1	full
5e-7	1	full
5e-5	6	half
5e-5	12	quarter

MATH Gemma2 9B

For all training runs with MATH and Gemma2 9B, we use the Adam optimizer, with a cosine decay learning rate scheduler with (0.1*total steps) warmup steps, a batch size of 32, and a max gradient norm of 1.

Learning Rate	Epochs	Dataset Size
5e-4	6	full
5e-5	6	full
5e-6	6	full
5e-7	6	full
5e-4	3	full
5e-5	3	full
5e-6	3	full
5e-7	3	full
5e-4	1	full
5e-5	1	full
5e-6	1	full
5e-7	1	full
5e-5	6	half
5e-5	12	quarter

C.3 Section 4.4 Prior Generalization Metrics

In this section we will more precisely describe each generalization metric.

Gradient Variance

We calculate the gradient of the model for 5 different minibatches, take the variance across the 5 samples for each element of each weight matrix, and take the average over each element of the model weights.

Distance from Initialization

We calculate the squared difference between each element of the model weights at initialization and after finetuning, and take the sun across all elements.

Average Thresholded Confidence (ATC)

ATC computes a threshold on a score computed on model confidence such that the fraction of examples above the threshold matches the test accuracy. For the score, we use the likelihood of greedily sampled responses under the model. We calculate the the score over the training data using a model trained for 3 epochs using learning rate 2e-5, and calculate the threshold over the score using the test dataset. We then predict the test accuracies over different models in our experiment by calculating the score associated with the training data using each model, and measuring the percentage of examples whose score surpass the threshold that we previously calculated.

C.4 Section 4.5 Implementation Details

For our approach for data curation, we implemented the process described in Algorithm 1, with 5 iterations (n) and using threshold (t) 0.75 for both GSM8k and MATH.

For the IFD approach for data curation, we calculated the IFD score using a model that was train on the test set associated each dataset for 2 epochs. This is because, in order to calculated the IFD score, we need a model which has been briefly trained for the task of interest, but which has not been exposed to the dataset for which we want to calculate the IFD score over. Note that this model is only used for calculating for the IFD score, and not used for evaluations in our experiments, so there is no data leakage.

For both the IFD approach and the heuristic approach, we take $P'(x)$ to be top 50 percentile of examples for GSM8k, and top 75 percentile of examples for MATH. We designed these percentiles to roughly match the percentile of examples that our approach selects from.

For all training runs, we use the AdamW optimizer, with a linear decay learning rate scheduler with 20 warmup steps, a batch size of 128, a max gradient norm of 2, a learning rate of 2e-5, and 3 epochs of training.