Reliable Representation Learning: Theory and Practice

Yaodong Yu

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2024-75
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2024/EECS-2024-75.html

May 10, 2024

Reliable Representation Learning: Theory and Practice

by

Yaodong Yu

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 Yi Ma, Co-chair
Professor Michael I. Jordan, Co-chair
Professor Jacob Steinhardt
Professor Song Mei

Spring 2024

Reliable Representation Learning: Theory and Practice

Abstract

Reliable Representation Learning: Theory and Practice

by

Yaodong Yu

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Yi Ma, Co-chair

Professor Michael I. Jordan, Co-chair

Machine learning models trained on vast amounts of data have achieved remarkable success across various applications. However, they also pose new challenges and risks for deployment in real-world high-stakes domains. Decisions made by deep learning models are often difficult to interpret, and the underlying mechanisms remain poorly understood, and large-scale foundational models can memorize and leak private personal information. Given that deep learning models operate as black-boxes, it is challenging to understand, let alone resolve, various types of failures in current machine learning systems.

In this dissertation, we present research towards building reliable machine learning systems through the lens of representation learning. The first part focuses on transparent representation learning. We first propose a principled and effective objective function, called coding rate reduction, for measuring the goodness of representations, and present a white-box approach to understanding transformer models. We then show how to derive a family of mathematically interpretable transformer-like deep network architectures by maximizing the information gain of the learned representations. The second part focuses on privacy-preserving representation learning. We first present our investigation on understanding the effectiveness of learned representations using federated optimization methods, and present our approach for overcoming data heterogeneity when training deep, non-convex models in the federated setting. Next, we describe our work on training the first set of vision foundation models with rigorous differential privacy guarantees, and demonstrate the promise of high-utility differentially private representation learning.

To my parents, Yueju Wang and Yong Yu.

Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, I must begin by expressing my deepest gratitude to my advisors, Yi Ma and Michael Jordan, for your invaluable guidance and unconditional support throughout this journey. I feel deeply privileged to have had the opportunity to learn from you. Your breadth of knowledge, vision, and enthusiasm have been, and continue to be, a constant source of inspiration for me. Thank you Mike, your dedicated pursuit of knowledge and exploration across numerous fields has had a profound and inspiring impact on my own academic journey. Thank you Yi, your boundless passion and energy have been a guiding light throughout my journey. Thank you for being the best advisors I could ever ask for. I aspire to have absorbed some of your exceptional mentoring abilities, striving to be as excellent an advisor to my students as you have been to me.

I was fortunate enough to also have had the opportunity to learn from Jacob Steinhardt; working with you has been a great honor and pleasure. Thank you to Jacob for your invaluable mentorship, guidance, and support throughout my journey. Thank you for connecting me with your lab, where I learned so much on various topics. I would like to thank Song Mei for serving on my qualification exam committee and dissertation committee. Thank you to Song for providing valuable feedback on preparing my presentation and for your encouragement on my research.

I want to thank the amazing mentors during my journey at Berkeley, including Bin Yu, Haiyan Huang, Jiantao Jiao, and Zeyu Zheng. I am grateful to Bin for your invaluable feedback on my research and advice for my career. I want to express my gratitude to Haiyan Huang for welcoming me into the field of precision medicine and for providing guidance and advice beyond research. I want to thank Jiantao for your mentorship and guidance as I started at Berkeley. It has been my absolute pleasure to collaborate with you on ML robustness, from which I have gained valuable insights through our discussions. I owe special thanks to Zeyu for being so generous with your time and for your advice and guidance throughout my journey.

I especially want to thank Sinno Jialin Pan for taking a chance on me, even though I had no background in machine learning before joining your lab at NTU. Thank you for introducing me to the field of machine learning and helping me build my very first research interests. Thank you for providing me with the freedom and guidance for any ideas I pursued.

Many thanks to the mentors during my stay at Fundamental AI Research (FAIR), Meta: Maziar Sanjabi, Chuan Guo, and Kamalika Chaudhuri. Thank you for introducing me to the field of privacy-preserving machine learning. I consider myself extremely fortunate to have had the opportunity to work with you on building foundational models with differential privacy over the past two years. I want to thank you for your encouragement and support when the very first project did not go well. Having lunch with Maziar and Chuan and chatting together was one of the highlights for me every week. I am grateful to Kamalika for your generous support during my job search. I would also like to thank Tom Sander for being such a great collaborator on the differentially private vision-language model project.

I am grateful to have had the opportunity to learn from Quanquan Gu during my master's program at the University of Virginia. Thank you to Quanquan for your mentorship and support. Working with you on non-convex optimization has been a great pleasure. I would like to thank you, Yangfeng Ji and Alfred Weaver, for your support during my master's program.

I would like to thank the remarkable staff members at Berkeley, Shirley Salanio, Jean Nguyen, Jessica Gamble, and Naomi Yamasaki. I am grateful to Shirley for your support throughout all stages of my PhD. I would like to thank Jean for your the logistics for my qualification exam and dissertation. Thanks to Jessica for your assistance with travel, reimbursement, and GSR appointment. Thank you to Naomi for booking rooms for my qualification exam and dissertation talk.

Throughout my PhD, I was lucky to have the opportunity to work with a set of wonderful and brilliant collaborators: Dara Bahri, Hao Bai, Stephen Bates, Jarosław Błasiok, Sam Buchanan, Eunice Chan, Kwan Ho Ryan Chan, Kamalika Chaudhuri, Jianfeng Chi, Tianzhe Chu, Xili Dai, Edgar Dobriban, Alain Durmus, Dhruba Ghosh, Gauthier Gidel, Joseph Gonzalez, Chuan Guo, Benjamin Haeffele, Tom Hu, Haiyan Huang, Yifei Huang, Heinrich Jiang, Michael Jordan, Sai Praneeth Karimireddy, Rajiv Khanna, Seungyeon Kim, Bogdan Kulynych, Nicolas Le Roux, Chris Junchi Li, Xianhang Li, Tianyi Lin, Nicolas Loizou, Charles Lu, Yi Ma, Eric Mazumdar, Hossein Mobahi, Preetum Nakkiran, Aldo Pacchiano, Druv Pai, Alan Pham, Michael Psenka, Haozhi Qi, Qing Qu, Ramesh Raskar, Ankit Singh Rawat, Tom Sander, Maziar Sanjabi, Heung-Yeung Shum, Chaobing Song, Vikranth Srivatsa, Jacob Steinhardt, Yuan Tian, Shengbang Tong, Andreas Veit, Peng Wang, Alex Wei, John Wright, Ziyang Wu, Cihang Xie, Eric Xing, Yao-Yuan Yang, Yaoqing Yang, Jinrui Yang, Zitong Yang, Yuan Yao, Chong You, Yuexiang Zhai, Hongyang Zhang, Yi-Fan Zhang, Hanlin Zhang, Han Zhao, Ruiqi Zhong, Yuyin Zhou, Zhihui Zhu, and the many others. I am deeply grateful for the synergy created through our collaborations. Thank you for all the things you taught me.

I want to thank Tom Hu for your invaluable advice and patient responses to my myriad of questions throughout my journey at Berkeley. I want to thank Weitong Wang for being a wonderful roommate and gym training partner during my undergraduate years at Nanjing University and my first year at Berkeley. I want to thank Hongyang Zhang for your mentorship and help during my second year of master's program. I want to thank I want to thank Xiao Zhang for driving me to dinners and grocery stores at Charlottesville.

I am immensely grateful to all the members of Mike's group (SAIL) and Yi's group. My sincere thanks also go to all the members of Jacob's group. I was incredibly fortunate to have the opportunity to learn from and be inspired by these three incredible groups. Special thanks to Haozhi Qi, Brent Yi, and Yichao Zhou for helping me set up the compute server; and to Tianyi Lin, Stephen Bates, and Sai Praneeth Karimireddy for your mentorship and guidance.

To all my friends, your presence has filled my PhD journey with vibrant colors and unforgettable moments. I am profoundly grateful for each and every one of you who has been a part of this incredible experience.

I would like to thank Sulin Liu, Yunxiang Liu, and Mengchen Zhao, a special group of friends whom I met at NTU and have been in close contact with since then. Interacting with you all has been a constant source of joy throughout my PhD journey.

I would like to express my sincere gratitude to my mentors, colleagues, and friends for your incredible support during my job search in the final year of my PhD. Your invaluable advice, constructive feedback, and encouraging words were instrumental in helping me navigate the intricate landscape of the academic job market. I look forward to carrying forward the wisdom I have learned from you into the next chapter of my career.

My final thanks go to my family. In particular, I would like to thank my wife, Honglu Zhang, for your unwavering support and accompanying me through all the ups and downs of this journey, for the beautiful memories you have brought, and for having always been there for me. I want to thank my brother, Yaolong Yu, for your support and encouragement. Lastly, thank you Mom and Dad for always having faith in me.

Chapter 1

Introduction

In recent years, the practice of machine learning—particularly deep learning models trained at immense scales—has captured people's imagination by its empirical successes in learning useful representations from real-world high-dimensional and multi-modal data [145, 100, 250, 201]. Much of this success is owed to deep networks' ability in effectively learning compressible low-dimensional structures in the data distribution and then transforming the distribution to a *compact and structured* representation. Such a representation then facilitates many downstream tasks, e.g., classification [145, 100, 61, 63, 26], recognition and segmentation [77, 168, 76, 29, 140], and generation [81, 136, 108, 201, 212, 214].

1.1 Representation Learning

To state the common problem behind all these practices more formally, one may view a given dataset as samples of a random vector $\boldsymbol{x}$ in a high-dimensional space, say $\mathbb{R}^D$. Typically, the distribution of $\boldsymbol{x}$ has much lower intrinsic dimension than the ambient space. Generally speaking, by *learning a representation*, we typically mean to learn a continuous mapping parameterized by $\boldsymbol{\theta}$, say $f(\cdot;\boldsymbol{\theta})$, that transforms $\boldsymbol{x}$ to a so-called *feature vector* $\boldsymbol{z}$ in another (typically lower-dimensional) space, say $\mathbb{R}^d$. It is hopeful that through such a mapping:

$$\boldsymbol{x} \in \mathbb{R}^D \xrightarrow{f(\boldsymbol{x};\boldsymbol{\theta})} \boldsymbol{z} \in \mathbb{R}^d, \tag{1.1}$$

the low-dimensional intrinsic structures of $\boldsymbol{x}$ are identified and represented by $\boldsymbol{z}$ in a more compact and structured way so as to facilitate subsequent tasks such as classification or generation. The feature $\boldsymbol{z}$ can be viewed as a (learned) compact code for the original data $\boldsymbol{x}$, so the mapping f is also called an *encoder*.

In recent developments, the predominant practice has been to learn first a *task-agnostic* representation by pre-training a large deep neural network, in some cases known as a *foundation model* [21]. The so-learned representation can subsequently be fine-tuned for multiple specific tasks. This has been shown to be more effective and efficient for many practical tasks across diverse data modalities, including speech [203], language [26], natural images [190], and vision-language [201]. Notice that representation learning in this context is very different from that for

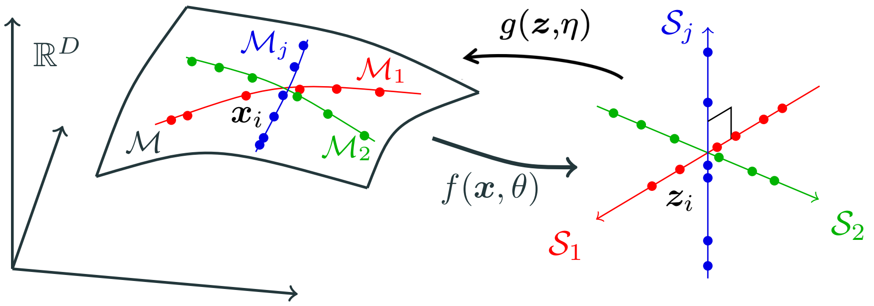


Figure 1.1: The distribution $\mathcal{D}$ of high-dim data $\boldsymbol{x}$ is supported on a manifold $\mathcal{M}$ and its classes on low-dim submanifolds $\mathcal{M}_j$, we aim to learn a mapping $f(\boldsymbol{x};\boldsymbol{\theta})$ such that it maps complicated high-dimensional data to a compact and structured representation (e.g., a union of maximally uncorrelated subspaces $\{\mathcal{S}_j\}$), and to learn a mapping $g(\boldsymbol{z};\boldsymbol{\eta})$ to (approximately) regenerate the original data.

a specific task, where $\boldsymbol{z}$ only needs to be good enough for predicting a specific label $\boldsymbol{y}$ (as in supervised learning [218]). In a task-agnostic setting, the learned representation $\boldsymbol{z}$ needs to encode *almost all essential information about the distribution of the data* $\boldsymbol{x}$. That is, the learned representation $\boldsymbol{z}$ not only is a more compact and structured representation for the intrinsic structures of $\boldsymbol{x}$, but can also recover $\boldsymbol{x}$ to a certain degree of faithfulness. Conceptually, one effective way to verify whether a representation $\boldsymbol{z}$ has encoded sufficient information about $\boldsymbol{x}$ is to see how well we can recover $\boldsymbol{x}$ from $\boldsymbol{z}$ through an (inverse) mapping, say $g(\cdot;\boldsymbol{\eta})$, known as a *decoder* (or a generator):

$$\boldsymbol{x} \in \mathbb{R}^D \xrightarrow{f(\boldsymbol{x};\boldsymbol{\theta})} \boldsymbol{z} \in \mathbb{R}^d \xrightarrow{g(\boldsymbol{z};\boldsymbol{\eta})} \widehat{\boldsymbol{x}} \in \mathbb{R}^D. \tag{1.2}$$

As the encoder f is typically compressive and lossy, we should not expect the inverse mapping to recover $\boldsymbol{x}$ exactly, but an approximate $\widehat{\boldsymbol{x}} = g \circ f(\boldsymbol{x}) \approx \boldsymbol{x}$. We normally seek optimal encoding and decoding mappings such that the decoded $\widehat{\boldsymbol{x}}$ is the closest to $\boldsymbol{x}$, either sample-wise—say, by minimizing the expected mean squared error—or in a relaxed distributional sense. We refer to the above process as *compressive encoding and decoding* or *compressive autoencoding*, as illustrated in the diagram in Figure 1.1. This idea is highly compatible with the original goals laid out for autoencoders by [143, 105], which can be viewed as a generalization of the classic principal component analysis [127] for the case where the low-dimensional structure of $\boldsymbol{x}$ is linear.

Through tremendous empirical efforts over the last eleven years, it has become clear that deep networks are very effective in modeling nonlinear encoding and decoding mappings. A wide range of applications of deep learning, including those mentioned above, rely on realizing such an encoding or decoding scheme partially or entirely by learning f or g separately or together.

1.2 Challenges

Despite much empirical success of deep representation learning, they also pose new challenges and risks that are critical to address for their deployment in real-world high-stakes domains. Researchers have shown that (1) decisions made by deep models are difficult to interpret [167]; (2)

large-scale foundation models can memorize and leak private personal information [51]; (3) models can fail unpredictably under mild adversarial attacks or distribution shifts [240, 206]; (4) models are poorly calibrated, especially under complex environment [86, 191]. In this dissertation, we mainly focus on transparency and privacy.

Transparency and interpretability. Along the development of deep learning, many deep network architectures have been proposed and practiced for f or g, from the classic LeNet [150] to AlexNet [146], to ResNet [99] and then to the more recent transformer [250]. Despite their popularity, these networks have largely been designed empirically and trained and used as “black-box” function approximators. Many of the popular techniques and recipes for designing and training deep networks were developed through heuristic and empirical means, as opposed to rigorous mathematical principles, modeling and analysis. Practitioners constantly face a series of challenges for any new data and tasks: What architecture or particular components they should use for the network? How wide or deep the network should be? Which parts of the networks need to be trained and which can be determined in advance? Last but not the least, after the network has been trained, learned and tested: how to interpret functions of the operators; what are the roles and relationships among the multiple attention heads and multilayer perceptron (MLP) blocks in the transformer architecture [250]? As a result, desired properties of the learned feature representation $\boldsymbol{z}$ are not clearly specified or justified, and many heuristic measures or loss functions have been proposed and practiced for training task-agnostic representations with these models.

Privacy risks. Despite the widespread deployment of large-scale deep learning models, there are significant privacy and legal risks of training these models on sensitive data that often contain personal information or copyrighted material [51]. Studies have shown that generative foundation models such as GPT-3 can sometimes regurgitate memorized information about individuals and licensed content from its training data when prompted to do so [31]; [177] showed that non-generative vision SSL models can also be probed to reveal sensitive information about individual samples in its training data when given partial information; and [227, 30] demonstrated that one could prompt text-to-image diffusion models to generate near-perfect copy of certain training samples. Even in the scenario that the training data for these models are considered public in most cases, some of the data may be sensitive; additionally, there are certain privacy and copyright laws that apply to model training even on such public data [102]. Given these risks, there is an urgent need to train foundation models that can adhere to relevant privacy and copyright laws.

1.3 Overview

1.3.1 Part I: Transparent Representation Learning

The fundamental question of representation learning and a central problem that we will address in this part, is: *What is a principled and effective measure for the goodness of representations?* Conceptually, the quality of a representation $\boldsymbol{z}$ depends on how well it identifies the most relevant and sufficient information of $\boldsymbol{x}$ for subsequent tasks, and how efficiently it represents this information.

In Chapter 2, we propose the principle of Maximal Coding Rate Reduction (MCR^2), an information theoretic measure that maximizes the coding rate difference between the whole dataset and the sum of each individual class, to learn intrinsic low-dimensional structures from high-dimensional data that most discriminate between classes.

In Chapter 3, we introduce a generalized coding rate function and propose a unified objective function called sparse rate reduction, which can be used for measuring the quality of the learned representation in an unsupervised manner. Furthermore, we show that a transformer-like deep architecture can be derived from unrolling an alternating minimization scheme for the sparse rate reduction objective. This creates a mathematically interpretable, transformer-like architecture called CRATE (**C**oding **RATE** transformer), where each layer performs a single step of an alternating minimization algorithm to optimize the sparse rate reduction objective.

Chapter 2 is based on a joint work with Kwan Ho Ryan Chan, Chong You, Chaobing Song, and Yi Ma [290]. Chapter 3 is based on a joint work with Sam Buchanan, Druv Pai, Tianzhe Chu, Ziyang Wu, Shengbang Tong, Benjamin D. Haeffele, and Yi Ma [289], and also a joint work with Tianzhe Chu, Shengbang Tong, Ziyang Wu, Druv Pai, Sam Buchanan, and Yi Ma [291].

1.3.2 Part II: Privacy-preserving Representation Learning

In this section, we explore two strategies for mitigating privacy risks in representation learning: federated learning [176] and differential privacy (DP) [65]. Federated learning is an emerging paradigm for machine learning where multiple data holders (clients) collaborate to train a model on their combined dataset. Clients only share updated models and other statistics computed from their local dataset, which ensures that their raw data remains local and private. Differential privacy focuses on limiting the influence of individual training data points on the trained model, and hence has the potential to mitigate privacy risks for sensitive information that is confined to a single or a few training examples. In Chapter 4, we study the effectiveness of learned representations using federated optimization methods and propose an approach called Train-Convexify-Train (TCT) to overcome data heterogeneity when training deep, nonconvex models in a federated setting. In Chapter 5, we introduce a method to train foundational vision models, called ViP (Vision Transformer with Differential Privacy), that offers a DP guarantee and demonstrates the promise of high-utility differentially private representation learning.

Chapter 4 is based on a joint work with Alexander Wei, Sai Praneeth Karimireddy, Yi Ma, and Michael I. Jordan [293]. Chapter 5 is based on a joint work with Maziar Sanjabi, Yi Ma, Kamalika Chaudhuri, and Chuan Guo [292].

1.3.3 Connecting theory and practice.

A primary objective of this dissertation is to bridge the gap between theory and practice in deep representation learning. Theoretical frameworks could offer unified understandings and formal guarantees, thereby enhancing the reliability and interpretability of deep learning models. For instance, the rate reduction framework in Chapter 2 and Chapter 3 connects seemingly disparate approaches such as compressive encoding/decoding, rate reduction, and deep network architectures. On the other hand, as demonstrated in this dissertation, scaling up principled models and algorithms can lead to promising performance on large-scale, real-world tasks. This is an exciting time to connect theory and practice and build machine learning systems that are both reliable and trustworthy.

Part I

Transparent Representation Learning

Chapter 2

Representation Learning via Maximal Coding Rate Reduction

To learn intrinsic low-dimensional structures from high-dimensional data that most discriminate between classes, we propose the principle of *Maximal Coding Rate Reduction* (MCR^2), an information-theoretic measure that maximizes the coding rate difference between the whole dataset and the sum of each individual class. We clarify its relationships with most existing frameworks such as cross-entropy, information bottleneck, information gain, contractive and contrastive learning, and provide theoretical guarantees for learning diverse and discriminative features. The coding rate can be accurately computed from finite samples of degenerate subspace-like distributions and can learn intrinsic representations in supervised, self-supervised, and unsupervised settings in a unified manner. Empirically, the representations learned using this principle alone are significantly more robust to label corruptions in classification than those using cross-entropy, and can lead to state-of-the-art results in clustering mixed data from self-learned invariant features.

2.1 Context and Motivation

Given a random vector $\boldsymbol{x} \in \mathbb{R}^D$ which is drawn from a mixture of, say k, distributions $\mathcal{D} = \{\mathcal{D}_j\}_{j=1}^k$, one of the most fundamental problems in machine learning is how to effectively and efficiently *learn the distribution* from a finite set of i.i.d samples, say $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m] \in \mathbb{R}^{D\times m}$. To this end, we *seek a good representation* through a continuous mapping, $f(\boldsymbol{x}, \theta) : \mathbb{R}^D \to \mathbb{R}^d$, that captures intrinsic structures of $\boldsymbol{x}$ and best facilitates subsequent tasks such as classification or clustering.

Supervised learning of discriminative representations. To ease the task of learning $\mathcal{D}$, in the popular supervised setting, a true class label, represented as a one-hot vector $\boldsymbol{y}_i \in \mathbb{R}^k$, is given for each sample $\boldsymbol{x}_i$. Extensive studies have shown that for many practical datasets (images, audios, and natural languages, etc.), the mapping from the data $\boldsymbol{x}$ to its class label $\boldsymbol{y}$ can be effectively modeled by training a deep network [80], here denoted as $f(\boldsymbol{x}, \theta) : \boldsymbol{x} \mapsto \boldsymbol{y}$ with network parameters $\theta \in \Theta$. This is typically done by minimizing the *cross-entropy loss* over a training set $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^m$, through backpropagation over the network parameters θ:

$$\min_{\theta\in\Theta} \mathrm{CE}(\theta, \boldsymbol{x}, \boldsymbol{y}) \doteq -\mathbb{E}[\langle \boldsymbol{y}, \log[f(\boldsymbol{x}, \theta)]\rangle] \approx -\frac{1}{m}\sum_{i=1}^{m}\langle \boldsymbol{y}_i, \log[f(\boldsymbol{x}_i, \theta)]\rangle. \tag{2.1}$$

Despite its effectiveness and enormous popularity, there are two serious limitations with this approach: 1) It aims only to predict the labels $\boldsymbol{y}$ even if they might be mislabeled. Empirical studies show that deep networks, used as a "black box," can even fit random labels [300]. 2) With such an end-to-end data fitting, despite plenty of empirical efforts in trying to interpret the so-learned features [298], it is not clear to what extent the intermediate features learned by the network capture the intrinsic structures of the data that make meaningful classification possible in the first place. The precise geometric and statistical properties of the learned features are also often obscured, which leads to the lack of interpretability and subsequent performance guarantees (e.g., generalizability, transferability, and robustness, etc.) in deep learning. Therefore, the goal of this work is to address such limitations of current learning frameworks by reformulating the objective towards learning *explicitly meaningful* representations for the data $\boldsymbol{x}$.

Minimal discriminative features via information bottleneck. One popular approach to interpret the role of deep networks is to view outputs of intermediate layers of the network as selecting certain latent features $\boldsymbol{z} = f(\boldsymbol{x}, \theta) \in \mathbb{R}^d$ of the data that are discriminative among multiple classes. Learned representations $\boldsymbol{z}$ then facilitate the subsequent classification task for predicting the class label $\boldsymbol{y}$ by optimizing a classifier $g(\boldsymbol{z})$:

$$\boldsymbol{x} \xrightarrow{f(\boldsymbol{x},\theta)} \boldsymbol{z}(\theta) \xrightarrow{g(\boldsymbol{z})} \boldsymbol{y}.$$

The *information bottleneck* (IB) formulation [244] further hypothesizes that the role of the network is to learn $\boldsymbol{z}$ as the minimal sufficient statistics for predicting $\boldsymbol{y}$. Formally, it seeks to maximize the mutual information $I(\boldsymbol{z}, \boldsymbol{y})$ [49] between $\boldsymbol{z}$ and $\boldsymbol{y}$ while minimizing $I(\boldsymbol{x}, \boldsymbol{z})$ between $\boldsymbol{x}$ and $\boldsymbol{z}$:

$$\max_{\theta\in\Theta} \mathrm{IB}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}(\theta)) \doteq I(\boldsymbol{z}(\theta), \boldsymbol{y}) - \beta I(\boldsymbol{x}, \boldsymbol{z}(\theta)), \quad \beta > 0. \tag{2.2}$$

Given one can overcome some caveats associated with this framework [142], such as how to accurately evaluate mutual information with finitely samples of degenerate distributions, this framework has been successful in describing certain behaviors of deep networks. But by being task-dependent (depending on the label $\boldsymbol{y}$) and seeking a *minimal* set of most informative features for the task at hand (for predicting the label $\boldsymbol{y}$ only), the network sacrifices generalizability, robustness, or transferability, in case the labels can be corrupted or the learned features be tackled. To address this, our framework uses label $\boldsymbol{y}$ only as side information to assist learning *diverse* and *discriminative* representations, hence making learned features more robust to mislabeled data.

Contractive learning of generative representations. Complementary to the above supervised discriminative approach, *auto-encoding* [12, 143] is another popular *unsupervised* (label-free) framework used to learn good latent representations, which can be viewed as a nonlinear extension

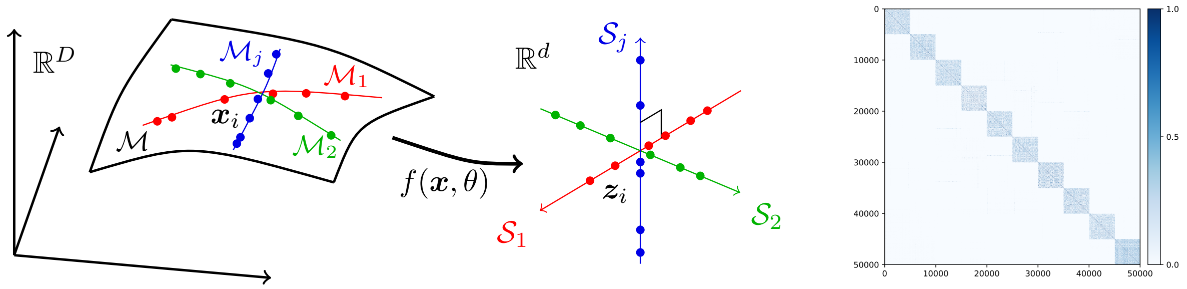


Figure 2.1: **Left and Middle:** The distribution $\mathcal{D}$ of high-dim data $\boldsymbol{x} \in \mathbb{R}^D$ is supported on a manifold $\mathcal{M}$ and its classes on low-dim submanifolds $\mathcal{M}_j$, we learn a map $f(\boldsymbol{x},\theta)$ such that $\boldsymbol{z}_i = f(\boldsymbol{x}_i,\theta)$ are on a union of maximally uncorrelated subspaces $\{\mathcal{S}_j\}$. **Right:** Cosine similarity between learned features by our method for the CIFAR10 training dataset. Each class has 5,000 samples and their features span a subspace of over 10 dimensions (see Figure 2.3c).

to the classical PCA [127]. The idea is to learn a compact latent representation $\boldsymbol{z} \in \mathbb{R}^d$ that adequately regenerates the original data $\boldsymbol{x}$ to certain extent, through optimizing decoder or generator $g(\boldsymbol{z},\eta)$:

$$\boldsymbol{x} \xrightarrow{f(\boldsymbol{x},\theta)} \boldsymbol{z}(\theta) \xrightarrow{g(\boldsymbol{z},\eta)} \widehat{\boldsymbol{x}}(\theta,\eta). \tag{2.3}$$

Typically, such representations are learned in an end-to-end fashion by imposing certain heuristics on geometric or statistical "compactness" of $\boldsymbol{z}$, such as its dimension, energy, or volume. For example, the *contractive* autoencoder [210] penalizes local volume expansion of learned features approximated by the Jacobian $\|\partial \boldsymbol{z}/\partial\theta\|$. Another key design factor of this approach is the choice of a proper, but often elusive, metric that can measure the desired *similarity* between $\boldsymbol{x}$ and the decoded $\widehat{\boldsymbol{x}}$, either between sample pairs $\boldsymbol{x}_i$ and $\widehat{\boldsymbol{x}}_i$ or between the two distributions $\mathcal{D}_{\boldsymbol{x}}$ and $\mathcal{D}_{\widehat{\boldsymbol{x}}}$. However, the distance between two distributions, say the KL divergence $\mathrm{KL}(\mathcal{D}_{\boldsymbol{x}}||\mathcal{D}_{\widehat{\boldsymbol{x}}})$, is very difficult to evaluate when the data distributions are discrete and degenerate. In practice, it can only be approximated with the help of an additional disriminative network, known as GAN [81, 8].

Representations learned through this framework can be arguably rich enough to regenerate the data to a certain extent. But depending on the choice of the regularizing heuristics on $\boldsymbol{z}$ and similarity metrics on $\boldsymbol{x}$ (or $\mathcal{D}_{\boldsymbol{x}}$), the objective is typically task-dependent and often grossly approximated [210, 81]. When the data contain complicated *multi-modal* structures, naive heuristics or inaccurate metrics may fail to capture all internal subclass structures or to explicitly discriminate among them for classification or clustering purposes. For example, one consequence of this is the phenomenon of *mode collapsing* in learning generative models for data that have mixed multi-modal structures [155]. To address this, we propose a principled measure (on $\boldsymbol{z}$) to learn representations that promotes multi-class discriminative property from data of mixed structures, which works in both supervised and unsupervised settings.

This work: Learning diverse and discriminative representations. Whether the given data $\boldsymbol{X}$ of a mixed distribution $\mathcal{D}$ can be effectively classified depends on how separable (or discriminative) the component distributions $\mathcal{D}_j$ are (or can be made). One popular working assumption is that the distribution of each class has relatively *low-dimensional* intrinsic structures. There are several reasons why this assumption is plausible: 1). High dimensional data are highly redundant; 2). Data that belong to the same class should be similar and correlated to each other; 3). Typically we only care about equivalent structures of $\boldsymbol{x}$ that are invariant to certain classes of deformation and augmentations. Hence we may assume the distribution $\mathcal{D}_j$ of each class has a support on a low-dimensional submanifold, say $\mathcal{M}_j$ with dimension $d_j \ll D$, and the distribution $\mathcal{D}$ of $\boldsymbol{x}$ is supported on the mixture of those submanifolds, $\mathcal{M} = \cup_{j=1}^{k}\mathcal{M}_j$, in the high-dimensional ambient space $\mathbb{R}^D$, as illustrated in Figure 2.1 left.

With the manifold assumption in mind, we want to learn a mapping $\boldsymbol{z} = f(\boldsymbol{x}, \theta)$ that maps each of the submanifolds $\mathcal{M}_j \subset \mathbb{R}^D$ to a *linear* subspace $\mathcal{S}_j \subset \mathbb{R}^d$ (see Figure 2.1 middle). To do so, we require our learned representation to have the following properties:

1. *Between-Class Discriminative:* Features of samples from different classes should be highly *uncorrelated* and belong to different low-dimensional linear subspaces.

2. *Within-Class Compressible:* Features of samples from the same class should be relatively *correlated* in a sense that they belong to a low-dimensional linear subspace.

3. *Maximally Diverse Representation:* Dimension (or variance) of features for each class should be *as large as possible* as long as they stay uncorrelated from the other classes.

Notice that, although the intrinsic structures of each class/cluster may be low-dimensional, they are by no means simply linear in their original representation $\boldsymbol{x}$. Here the subspaces $\{\mathcal{S}_j\}$ can be viewed as nonlinear *generalized principal components* for $\boldsymbol{x}$ [253]. Furthermore, for many clustering or classification tasks (such as object recognition), we consider two samples as *equivalent* if they differ by certain class of domain deformations or augmentations $\mathcal{T} = \{\tau\}$. Hence, we are only interested in low-dimensional structures that are *invariant* to such deformations (i.e., $\boldsymbol{x} \in \mathcal{M}$ iff $\tau(\boldsymbol{x}) \in \mathcal{M}$ for all $\tau \in \mathcal{T}$), which are known to have sophisticated geometric and topological structures [256] and can be difficult to learn in a principled manner even with CNNs [46, 47].

There are previous attempts to directly enforce subspace structures on features learned by a deep network for supervised [153] or unsupervised learning [124, 302, 197, 306, 303, 301, 153]. However, the *self-expressive* property of subspaces exploited by [124] does not enforce all the desired properties listed above [90]; [153] uses a nuclear norm based geometric loss to enforce orthogonality between classes, but does not promote diversity in the learned representations, as we will soon see. Figure 2.1 right illustrates a representation learned by our method on the CIFAR10 dataset. More details can be found in the experimental Section 2.3.

2.2 Technical Approach and Method

2.2.1 Measure of Compactness for a Representation

Although the above properties are all highly desirable for the latent representation $\boldsymbol{z}$, they are by no means easy to obtain: Are these properties compatible so that we can expect to achieve them all at once? If so, is there a *simple but principled* objective that can measure the goodness of the resulting representations in terms of all these properties? The key to these questions is to find a principled "measure of compactness" for the distribution of a random variable $\boldsymbol{z}$ or from its finite samples $\boldsymbol{Z}$. Such a measure should directly and accurately characterize intrinsic geometric or statistical properties of the distribution, in terms of its intrinsic dimension or volume. Unlike cross-entropy (2.1) or information bottleneck (2.2), such a measure should not depend explicitly on class labels so that it can work in all supervised, self-supervised, semi-supervised, and unsupervised settings.

Low-dimensional degenerate distributions. In information theory [49], the notion of entropy $H(\boldsymbol{z})$ is designed to be such a measure. However, entropy is not well-defined for continuous random variables with degenerate distributions. The same difficulty resides with evaluating mutual information $I(\boldsymbol{x}, \boldsymbol{z})$ for degenerate distributions. This is unfortunately the case here. To alleviate this difficulty, another related concept in information theory, more specifically in lossy data compression, that measures the "compactness" of a random distribution is the so-called *rate distortion* [49]: Given a random variable $\boldsymbol{z}$ and a prescribed precision $\epsilon > 0$, the rate distortion $R(\boldsymbol{z}, \epsilon)$ is the minimal number of binary bits needed to encode $\boldsymbol{z}$ such that the expected decoding error is less than ϵ, i.e., the decoded $\widehat{\boldsymbol{z}}$ satisfies $\mathbb{E}[\|\boldsymbol{z} - \widehat{\boldsymbol{z}}\|_2] \leq \epsilon$. Although this framework has been successful in explaining feature selection in deep networks [174], the rate distortion of a random variable is difficult, if not impossible to compute, except for simple distributions such as discrete and Gaussian.

Nonasymptotic rate distortion for finite samples. When evaluating the lossy coding rate R, one practical difficulty is that we normally do not know the distribution of $\boldsymbol{z}$. Instead, we have a finite number of samples as learned representations where $\boldsymbol{z}_i = f(\boldsymbol{x}_i, \theta) \in \mathbb{R}^d, i = 1, \dots, m$, for the given data samples $\boldsymbol{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_m]$. Fortunately, [172] provides a precise estimate on the number of binary bits needed to encoded finite samples from a subspace-like distribution. In order to encode the learned representation $\boldsymbol{Z} = [\boldsymbol{z}_1, \dots, \boldsymbol{z}_m]$ up to a precision ϵ, the total number of bits needed is given by the following expression: $\mathcal{L}(\boldsymbol{Z}, \epsilon) \doteq \left(\frac{m+d}{2}\right) \log\det\left(\boldsymbol{I} + \frac{d}{m\epsilon^2}\boldsymbol{Z}\boldsymbol{Z}^\top\right)$. This formula can be derived either by packing ϵ-balls into the space spanned by $\boldsymbol{Z}$ or by computing the number of bits needed to quantize the SVD of $\boldsymbol{Z}$ subject to the precision, see [172] for proofs. Therefore, the compactness of learned features *as a whole* can be measured in terms of the average coding length per sample (as the sample size m is large), a.k.a. the *coding rate* subject to the distortion ϵ:

$$R(\boldsymbol{Z}, \epsilon) \doteq \frac{1}{2} \log\det\left(\boldsymbol{I} + \frac{d}{m\epsilon^2}\boldsymbol{Z}\boldsymbol{Z}^\top\right). \tag{2.4}$$

Rate distortion of data with a mixed distribution. In general, the features $\boldsymbol{Z}$ of multi-class data may belong to multiple low-dimensional subspaces. To evaluate the rate distortion of such mixed data *more accurately*, we may partition the data $\boldsymbol{Z}$ into multiple subsets: $\boldsymbol{Z} = \boldsymbol{Z}_1 \cup \dots \cup \boldsymbol{Z}_k$, with each in one low-dim subspace. So the above coding rate (2.4) is accurate for each subset. For convenience, let $\boldsymbol{\Pi} = \{\boldsymbol{\Pi}_j \in \mathbb{R}^{m \times m}\}_{j=1}^k$ be a set of diagonal matrices whose diagonal entries encode the membership of the m samples in the k classes. More specifically, the diagonal entry $\boldsymbol{\Pi}_j(i,i)$ of $\boldsymbol{\Pi}_j$ indicates the probability of sample i belonging to subset j. Therefore $\boldsymbol{\Pi}$ lies in a simplex: $\Omega \doteq \{\boldsymbol{\Pi} \mid \boldsymbol{\Pi}_j \geq \boldsymbol{0},\ \boldsymbol{\Pi}_1 + \dots + \boldsymbol{\Pi}_k = \boldsymbol{I}\}$. Then, according to [172], with respect to this partition, the average number of bits per sample (the coding rate) is

$$R^c(\boldsymbol{Z}, \epsilon \mid \boldsymbol{\Pi}) \doteq \sum_{j=1}^{k} \frac{\operatorname{tr}(\boldsymbol{\Pi}_j)}{2m} \log\det\left(\boldsymbol{I} + \frac{d}{\operatorname{tr}(\boldsymbol{\Pi}_j)\epsilon^2} \boldsymbol{Z}\boldsymbol{\Pi}_j\boldsymbol{Z}^\top\right). \tag{2.5}$$

When $\boldsymbol{Z}$ is given, $R^c(\boldsymbol{Z}, \epsilon \mid \boldsymbol{\Pi})$ is a concave function of $\boldsymbol{\Pi}$. The function $\log\det(\cdot)$ in the above expressions has been long known as an effective heuristic for rank minimization problems, with guaranteed convergence to local minimum [70]. As it nicely characterizes the rate distortion of Gaussian or subspace-like distributions, $\log\det(\cdot)$ can be very effective in clustering or classification of mixed data [172, 271, 131].

2.2.2 Principle of Maximal Coding Rate Reduction

On one hand, for learned features to be discriminative, features of different classes/clusters are preferred to be *maximally incoherent* to each other. Hence they together should span a space of the largest possible volume (or dimension) and the coding rate of the whole set $\boldsymbol{Z}$ should be as large as possible. On the other hand, learned features of the same class/cluster should be highly correlated and coherent. Hence, each class/cluster should only span a space (or subspace) of a very small volume and the coding rate should be as small as possible. Therefore, a good representation $\boldsymbol{Z}$ of $\boldsymbol{X}$ is one such that, given a partition $\boldsymbol{\Pi}$ of $\boldsymbol{Z}$, achieves a large difference between the coding rate for the whole and that for all the subsets:

$$\Delta R(\boldsymbol{Z}, \boldsymbol{\Pi}, \epsilon) \doteq R(\boldsymbol{Z}, \epsilon) - R^c(\boldsymbol{Z}, \epsilon \mid \boldsymbol{\Pi}). \tag{2.6}$$

If we choose our feature mapping $\boldsymbol{z} = f(\boldsymbol{x}, \theta)$ to be a deep neural network, the overall process of the feature representation and the resulting rate reduction w.r.t. certain partition $\boldsymbol{\Pi}$ can be illustrated by the following diagram:

$$\boldsymbol{X} \xrightarrow{f(\boldsymbol{x},\theta)} \boldsymbol{Z}(\theta) \xrightarrow{\boldsymbol{\Pi},\epsilon} \Delta R(\boldsymbol{Z}(\theta), \boldsymbol{\Pi}, \epsilon). \tag{2.7}$$

Note that ΔR is *monotonic* in the scale of the features $\boldsymbol{Z}$. So to make the amount of reduction comparable between different representations, we need to *normalize the scale* of the learned features, either by imposing the Frobenius norm of each class $\boldsymbol{Z}_j$ to scale with the number of features in $\boldsymbol{Z}_j \in \mathbb{R}^{d\times m_j}$: $\|\boldsymbol{Z}_j\|_F^2 = m_j$ or by normalizing each feature to be on the unit sphere: $\boldsymbol{z}_i \in \mathbb{S}^{d-1}$. This formulation offers a natural justification for the need of "batch normalization" in the practice of training deep neural networks [120]. An alternative, arguably simpler, way to normalize the scale of learned representations is to ensure that the mapping of each layer of the network is approximately *isometric* [199].

Once the representations are comparable, our goal becomes to learn a set of features $\boldsymbol{Z}(\theta) = f(\boldsymbol{X}, \theta)$ and their partition $\boldsymbol{\Pi}$ (if not given in advance) such that they maximize the reduction between the coding rate of all features and that of the sum of features w.r.t. their classes:

$$\max_{\theta,\boldsymbol{\Pi}} \Delta R\big(\boldsymbol{Z}(\theta), \boldsymbol{\Pi}, \epsilon\big) = R(\boldsymbol{Z}(\theta), \epsilon) - R^c(\boldsymbol{Z}(\theta), \epsilon \mid \boldsymbol{\Pi}), \quad \text{s.t.} \ \ \|\boldsymbol{Z}_j(\theta)\|_F^2 = m_j, \, \boldsymbol{\Pi} \in \Omega. \tag{2.8}$$

We refer to this as the principle of *maximal coding rate reduction* (MCR^2), an embodiment of Aristotle's famous quote: "*the whole is greater than the sum of the parts.*" Note that for the clustering purpose alone, one may only care about the sign of ΔR for deciding whether to partition the data or not, which leads to the greedy algorithm in [172]. More specifically, in the context of clustering *finite* samples, one needs to use the more precise measure of the coding length mentioned earlier, see [172] for more details. Here to seek or learn the best representation, we further desire the whole is *maximally* greater than its parts.

Relationship to information gain. The maximal coding rate reduction can be viewed as a generalization to *Information Gain* (IG), which aims to maximize the reduction of entropy of a random variable, say $\boldsymbol{z}$, with respect to an observed attribute, say $\boldsymbol{\pi}$: $\max_{\boldsymbol{\pi}}\ \mathrm{IG}(\boldsymbol{z}, \boldsymbol{\pi}) \doteq H(\boldsymbol{z}) - H(\boldsymbol{z} \mid \boldsymbol{\pi})$, i.e., the *mutual information* between $\boldsymbol{z}$ and $\boldsymbol{\pi}$ [49]. Maximal information gain has been widely used in areas such as decision trees [200]. However, MCR^2 is used differently in several ways: 1) One typical setting of MCR^2 is when the data class labels are given, i.e. $\boldsymbol{\Pi}$ is known, MCR^2 focuses on learning representations $\boldsymbol{z}(\theta)$ rather than fitting labels. 2) In traditional settings of IG, the number of attributes in $\boldsymbol{z}$ cannot be so large and their values are discrete (typically binary). Here the "attributes" $\boldsymbol{\Pi}$ represent the probability of a multi-class partition for all samples and their values can even be continuous. 3) As mentioned before, entropy $H(\boldsymbol{z})$ or mutual information $I(\boldsymbol{z}, \boldsymbol{\pi})$ [107] is not well-defined for degenerate continuous distributions whereas the rate distortion $R(\boldsymbol{z}, \epsilon)$ is and can be accurately and efficiently computed for (mixed) subspaces, at least.

2.2.3 Properties of the Rate Reduction Function

In theory, the MCR^2 principle (2.8) benefits from great generalizability and can be applied to representations $\boldsymbol{Z}$ of *any* distributions with *any* attributes $\boldsymbol{\Pi}$ as long as the rates R and R^c for the distributions can be accurately and efficiently evaluated. The optimal representation $\boldsymbol{Z}^*$ and partition $\boldsymbol{\Pi}^*$ should have some interesting geometric and statistical properties. We here reveal nice properties of the optimal representation with the special case of subspaces, which have many important use cases in machine learning. When the desired representation for $\boldsymbol{Z}$ is multiple subspaces, the rates R and R^c in (2.8) are given by (2.4) and (2.5), respectively. At the maximal rate reduction, MCR^2 achieves its optimal representations, denoted as $\boldsymbol{Z}^* = \boldsymbol{Z}_1^* \cup \cdots \cup \boldsymbol{Z}_k^* \subset \mathbb{R}^d$ with $\mathsf{rank}(\boldsymbol{Z}_j^*) \leq d_j$. One can show that $\boldsymbol{Z}^*$ has the following desired properties (see Appendix A for a formal statement and detailed proofs).

Theorem 2.2.1 (Informal Statement). *Suppose $\boldsymbol{Z}^* = \boldsymbol{Z}_1^* \cup \cdots \cup \boldsymbol{Z}_k^*$ is the optimal solution that maximizes the rate reduction* (2.8). *We have:*

- Between-class Discriminative*: As long as the ambient space is adequately large ($d \geq \sum_{j=1}^{k} d_j$), the subspaces are all orthogonal to each other,* i.e. $(\boldsymbol{Z}_i^*)^\top \boldsymbol{Z}_j^* = \boldsymbol{0}$ *for $i \neq j$.*

- Maximally Diverse Representation*: As long as the coding precision is adequately high, i.e., $\epsilon^4 < c \cdot \min_j \left\{\frac{m_j}{m}\frac{d^2}{d_j^2}\right\}$, where $c > 0$ is a constant. Each subspace achieves its maximal dimension, i.e. $\mathsf{rank}(\boldsymbol{Z}_j^*) = d_j$. In addition, the largest $d_j - 1$ singular values of $\boldsymbol{Z}_j^*$ are equal.*

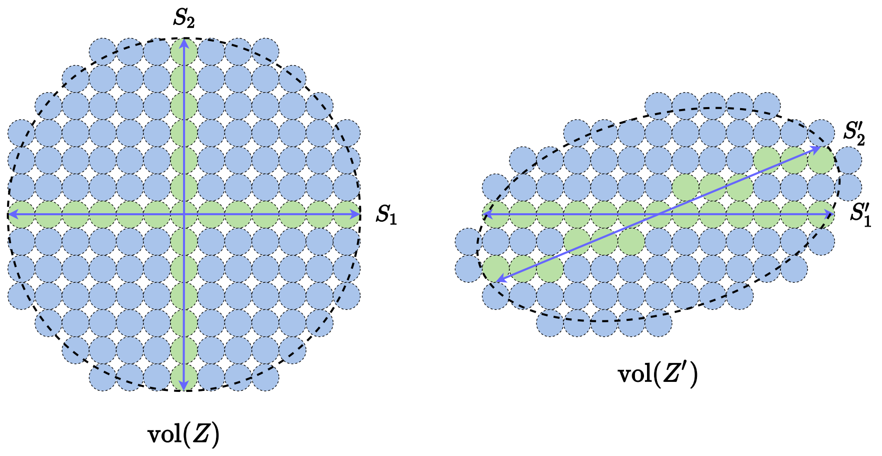


Figure 2.2: Comparison of two learned representations $\boldsymbol{Z}$ and $\boldsymbol{Z}'$ via reduced rates: R is the number of ϵ-balls packed in the joint distribution and R^c is the sum of the numbers for all the subspaces (the green balls). ΔR is their difference (the number of blue balls). The MCR^2 principle prefers $\boldsymbol{Z}$ (the left one).

In other words, in the case of subspaces, the MCR^2 principle promotes embedding of data into multiple independent subspaces, with features distributed *isotropically* in each subspace (except for possibly one dimension). In addition, among all such discriminative representations, it prefers the one with the highest dimensions in the ambient space. This is substantially different from the objective of information bottleneck (2.2).

Comparison to the geometric OLE loss. To encourage the learned features to be uncorrelated between classes, the work of [153] has proposed to maximize the difference between the nuclear norm of the whole $\boldsymbol{Z}$ and its subsets $\boldsymbol{Z}_j$, called the *orthogonal low-rank embedding* (OLE) loss: $\max_\theta \text{OLE}(\boldsymbol{Z}(\theta), \boldsymbol{\Pi}) \doteq \|\boldsymbol{Z}(\theta)\|_* - \sum_{j=1}^k \|\boldsymbol{Z}_j(\theta)\|_*$, added as a regularizer to the cross-entropy loss (2.1). The nuclear norm $\|\cdot\|_*$ is a *nonsmooth convex* surrogate for low-rankness and the non-smoothness potentially poses additional difficulties in using this loss to learn features via gradient descent, whereas $\log\det(\cdot)$ is *smooth concave* instead. Unlike the rate reduction ΔR, OLE is always *negative* and achieves the maximal value 0 when the subspaces are orthogonal, regardless of their dimensions. So in contrast to ΔR, this loss serves as a geometric heuristic and does not promote diverse representations. In fact, OLE typically promotes learning one-dim representations per class, whereas MCR^2 encourages learning subspaces with maximal dimensions (Figure 7 of [153] versus our Figure A.1).

Relation to contrastive learning. If samples are *evenly* drawn from k classes, a randomly chosen pair $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is of high probability belonging to difference classes if k is large. For example, when $k \geq 100$, a random pair is of probability 99% belonging to different classes. We may view the learned features of two samples together with their augmentations $\boldsymbol{Z}_i$ and $\boldsymbol{Z}_j$ as two classes. Then the rate reduction $\Delta R_{ij} = R(\boldsymbol{Z}_i \cup \boldsymbol{Z}_j, \epsilon) - \frac{1}{2}(R(\boldsymbol{Z}_i, \epsilon) + R(\boldsymbol{Z}_j, \epsilon))$ gives a "distance" measure for how far the two sample sets are. We may try to further "expand" pairs that likely belong to different classes. From Theorem 2.2.1, the (averaged) rate reduction ΔR_{ij} is maximized when features from different samples are uncorrelated $\boldsymbol{Z}_i^\top \boldsymbol{Z}_j = \boldsymbol{0}$ (see Figure 2.2) and features $\boldsymbol{Z}_i$ from the same sample are highly correlated. Hence, when applied to sample pairs, MCR^2 naturally conducts the so-called *contrastive learning* [89, 189, 94]. But MCR^2 is *not* limited to expand (or compress) pairs of samples and can uniformly conduct "contrastive learning" for a subset with *any number* of samples as long as we know they likely belong to different (or the same) classes, say by randomly sampling subsets from a large number of classes or with a good clustering method.

2.3 Experiments with Instantiations of MCR^2

Our theoretical analysis above shows how the *maximal coding rate reduction* (MCR^2) is a principled measure for learning discriminative and diverse representations for mixed data. In this section, we demonstrate experimentally how this principle alone, *without any other heuristics,* is adequate to learning good representations in the supervised, self-supervised, and unsupervised learning settings in a unified fashion. Our goal here is to validate effectiveness of this principle through its most basic usage and fair comparison with existing frameworks. More implementation details and experiments are given in Appendix A.2. The code can be found in `https://github.com/ryanchankh/mcr2`.

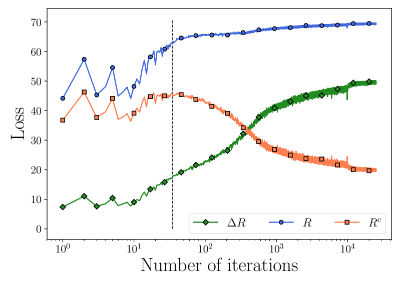
2.3.1 Supervised Learning of Robust Discriminative Features

Supervised learning via rate reduction. When class labels are provided during training, we assign the membership (diagonal) matrix $\mathbf{\Pi} = \{\mathbf{\Pi}_j\}_{j=1}^k$ as follows: for each sample $\boldsymbol{x}_i$ with label j, set $\mathbf{\Pi}_j(i,i) = 1$ and $\mathbf{\Pi}_l(i,i) = 0, \forall l \neq j$. Then the mapping $f(\cdot, \theta)$ can be learned by optimizing (2.8), where $\mathbf{\Pi}$ remains constant. We apply stochastic gradient descent to optimize MCR^2, and for each iteration we use mini-batch data $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^m$ to approximate the MCR^2 loss.

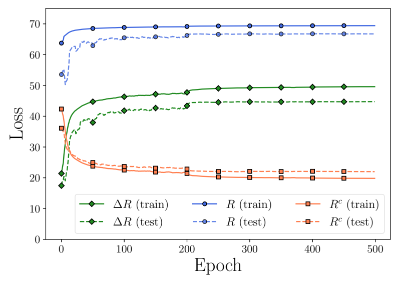
Evaluation via classification. As we will see, in the supervised setting, the learned representation has very clear subspace structures. So to evaluate the learned representations, we consider a natural nearest subspace classifier. For each class of learned features $\boldsymbol{Z}_j$, let $\boldsymbol{\mu}_j \in \mathbb{R}^p$ be its mean and $\boldsymbol{U}_j \in \mathbb{R}^{p \times r_j}$ be the first r_j principal components for $\boldsymbol{Z}_j$, where r_j is the estimated dimension of class j. The predicted label of a test data $\boldsymbol{x}'$ is given by $j' = \arg\min_{j \in \{1,\dots,k\}} \|(\boldsymbol{I} - \boldsymbol{U}_j\boldsymbol{U}_j^\top)(f(\boldsymbol{x}', \theta) - \boldsymbol{\mu}_j)\|_2^2$.

Experiments on real data. We consider CIFAR10 dataset [144] and ResNet-18 [100] for $f(\cdot, \theta)$. We replace the last linear layer of ResNet-18 by a two-layer fully connected network with ReLU activation function such that the output dimension is 128. We set the mini-batch size as $m = 1{,}000$ and the precision parameter $\epsilon^2 = 0.5$. More results can be found in Appendix A.2.3.2.

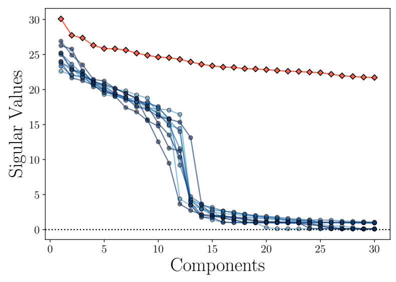
Figure 2.3a illustrates how the two rates and their difference (for both training and test data) evolves over epochs of training: After an initial phase, R gradually increases while R^c decreases, indicating that features $\boldsymbol{Z}$ are expanding as a whole while each class $\boldsymbol{Z}_j$ is being compressed. Figure 2.3c shows the distribution of singular values per $\boldsymbol{Z}_j$ and Figure 2.1 (right) shows the angles of features sorted by class. Compared to the geometric loss [153], our features are *not only orthogonal but also of much higher dimension.* We compare the singular values of representations, both overall data and individual classes, learned by using cross-entropy and MCR^2 in Figure A.1



(a) Evolution of R, R^c, ΔR during the training process.

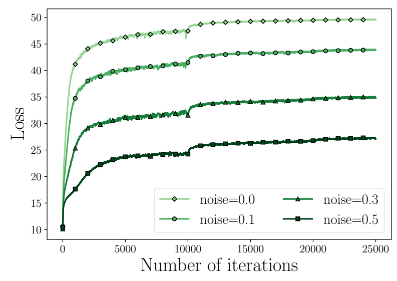


(b) Training loss versus testing loss.

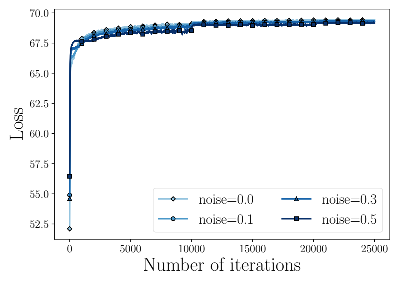


(c) PCA: (**red**) overall data; (**blue**) individual classes.

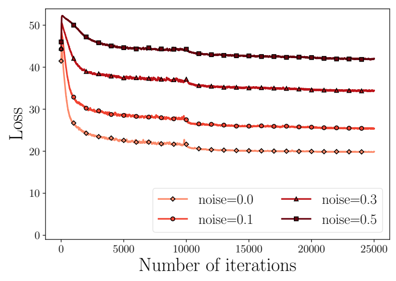
Figure 2.3: Evolution of the rates of MCR^2 in the training process and principal components of learned features.



(a) $\Delta R\big(\boldsymbol{Z}(\theta), \boldsymbol{\Pi}, \epsilon\big)$.



(b) $R(\boldsymbol{Z}(\theta), \epsilon)$.



(c) $R^c(\boldsymbol{Z}(\theta), \epsilon \mid \boldsymbol{\Pi})$.

Figure 2.4: Evolution of rates $R, R^c, \Delta R$ of MCR^2 during training with corrupted labels.

and Figure A.2 in Appendix A.2.3.1. We find that the representations learned by using MCR^2 loss are much more diverse than the ones learned by using cross-entropy loss. In addition, we find that we are able to select diverse images from the same class according to the "principal" components of the learned features (see Figure A.3 and Figure A.4 in Appendix A.2.3.1).

Robustness to corrupted labels. Because MCR^2 by design encourages richer representations that preserves intrinsic structures from the data $\boldsymbol{X}$, training relies less on class labels than traditional loss such as cross-entropy (CE). To verify this, we train the same network using both CE and MCR^2 with certain ratios of *randomly corrupted* training labels. Figure 2.4 illustrates the learning process: for different levels of corruption, while the rate for the whole set always converges to the same value, the rates for the classes are inversely proportional to the ratio of corruption, indicating our method only compresses samples with valid labels. The classification results are summarized in Table 2.1. By applying *exact the same* training parameters, MCR^2 is significantly more robust than CE, especially with higher ratio of corrupted labels. This can be an advantage in the settings of self-supervised learning or constrastive learning when the grouping information can be very noisy. More detailed comparison between MCR^2 and OLE [153], Large Margin Deep Networks [67], and ITLM [220] on learning from noisy labels can be found in Appendix A.2.4 (Table A.5).

Table 2.1: Classification results with features learned with labels corrupted at different levels.

	Ratio=0.1	Ratio=0.2	Ratio=0.3	Ratio=0.4	Ratio=0.5
CE Training	90.91%	86.12%	79.15%	72.45%	60.37%
MCR^2 Training	**91.16%**	**89.70%**	**88.18%**	**86.66%**	**84.30%**

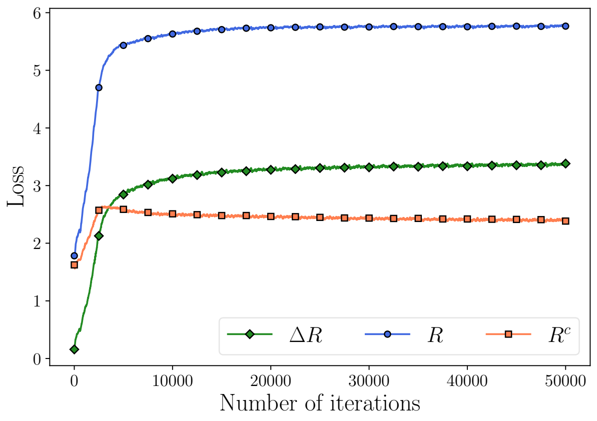
2.3.2 Self-supervised Learning of Invariant Features

Learning invariant features via rate reduction. Motivated by self-supervised learning algorithms [151, 138, 189, 94, 276], we use the MCR^2 principle to learn representations that are *invariant* to certain class of transformations/augmentations, say $\mathcal{T}$ with a distribution $P_{\mathcal{T}}$. Given a mini-batch of data $\{\boldsymbol{x}_j\}_{j=1}^{k}$, we augment each sample $\boldsymbol{x}_j$ with n transformations/augmentations $\{\tau_i(\cdot)\}_{i=1}^{n}$ randomly drawn from $P_{\mathcal{T}}$. We simply label all the augmented samples $\boldsymbol{X}_j = [\tau_1(\boldsymbol{x}_j), \dots, \tau_n(\boldsymbol{x}_j)]$ of $\boldsymbol{x}_j$ as the j-th class, and $\boldsymbol{Z}_j$ the corresponding learned features. Using this self-labeled data, we train our feature mapping $f(\cdot, \theta)$ the same way as the supervised setting above. For every mini-batch, the total number of samples for training is $m = kn$.

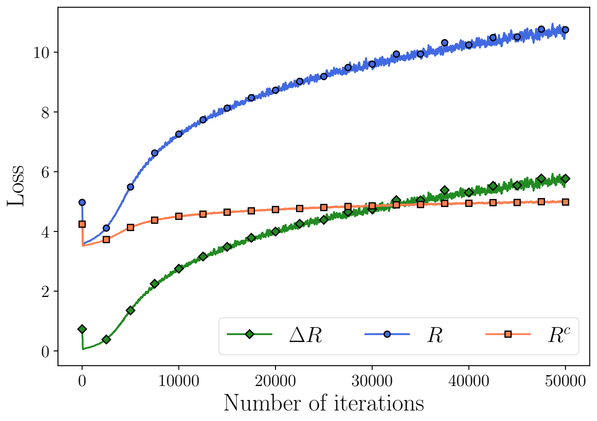
Evaluation via clustering. To learn invariant features, our formulation itself does *not* require the original samples $\boldsymbol{x}_j$ come from a fixed number of classes. For evaluation, we may train on a few classes and observe how the learned features facilitate classification or clustering of the data. A common method to evaluate learned features is to train an additional linear classifier [189, 94], with ground truth labels. But for our purpose, because we explicitly verify whether the so-learned invariant features have good subspace structures when the samples come from k classes, we use an off-the-shelf subspace clustering algorithm EnSC [284], which is computationally efficient and is provably correct for data with well-structured subspaces. We also use K-Means on the original data $\boldsymbol{X}$ as our baseline for comparison. We use normalized mutual information (NMI), clustering accuracy (ACC), and adjusted rand index (ARI) for our evaluation metrics, see Appendix A.2.4.2 for their detailed definitions.

Controlling dynamics of expansion and compression. By directly optimizing the rate reduction $\Delta R = R - R^c$, we achieve 0.570 clustering accuracy on CIFAR10 dataset, which is the second best result compared with previous methods. More details can be found in Appendix A.2.4.1. Empirically, we observe that, without class labels, the overall *coding rate* R expands quickly and the MCR^2 loss saturates (at a local maximum), see Fig 2.5a. Our experience suggests that learning a good representation from unlabeled data might be too ambitious when directly optimizing the original ΔR. Nonetheless, from the geometric meaning of R and R^c, one can design a different learning strategy by controlling the dynamics of expansion and compression differently during training. For instance, we may re-scale the rate by replacing $R(\boldsymbol{Z}, \epsilon)$ with $\widetilde{R}(\boldsymbol{Z}, \epsilon) \doteq \frac{1}{2\gamma_1} \log\det(\boldsymbol{I} + \frac{\gamma_2 d}{m\epsilon^2}\boldsymbol{Z}\boldsymbol{Z}^\top)$. With $\gamma_1 = \gamma_2 = k$, the learning dynamics change from Fig 2.5a to Fig 2.5b: All features are first compressed then gradually expand. We denote the controlled MCR^2 training by MCR^2-CTRL.

Experiments on real data. Similar to the supervised learning setting, we train *exactly the same* ResNet-18 network on the CIFAR10, CIFAR100, and STL10 [45] datasets. We set the mini-batch



(a) MCR^2.



(b) MCR^2-CTRL.

Figure 2.5: Evolution of the rates of (**left**) MCR^2 and (**right**) MCR^2-CTRL in the training process in the self-supervised setting on CIFAR10 dataset.

Table 2.2: Clustering results on CIFAR10, CIFAR100, and STL10 datasets.

Dataset	Metric	K-Means	JULE	RTM	DEC	DAC	DCCM	MCR^2-Ctrl
CIFAR10	NMI	0.087	0.192	0.197	0.257	0.395	0.496	**0.630**
	ACC	0.229	0.272	0.309	0.301	0.521	0.623	**0.684**
	ARI	0.049	0.138	0.115	0.161	0.305	0.408	**0.508**
CIFAR100	NMI	0.084	0.103	-	0.136	0.185	0.285	**0.387**
	ACC	0.130	0.137	-	0.185	0.237	0.327	**0.375**
	ARI	0.028	0.033	-	0.050	0.087	0.173	**0.178**
STL10	NMI	0.124	0.182	-	0.276	0.365	0.376	**0.446**
	ACC	0.192	0.182	-	0.359	0.470	0.482	**0.491**
	ARI	0.061	0.164	-	0.186	0.256	0.262	**0.290**

size as $k = 20$, number of augmentations for each sample as $n = 50$ and the precision parameter as $\epsilon^2 = 0.5$. Table 2.2 shows the results of the proposed MCR^2-CTRL in comparison with methods JULE [282], RTM [186], DEC [279], DAC [34], and DCCM [272] that have achieved the best results on these datasets. Surprisingly, without utilizing any inter-class or inter-sample information and heuristics on the data, the invariant features learned by our method with augmentations alone achieves a better performance over other highly engineered clustering methods. More comparisons and ablation studies can be found in Appendix A.2.4.2.

Nevertheless, compared to the representations learned in the supervised setting where the optimal partition $\boldsymbol{\Pi}$ in (2.8) is initialized by correct class information, the representations here learned with self-supervised classes are far from being optimal. It remains wide open how to design better optimization strategies and dynamics to learn from unlabelled or partially-labelled data better representations (and the associated partitions) close to the global maxima of the MCR^2 objective.

2.4 Conclusion

This work provides rigorous theoretical justifications and clear empirical evidences for why the maximal coding rate reduction (MCR^2) is a fundamental principle for learning discriminative low-dim representations in almost all learning settings. It unifies and explains existing effective frameworks and heuristics widely practiced in the (deep) learning literature. It remains open *why* MCR^2 is robust to label noises in the supervised setting, *why* self-learned features with MCR^2 alone are effective for clustering, and *how* in future practice instantiations of this principle can be systematically harnessed to further improve clustering or classification tasks.

We believe that MCR^2 gives a principled and practical objective for (deep) learning and can potentially lead to better design operators and architectures of a deep network. A potential direction is to monitor quantitatively the amount of rate reduction ΔR gained through every layer of the deep network. By optimizing the rate reduction through the network layers, it is no longer engineered as a "black box."

On the learning theoretical aspect, although this work has demonstrated only with mixed subspaces, this principle applies to any mixed distributions or structures, for which configurations that achieve maximal rate reduction are of independent theoretical interest. Another interesting note is that the MCR^2 formulation goes beyond the supervised multi-class learning setting often studied through empirical risk minimization (ERM) [53]. It is more related to the expectation maximization (EMX) framework [18], in which the notion of "compression" plays a crucial role for purely theoretical analysis. We hope this work provides a good connection between machine learning theory and its practice.

Chapter 3

Interpretable White-Box Transformers via Sparse Rate Reduction

In this work, we contend that the objective of representation learning is to compress and transform the distribution of the data, say sets of tokens, towards a mixture of low-dimensional Gaussian distributions supported on incoherent subspaces. The quality of the final representation can be measured by a unified objective function called *sparse rate reduction*. From this perspective, popular deep networks such as transformers can be naturally viewed as realizing iterative schemes to optimize this objective incrementally. Particularly, we show that the standard transformer block can be derived from alternating optimization on complementary parts of this objective: the multi-head self-attention operator can be viewed as a gradient descent step to compress the token sets by minimizing their lossy coding rate, and the subsequent multi-layer perceptron can be viewed as attempting to sparsify the representation of the tokens. This leads to a family of *white-box* transformer-like deep network architectures which are mathematically fully interpretable. Despite their simplicity, experiments show that these networks indeed learn to optimize the designed objective: they compress and sparsify representations of large-scale real-world vision datasets such as ImageNet, and achieve performance very close to thoroughly engineered transformers such as ViT. Code is at `https://github.com/Ma-Lab-Berkeley/CRATE`.

3.1 Introduction

In recent years, deep learning has seen tremendous empirical success in processing massive amounts of high-dimensional and multi-modal data. Much of this success is owed to effective learning of the data distribution and then transforming the distribution to a parsimonious, i.e. *structured and compact*, representation [202, 40, 92, 173], which facilitates many downstream tasks (e.g., in vision, classification [99, 63], recognition and segmentation [29, 97, 140], and generation [136, 212, 214]). To this end, many models and methods have been proposed and practiced, each with its own strengths and limitations. Here, we give several popular methods a brief accounting as context for a complete understanding and unification that we seek in this work.

Transformer models and self-attention. Transformers [250] are one of the latest popular models for learning a representation for high-dimensional structured data, such as text [250, 61, 26], images [63, 58], and other types of signals [79, 9]. After the first block, which converts each data point (such as a text corpus or image) into a set or sequence of *tokens*, further processing is performed on the token sets, in a medium-agnostic manner [250, 63]. A cornerstone of the transformer model is the so-called *self-attention layer*, which exploits the statistical correlations among the sequence of tokens to refine the token representation. Transformers have been highly successful in learning compact representations that perform well on many downstream tasks. Yet the transformer network architecture is empirically designed and lacks a rigorous mathematical interpretation. In fact, the output of the attention layer itself has several competing interpretations [252, 154]. As a result, the statistical and geometric relationship between the data distribution and the final representation learned by a transformer largely remains a mysterious black box.

Diffusion models and denoising. Diffusion models [226, 108, 231, 232, 228] have recently become a popular method for learning the data distribution, particularly for generative tasks and natural image data which are highly structured but notoriously difficult to effectively model [257, 62]. The core concept of diffusion models is to start with features sampled from a Gaussian noise distribution (or some other standard template) and *iteratively denoise* and deform the feature distribution until it converges to the original data distribution. This process is computationally intractable if modeled in just one step [141], so it is typically broken into multiple incremental steps. The key to each step is the so-called *score function*, or equivalently [66] an estimate for the "optimal denoising function"; in practice this function is modeled using a generic black-box deep network. Diffusion models have shown effectiveness at learning and sampling from the data distribution [135, 37, 212]. However, despite some recent efforts [230], they generally do not establish any clear correspondence between the initial features and data samples. Hence, diffusion models themselves do not offer a parsimonious or interpretable representation of the data distribution.

Structure-seeking models and rate reduction. In both of the previous two methods, the representations were constructed implicitly as a byproduct of solving a downstream task (e.g., classification or generation/sampling) using deep networks. However, one can also explicitly learn a representation of the data distribution as a task in and of itself; this is most commonly done by trying to identify and represent low-dimensional structures in the input data. Classical examples of this paradigm include model-based approaches such as sparse coding [187, 44] and dictionary learning [233, 84, 299], out of which grew early attempts at designing and interpreting deep network architectures [194, 27]. More recent approaches build instead from a model-free perspective, where one learns a representation through a sufficiently-informative pretext task (such as compressing similar and separating dissimilar data in contrastive learning [243, 258, 222], or maximizing the information gain in the class of maximal coding rate reduction methods [172, 290, 33]). Compared to black-box deep learning approaches, both model-based and model-free representation learning schemes have the advantage of being more interpretable: they allow users to explicitly design desired properties of the learned representation [290, 33, 193]. Furthermore, they allow users to construct new white-box forward-constructed deep network architectures [83,

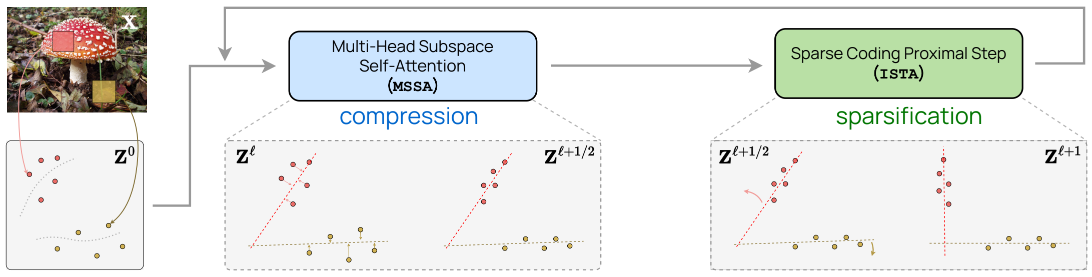

X
Z^0
Multi-Head Subspace Self-Attention (MSSA)
compression
Z^ℓ
Z^{ℓ+1/2}
Sparse Coding Proximal Step (ISTA)
sparsification
Z^{ℓ+1/2}
Z^{ℓ+1}

Figure 3.1: **The 'main loop' of the CRATE white-box deep network design.** After encoding input data $\boldsymbol{X}$ as a sequence of tokens $\boldsymbol{Z}^0$, CRATE constructs a deep network that transforms the data to a canonical configuration of low-dimensional subspaces by successive ***compression*** against a local model for the distribution, generating $\boldsymbol{Z}^{\ell+1/2}$, and ***sparsification*** against a global dictionary, generating $\boldsymbol{Z}^{\ell+1}$. Repeatedly stacking these blocks and training the model parameters via backpropagation yields a powerful and interpretable representation of the data.

33, 104] by *unrolling the optimization strategy for the representation learning objective*, such that each layer of the constructed network implements an iteration of the optimization algorithm [83, 33, 245]. Several recent works [283, 109, 55] consider the connections between transformer architectures [250] and unrolled optimization. Unfortunately, in this paradigm, if the desired properties are narrowly defined, it may be difficult to achieve good practical performance on large real-world datasets.

Our contributions, and outline of this work. In this work, we aim to remedy the limitations of these existing methods with a more unified framework for designing transformer-like network architectures that leads to both mathematical interpretability and good practical performance. To this end, we propose to learn a sequence of *incremental mappings* to obtain a most *compressed and sparse* representation for the input data (or their token sets) that optimizes *a unified objective function* known as the sparse rate reduction, specified later in Eq. (3.1). The goal of the mapping is illustrated in Chapter 3.1. Within this framework, we unify the above three seemingly disparate approaches and show that *transformer-like deep network layers can be naturally derived from unrolling iterative optimization schemes to incrementally optimize the sparse rate reduction objective.* In particular, our contributions and outline of this chapter are as follows:

- In Chapter 3.2.2 we show, using an idealized model for the token distribution, that if one *iteratively denoises* the tokens towards a family of low-dimensional subspaces, the associated score function assumes an explicit form similar to a self-attention operator seen in transformers.

- In Chapter 3.2.3 we derive the multi-head self-attention layer as an unrolled gradient descent step to minimize the lossy coding rate part of the rate reduction, showing another interpretation of the self-attention layer as compressing the token representation.

- In Chapter 3.2.4 we show that the multi-layer perceptron which immediately follows the multi-head self-attention in transformer blocks can be interpreted as (and replaced by) a layer which incrementally optimizes the remaining part of the sparse rate reduction objective by constructing a sparse coding of the token representations.

- In Chapter 3.2.5 we use this understanding to create a new white-box (fully mathematically interpretable) transformer architecture called CRATE (i.e., Coding RAte reduction TransformEr), where each layer performs a *single step* of an alternating minimization algorithm to optimize the sparse rate reduction objective.

Hence, within our framework, the learning objective function, the deep learning architecture, and the final learned representation *all become white boxes* that are fully mathematically interpretable. As the experiments in Chapter 3.3 show, the CRATE networks, despite being simple, can already learn the desired compressed and sparse representations on large-scale real-world datasets and achieve performance on par with much more heavily engineered transformer networks (such as ViT) on a wide variety of tasks (e.g., classification and transfer learning).

3.2 Technical Approach and Justification

3.2.1 Objective and Approach

We consider a general learning setup associated with real-world signals. We have some random variable $\boldsymbol{X} = \begin{bmatrix}\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\end{bmatrix} \in \mathbb{R}^{D \times N}$ which is our data source; each $\boldsymbol{x}_i \in \mathbb{R}^D$ is interpreted as a *token*[1], and the $\boldsymbol{x}_i$'s may have arbitrary correlation structures. We use $\boldsymbol{Z} = \begin{bmatrix}\boldsymbol{z}_1, \dots, \boldsymbol{z}_N\end{bmatrix} \in \mathbb{R}^{d \times N}$ to denote the random variable which defines our representations. Each $\boldsymbol{z}_i \in \mathbb{R}^d$ is the representation of the corresponding token $\boldsymbol{x}_i$. We are given $B \geq 1$ i.i.d. samples $\boldsymbol{X}_1, \dots, \boldsymbol{X}_B \sim \boldsymbol{X}$, whose tokens are $\boldsymbol{x}_{i,b}$. The representations of our samples are denoted $\boldsymbol{Z}_1, \dots, \boldsymbol{Z}_B \sim \boldsymbol{Z}$, and those of our tokens are $\boldsymbol{z}_{i,b}$. Finally, for a given network, we use $\boldsymbol{Z}^\ell$ to denote the output of the first ℓ layers when given $\boldsymbol{X}$ as input. Correspondingly, the sample outputs are $\boldsymbol{Z}_i^\ell$ and the token outputs are $\boldsymbol{z}_{i,b}^\ell$.

Objective for learning a structured and compact representation. Following the framework of rate reduction [33], we contend that the goal of representation learning is to find a feature mapping $f\colon \boldsymbol{X} \in \mathbb{R}^{D \times N} \to \boldsymbol{Z} \in \mathbb{R}^{d \times N}$ which transforms input data $\boldsymbol{X} \in \mathbb{R}^{D \times N}$ with a potentially nonlinear and multi-modal distribution to a (piecewise) *linearized and compact* feature representation $\boldsymbol{Z} \in \mathbb{R}^{d \times N}$. While the joint distribution of tokens $(\boldsymbol{z}_i)_{i=1}^N$ in $\boldsymbol{Z}$ may be sophisticated (and task-specific), we further contend that it is reasonable and practical to require that the target marginal distribution of individual tokens $\boldsymbol{z}_i$ should be highly compressed and structured, amenable for compact coding. Particularly, we require the distribution to be *a mixture of low-dimensional (say K) Gaussian distributions*, such that the k^{th} Gaussian has mean $\mathbf{0} \in \mathbb{R}^d$,

[1]For language transformers, tokens roughly correspond to words [250], while for vision transformers, tokens correspond to image patches [63].

covariance $\boldsymbol{\Sigma}_k \succeq \mathbf{0} \in \mathbb{R}^{d\times d}$, and support spanned by the orthonormal basis $\boldsymbol{U}_k \in \mathbb{R}^{d\times p}$. We denote $\boldsymbol{U}_{[K]} = (\boldsymbol{U}_k)_{k=1}^{K}$ to be the set of bases of all Gaussians. Hence to maximize the *information gain* [173] for the final token representation, we wish to maximize the rate reduction [172, 290] of the tokens, i.e., $\max_{\boldsymbol{Z}} \Delta R(\boldsymbol{Z}; \boldsymbol{U}_{[K]}) = R(\boldsymbol{Z}) - R^c(\boldsymbol{Z}; \boldsymbol{U}_{[K]})$, where R and R^c are estimates of lossy coding rates to be formally defined in Eq. (3.7) and Eq. (3.8). This also promotes token representations $\boldsymbol{z}_i$ from different Gaussians to be *incoherent* [290].

Since rate reduction is an intrinsic measure of goodness for the representation, it is invariant to arbitrary rotations of the representations. Therefore, to ensure the final representations are amenable to more compact coding, we would like to transform the representations (and their supporting subspaces) so that they become *sparse* with respect to the standard coordinates of the resulting representation space.[2] The combined rate reduction and sparsification process is illustrated in Chapter 3.1. Computationally, we may combine the above two goals into a unified objective for optimization:

$$\begin{aligned} \max_{f\in\mathcal{F}} \mathbb{E}_{\boldsymbol{Z}}\big[\Delta R(\boldsymbol{Z}; \boldsymbol{U}_{[K]}) - \lambda\|\boldsymbol{Z}\|_0\big] = \max_{f\in\mathcal{F}} \mathbb{E}_{\boldsymbol{Z}}\big[R(\boldsymbol{Z}) - R^c(\boldsymbol{Z}; \boldsymbol{U}_{[K]}) - \lambda\|\boldsymbol{Z}\|_0\big] \\ \text{s.t. } \boldsymbol{Z} = f(\boldsymbol{X}), \end{aligned} \tag{3.1}$$

where the ℓ^0 norm $\|\boldsymbol{Z}\|_0$ promotes the sparsity of the final token representations $\boldsymbol{Z} = f(\boldsymbol{X})$.[3] We call this objective "*sparse rate reduction.*"

White-box deep architecture as unrolled incremental optimization. Although easy to state, each term of the above objective can be computationally very challenging to optimize [270, 33]. Hence it is natural to take an approximation approach that realizes the global transformation f optimizing (3.1) through a concatenation of multiple, say L, simple *incremental and local* operations f^ℓ that push the representation distribution towards the desired parsimonious model distribution:

$$f\colon \boldsymbol{X} \xrightarrow{f^0} \boldsymbol{Z}^0 \rightarrow \cdots \rightarrow \boldsymbol{Z}^\ell \xrightarrow{f^\ell} \boldsymbol{Z}^{\ell+1} \rightarrow \cdots \rightarrow \boldsymbol{Z}^L = \boldsymbol{Z}, \tag{3.2}$$

where $f^0 : \mathbb{R}^D \rightarrow \mathbb{R}^d$ is the pre-processing mapping that transforms input tokens $\boldsymbol{x}_i \in \mathbb{R}^D$ to their token representations $\boldsymbol{z}_i^1 \in \mathbb{R}^d$.

Each incremental *forward mapping* $\boldsymbol{Z}^{\ell+1} = f^\ell(\boldsymbol{Z}^\ell)$, or a "layer", transforms the token distribution to *optimize* the above sparse rate reduction objective Eq. (3.1), conditioned on the distribution of its input tokens $\boldsymbol{Z}^\ell$. In contrast to other unrolled optimization approaches such as the ReduNet [33], we *explicitly model* the distribution of $\boldsymbol{Z}^\ell$ at each layer, say as a mixture of linear subspaces or sparsely generated from a dictionary. The model parameters are learned from data (say via *backward propagation* with end-to-end training). This separation of forward "optimization" and backward "learning" clarifies the mathematical role of each layer as an operator transforming the distribution of its input, whereas the input distribution is in turn modeled (and subsequently learned) by the parameters of the layer.

[2]That is, having the fewest nonzero entries.

[3]To simplify the notation, we will discuss the objective for one sample $\boldsymbol{X}$ at a time with the understanding that we always mean to optimize the expectation.

We show that we can derive these incremental, local operations through an unrolled optimization perspective to achieve Eq. (3.1) through Chapter 3.2.3, Chapter 3.2.4, Chapter 3.2.5. Once we decide on using an incremental approach to optimizing (3.1), there are a variety of possible choices to achieve the optimization. Given a model for $\boldsymbol{Z}^{\ell}$, say a mixture of subspaces $\boldsymbol{U}_{[K]}$, we opt for a two-step *alternating minimization* process with a strong conceptual basis: first in Chapter 3.2.3, we *compress* the tokens $\boldsymbol{Z}^{\ell}$ via a gradient step to minimize the coding rate term $\min_{\boldsymbol{Z}} R^{c}(\boldsymbol{Z}; \boldsymbol{U}_{[K]})$; second, in Chapter 3.2.4, we *sparsify* the compressed tokens, with a suitably-relaxed proximal gradient step on the difference of the sparsity penalty and the expansion term, i.e., $\min_{\boldsymbol{Z}}[\lambda \|\boldsymbol{Z}\|_0 - R(\boldsymbol{Z})]$. Both actions are applied incrementally and repeatedly, as each f^{ℓ} in Eq. (3.2) is instantiated with these two steps.

3.2.2 Self-Attention via Denoising Tokens Towards Multiple Subspaces

There are many different ways to optimize the objective Eq. (3.1) incrementally. In this work, we propose arguably *the most basic* scheme. To help clarify the intuition behind our derivation and approximation, in this section (and Appendix B.1.1) we study a largely idealized model which nevertheless captures the essence of nearly the whole process and particularly reveals the reason why self-attention-like operators arise in many contexts. Assume that $N = 1$, and the single token $\boldsymbol{x}$ is drawn i.i.d. from an unknown mixture of Gaussians $(\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_k))_{k=1}^{K}$ supported on low-dimensional subspaces with orthonormal bases $\boldsymbol{U}_{[K]} = (\boldsymbol{U}_k)_{k=1}^{K}$ and corrupted with additive Gaussian noise $\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, i.e.,

$$\boldsymbol{x} = \boldsymbol{z} + \sigma \boldsymbol{w}, \tag{3.3}$$

where $\boldsymbol{z}$ is distributed according to the mixture. Our goal is simply to transform the distribution of the noisy token $\boldsymbol{x}$ to the mixture of low-dimensional Gaussians $\boldsymbol{z}$. Towards incremental construction of a representation f for this model following Eq. (3.2), we reason inductively: if $\boldsymbol{z}^{\ell}$ is a noisy token Eq. (3.3) at noise level σ^{ℓ}, it is natural to produce $\boldsymbol{z}^{\ell+1}$ by denoising at the level σ^{ℓ}. In the mean-square sense, the optimal estimate is $\mathbb{E}[\boldsymbol{z} | \boldsymbol{z}^{\ell}]$, which has a variational characterization (e.g. [87]):

$$\mathbb{E}[\boldsymbol{z} | \cdot\,] = \arg\min_{f} \mathop{\mathbb{E}}_{\boldsymbol{z}, \boldsymbol{w}} \left[\| f(\boldsymbol{z} + \sigma^{\ell} \boldsymbol{w}) - \boldsymbol{z} \|_2^2 \right]. \tag{3.4}$$

Setting $\boldsymbol{z}^{\ell+1} = \mathbb{E}[\boldsymbol{z} | \boldsymbol{z}^{\ell}]$, Eq. (3.4) thus characterizes the next stage of Eq. (3.2) in terms of an optimization objective based on a *local signal model* for $\boldsymbol{z}^{\ell}$. Moreover, letting $\boldsymbol{x} \mapsto q^{\ell}(\boldsymbol{x})$ denote the density of $\boldsymbol{z}^{\ell}$, Tweedie's formula [66] allows us to express the optimal representation solving Eq. (3.4) in closed-form:

$$\boldsymbol{z}^{\ell+1} = \boldsymbol{z}^{\ell} + (\sigma^{\ell})^2 \nabla_{\boldsymbol{x}} \log q^{\ell}(\boldsymbol{z}^{\ell}). \tag{3.5}$$

Tweedie's formula expresses the optimal representation in terms of an additive correction (in general a nonlinear function of $\boldsymbol{z}^{\ell}$) to the noisy observations by the gradient of the *log-likelihood* of the distribution of the noisy observations, giving the optimal representation a clear interpretation as an incremental perturbation to the current noisy distribution q^{ℓ}. This connection is well-known in the areas of estimation theory and inverse problems [66, 234, 205, 178, 129, 251, 211], and more recently has found powerful applications in the training of generative models for natural images

[118, 255, 226, 232, 228]. Here, we can calculate a closed-form expression for this *score function* $\nabla_{\boldsymbol{x}} \log q^{\ell}$, which, when combined with Eq. (3.5) and some technical assumptions[4], gives the following approximation (shown in Chapter B.1.1). Let $\otimes$ denote the Kronecker product; then we have

$$\boldsymbol{z}^{\ell+1} \approx \begin{bmatrix}\boldsymbol{U}_1, \dots, \boldsymbol{U}_K\end{bmatrix} \left[\operatorname{diag}\left(\operatorname{softmax}\left(\frac{1}{2(\sigma^{\ell})^2}\begin{bmatrix}\|\boldsymbol{U}_1^{\top}\boldsymbol{z}^{\ell}\|_2^2 \\ \vdots \\ \|\boldsymbol{U}_K^{\top}\boldsymbol{z}^{\ell}\|_2^2\end{bmatrix}\right)\right) \otimes \boldsymbol{I}_p\right] \begin{bmatrix}\boldsymbol{U}_1^{\top}\boldsymbol{z}^{\ell} \\ \vdots \\ \boldsymbol{U}_K^{\top}\boldsymbol{z}^{\ell}\end{bmatrix}, \tag{3.6}$$

This operation resembles a self-attention layer in a standard transformer architecture with K heads, sequence length $N = 1$, the "query-key-value" constructs being replaced by a single linear projection $\boldsymbol{U}_k^{*}\boldsymbol{z}^{\ell}$ of the token $\boldsymbol{z}^{\ell}$, and the aggregation of head outputs (conventionally modeled by an MLP) done with the two leftmost matrices in Eq. (3.6). We thus derive the following useful interpretation, which we will exploit in the sequel: *Gaussian denoising against a mixture of subspaces model leads to self-attention-type layers in the transformation f.* Given an initial sample $\boldsymbol{x}$ following the model (3.3), we can repeatedly apply local transformations to the distribution with Eq. (3.6) in order to realize the incremental mapping $f\colon \boldsymbol{x} \to \boldsymbol{z}$ in (3.2).[5] These insights will guide us in the design of our white-box transformer architecture in the upcoming subsections.

3.2.3 Self-Attention via Compressing Token Sets through Optimizing Rate Reduction

In the last subsection, we have seen that the multi-head attention in a transformer resembles the score-matching operator that aims to transform a token $\boldsymbol{z}^{\ell}$ towards a mixture of subspaces (or degenerate Gaussians). Nevertheless, to carry out such an operation on any data, one needs to first learn or estimate, typically from finite samples, the parameters of the mixture of (degenerate) Gaussians, which is known to be a challenging task [172, 254]. This challenge is made even harder because in a typical learning setting, the given set of tokens are *not* i.i.d. samples from the mixture of subspaces. The joint distribution among these tokens can encode rich information about the data—for example, co-occurrences between words or object parts in language and image data (resp.)—which we should also learn. Thus, we should compress / denoise / transform such a set of tokens together. To this end, we need a measure of quality, i.e., compactness, for the resulting representation of the set of tokens.

A natural measure of the compactness of such a set of tokens is the (lossy) coding rate to encode them up to a certain precision $\varepsilon > 0$ [172, 290]. For a zero-mean Gaussian, this measure takes a closed form. If we view the tokens in $\boldsymbol{Z} \in \mathbb{R}^{d \times N}$ as drawn from a single zero-mean Gaussian, an

[4]Such as σ being smaller than the nonzero eigenvalues of $\boldsymbol{\Sigma}_k$ and the normalization assumption $\pi_i \det(\boldsymbol{\Sigma}_i + \sigma^2\boldsymbol{I})^{-1/2} = \pi_j \det(\boldsymbol{\Sigma}_j + \sigma^2\boldsymbol{I})^{-1/2}$ for all $i, j \in [K]$, where π_k is the mixture proportion for the k^{th} Gaussian.

[5]This statement can be made mathematically rigorous by exploiting a deep connection between neural ODEs and diffusion models, following ideas in Song et al. [232] and Chen, Daras, and Dimakis [38].

estimate of their (lossy) coding rate, subject to quantization precision $\varepsilon > 0$, is given in [172] as:

$$R(\boldsymbol{Z}) \doteq \frac{1}{2}\text{logdet}\left(\boldsymbol{I} + \frac{d}{N\varepsilon^2}\boldsymbol{Z}^*\boldsymbol{Z}\right) = \frac{1}{2}\text{logdet}\left(\boldsymbol{I} + \frac{d}{N\varepsilon^2}\boldsymbol{Z}\boldsymbol{Z}^*\right). \tag{3.7}$$

In practice, the data distribution is typically multi-modal, say an image set consisting of many classes or a collection of image patches as in Chapter 3.1. It is more appropriate to require that the set of tokens map to a mixture of, say K, subspaces (degenerate Gaussians) [33]. As before we denote the (to be learned) bases of these subspaces as $\boldsymbol{U}_{[K]} = (\boldsymbol{U}_k)_{k=1}^K$, where $\boldsymbol{U}_k \in \mathbb{R}^{d\times p}$. Although the joint distribution of the tokens $\boldsymbol{Z}$ is unknown, the desired marginal distribution of each token $\boldsymbol{z}_i$ is a mixture of subspaces. So we may obtain an upper bound of the coding rate for the token set $\boldsymbol{Z}$ by projecting its tokens onto these subspaces and summing up the respective coding rates:

$$R^c(\boldsymbol{Z}; \boldsymbol{U}_{[K]}) = \sum_{k=1}^K R(\boldsymbol{U}_k^*\boldsymbol{Z}) = \frac{1}{2}\sum_{k=1}^K \text{logdet}\left(\boldsymbol{I} + \frac{p}{N\varepsilon^2}(\boldsymbol{U}_k^*\boldsymbol{Z})^*(\boldsymbol{U}_k^*\boldsymbol{Z})\right). \tag{3.8}$$

We would like to compress (or denoise) the set of tokens against these subspaces by minimizing the coding rate. The gradient of $R^c(\boldsymbol{Z}; \boldsymbol{U}_{[K]})$ is

$$\nabla_{\boldsymbol{Z}} R^c(\boldsymbol{Z}; \boldsymbol{U}_{[K]}) = \frac{p}{N\varepsilon^2}\sum_{k=1}^K \boldsymbol{U}_k\boldsymbol{U}_k^*\boldsymbol{Z}\left(\boldsymbol{I} + \frac{p}{N\varepsilon^2}(\boldsymbol{U}_k^*\boldsymbol{Z})^*(\boldsymbol{U}_k^*\boldsymbol{Z})\right)^{-1}. \tag{3.9}$$

The above expression approximates the residual of each projected token $\boldsymbol{U}_k^*\boldsymbol{z}_i$ regressed by other tokens $\boldsymbol{U}_k^*\boldsymbol{z}_j$ [33]. But, differently from [33], not all tokens in $\boldsymbol{Z}$ are from the same subspace. Hence, to denoise each token with tokens from its own group, we can compute their similarity through an auto-correlation among the projected tokens as $(\boldsymbol{U}_k^*\boldsymbol{Z})^*(\boldsymbol{U}_k^*\boldsymbol{Z})$ and convert it to a distribution of membership with a softmax, namely $\text{softmax}\left((\boldsymbol{U}_k^*\boldsymbol{Z})^*(\boldsymbol{U}_k^*\boldsymbol{Z})\right)$. Then, as we show in Chapter B.1.2, if we only use similar tokens to regress and denoise each other, then a gradient step on the coding rate with learning rate κ can be naturally approximated as follows:

$$\boldsymbol{Z}^{\ell+1/2} = \boldsymbol{Z}^{\ell} - \kappa\nabla_{\boldsymbol{Z}} R^c(\boldsymbol{Z}^{\ell}; \boldsymbol{U}_{[K]}) \approx \left(1 - \kappa\cdot\frac{p}{N\varepsilon^2}\right)\boldsymbol{Z}^{\ell} + \kappa\cdot\frac{p}{N\varepsilon^2}\cdot\texttt{MSSA}(\boldsymbol{Z}^{\ell} \mid \boldsymbol{U}_{[K]}), \tag{3.10}$$

where `MSSA` is defined through an `SSA` operator as:

$$\texttt{SSA}(\boldsymbol{Z} \mid \boldsymbol{U}_k) \doteq (\boldsymbol{U}_k^*\boldsymbol{Z})\text{softmax}\left((\boldsymbol{U}_k^*\boldsymbol{Z})^*(\boldsymbol{U}_k^*\boldsymbol{Z})\right), \quad k \in [K], \tag{3.11}$$

$$\texttt{MSSA}(\boldsymbol{Z} \mid \boldsymbol{U}_{[K]}) \doteq \frac{p}{N\varepsilon^2}\cdot\begin{bmatrix}\boldsymbol{U}_1, \dots, \boldsymbol{U}_K\end{bmatrix}\begin{bmatrix}\texttt{SSA}(\boldsymbol{Z} \mid \boldsymbol{U}_1) \\ \vdots \\ \texttt{SSA}(\boldsymbol{Z} \mid \boldsymbol{U}_K)\end{bmatrix}. \tag{3.12}$$

3.2.4 MLP via Iterative Shrinkage-Thresholding Algorithms (ISTA) for Sparse Coding

In the previous subsection, we focused on how to compress a set of tokens against a set of (learned) low-dimensional subspaces. Optimizing the remaining terms in the sparse rate reduction objective (3.1), including the non-smooth term, serves to sparsify the compressed tokens, hence leading to a more compact and structured (i.e., *parsimonious*) representation. From Eq. (3.1) amd Eq. (3.7), this term is

$$\max_{\boldsymbol{Z}}\,[R(\boldsymbol{Z}) - \lambda\|\boldsymbol{Z}\|_0] = \min_{\boldsymbol{Z}}\left[\lambda\|\boldsymbol{Z}\|_0 - \frac{1}{2}\mathrm{logdet}\left(\boldsymbol{I} + \frac{d}{N\varepsilon^2}\boldsymbol{Z}^*\boldsymbol{Z}\right)\right], \tag{3.13}$$

where $R(\boldsymbol{Z})$ denotes the coding rate of the whole token set, as defined in Eq. (3.7). In addition to sparsification via the $\|\boldsymbol{Z}\|_0$ term, the expansion term $R(\boldsymbol{Z})$ in (3.13) promotes diversity and non-collapse of the representation, a highly desirable property. However, prior work has struggled to realize this benefit on large-scale datasets due to poor scalability of the gradient $\nabla_{\boldsymbol{Z}} R(\boldsymbol{Z})$, which requires a matrix inverse [33].

To simplify things, we therefore take a different approach to trading off between representational diversity and sparsification: we posit a (complete) incoherent or orthogonal dictionary $\boldsymbol{D} \in \mathbb{R}^{d\times d}$, and ask to sparsify the intermediate iterates $\boldsymbol{Z}^{\ell+1/2}$ with respect to $\boldsymbol{D}$. That is, $\boldsymbol{Z}^{\ell+1/2} = \boldsymbol{D}\boldsymbol{Z}^{\ell+1}$ where $\boldsymbol{Z}^{\ell+1}$ is more sparse. The dictionary $\boldsymbol{D}$ is global, i.e., is used to sparsify all tokens simultaneously.

By the incoherence assumption, we have $\boldsymbol{D}^*\boldsymbol{D} \approx \boldsymbol{I}_d$; thus from Eq. (3.7) we have $R(\boldsymbol{Z}^{\ell+1}) \approx R(\boldsymbol{D}\boldsymbol{Z}^{\ell+1}) = R(\boldsymbol{Z}^{\ell+1/2})$. Thus we approximately solve Eq. (3.13) with the following program:

$$\boldsymbol{Z}^{\ell+1} = \arg\min_{\boldsymbol{Z}} \|\boldsymbol{Z}\|_0 \quad \text{subject to} \quad \boldsymbol{Z}^{\ell+1/2} = \boldsymbol{D}\boldsymbol{Z}. \tag{3.14}$$

The above sparse representation program is usually solved by relaxing it to an unconstrained convex program, known as LASSO:

$$\boldsymbol{Z}^{\ell+1} = \arg\min_{\boldsymbol{Z}} \left[\lambda\|\boldsymbol{Z}\|_1 + \|\boldsymbol{Z}^{\ell+1/2} - \boldsymbol{D}\boldsymbol{Z}\|_F^2\right]. \tag{3.15}$$

In our implementation, motivated by Sun, Nasrabadi, and Tran [237] and Zarka et al. [296], we also add a non-negative constraint to $\boldsymbol{Z}^{\ell+1}$,

$$\boldsymbol{Z}^{\ell+1} = \arg\min_{\boldsymbol{Z}\geq \mathbf{0}} \left[\lambda\|\boldsymbol{Z}\|_1 + \|\boldsymbol{Z}^{\ell+1/2} - \boldsymbol{D}\boldsymbol{Z}\|_F^2\right], \tag{3.16}$$

which we then incrementally optimize by performing an unrolled proximal gradient descent step, known as an ISTA step [17], to give the update:

$$\boldsymbol{Z}^{\ell+1} = \mathrm{ReLU}(\boldsymbol{Z}^{\ell+1/2} + \eta\boldsymbol{D}^*(\boldsymbol{Z}^{\ell+1/2} - \boldsymbol{D}\boldsymbol{Z}^{\ell+1/2}) - \eta\lambda\mathbf{1}) \doteq \mathtt{ISTA}(\boldsymbol{Z}^{\ell+1/2} \mid \boldsymbol{D}). \tag{3.17}$$

As shown in Appendix A.3 of [289], one can arrive at a similar operator to the above ISTA-like update for optimizing Eq. (3.13) by properly linearizing and approximating the rate term $R(\boldsymbol{Z})$.

Here the SSA operator in Eq. (3.11) resembles the *attention operator* in a typical transformer [250], except that here the linear operators of value, key, and query are all set to be *the same* as the subspace basis, i.e., $\boldsymbol{V} = \mathcal{K} = \boldsymbol{Q} = \boldsymbol{U}_k^*$.[6] Hence, we name $\texttt{SSA}(\,\cdot\,|\boldsymbol{U}_k) : \mathbb{R}^{d\times N} \to \mathbb{R}^{p\times N}$ the **S**ubspace **S**elf-**A**ttention (SSA) operator (more details and justification can be found in Eq. (B.18) in Chapter B.1.2). Then, the whole MSSA operator in Eq. (3.12), formally defined as $\texttt{MSSA}(\,\cdot\,|\boldsymbol{U}_{[K]}) \colon \mathbb{R}^{d\times N} \to \mathbb{R}^{d\times N}$ and called the **M**ulti-Head **S**ubspace **S**elf-**A**ttention (MSSA) operator, aggregates the attention head outputs by averaging using model-dependent weights, similar in concept to the popular multi-head self-attention operator in existing transformer networks. The overall gradient step Eq. (3.10) resembles the multi-head self-attention implemented with a skip connection in transformers.

Notice that if we have $N = 1$ tokens as well as take an aggressive gradient step ($\kappa = 1$) and tune the quantization error ($\varepsilon = \sqrt{p/N}$), the multi-head subspace self-attention operator in Eq. (3.12) becomes the ideal denoiser defined in Eq. (3.6), with the one minor difference that the aggregation of the heads is done by a linear function here, while in Eq. (3.6) it is done by a nonlinear mixture-of-experts type function.[7] This provides two very related interpretations of the multi-head self-attention operator, as denoising and compression against a mixture of low-dimensional subspaces.

3.2.5 The Overall White-Box CRATE Architecture

By combining the above two steps:

1. (Chapter 3.2.2 and Chapter 3.2.3) Local denoising and compression of tokens within a sample towards a mixture-of-subspace structure, leading to the multi-head subspace self-attention block – MSSA;

2. (Chapter 3.2.4) Global compression and sparsification of token sets across all samples through sparse coding, leading to the sparsification block – ISTA;

we can get the following rate-reduction-based transformer layer, illustrated in Figure 3.2,

$$\boldsymbol{Z}^{\ell+1/2} \doteq \boldsymbol{Z}^{\ell} + \texttt{MSSA}(\boldsymbol{Z}^{\ell} \mid \boldsymbol{U}_{[K]}^{\ell}), \qquad \boldsymbol{Z}^{\ell+1} \doteq \texttt{ISTA}(\boldsymbol{Z}^{\ell+1/2} \mid \boldsymbol{D}^{\ell}). \tag{3.18}$$

Composing multiple such layers following the incremental construction of our representation in Eq. (3.2), we obtain a white-box transformer architecture that transforms the data tokens towards a compact and sparse union of incoherent subspaces.

This model has the parameters $(\boldsymbol{U}_{[K]}^{\ell})_{\ell=1}^{L}$ and $(\boldsymbol{D}^{\ell})_{\ell=1}^{L}$, which are learned from data via *back-propagation*. Notably, in each layer ℓ, the learned $\boldsymbol{U}_{[K]}^{\ell}$ retain their interpretation as incoherent bases for supporting subspaces for the mixture-of-Gaussians model at layer ℓ, and the learned $\boldsymbol{D}^{\ell}$

[6]We note a recent suggestion of Hinton [103] that it is more sensible to set the "value, key, and query" projection matrices in a transformer to be equal. Our derivation in this section confirms this mathematically.

[7]This suggests that we could also consider such a mixture of expert type aggregation of the multiple attention heads. In this work, we use linear aggregation, and leave evaluation of more variants for future work.

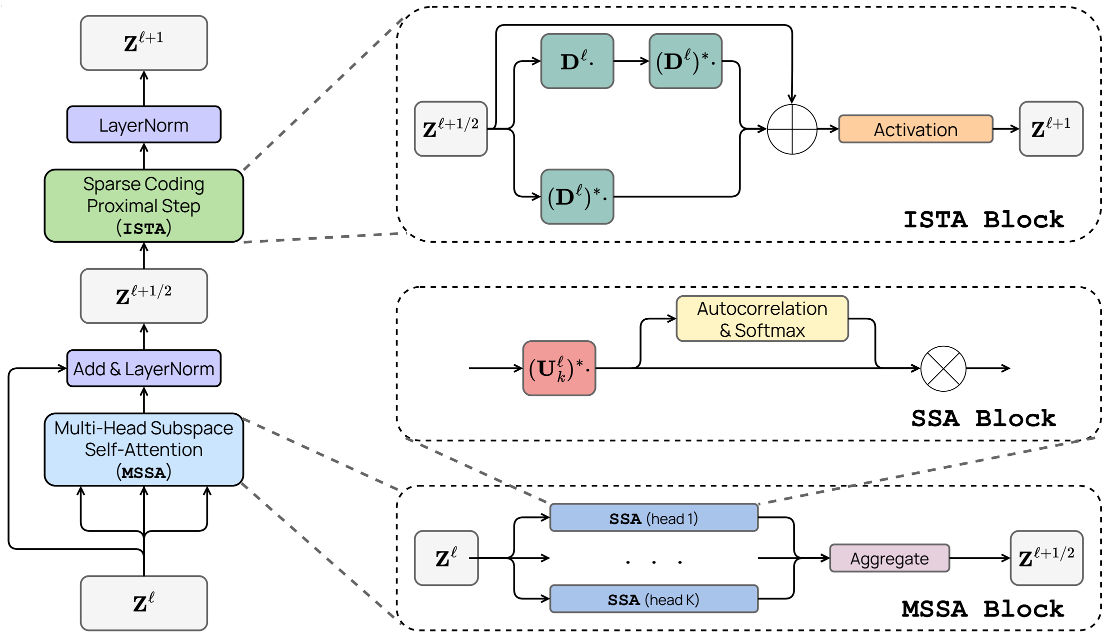


Figure 3.2: One layer of the CRATE architecture. The full architecture is simply a concatenation of such layers, with some initial tokenizer and final task-specific architecture (i.e., a classification head).

retains its interpretation as a sparsifying dictionary at layer ℓ. We emphasize that the parameters $\boldsymbol{U}_{[K]}^{\ell}$ and $\boldsymbol{D}^{\ell}$ are dependent on the layer ℓ — that is, we learn a different set of parameters at each layer. This is because at each layer we learn an approximate local parametric model for the input data distribution, then use that learned model to construct the layer operators that transform the distribution. Our procedure of parameterizing the data distribution at each layer distinguishes this work from previous works on unrolled optimization for neural networks such as the ReduNet [33]. Our interpretation clarifies the roles of the network forward pass (given local signal models at each layer, denoise/compress/sparsify the input) and the backward pass (learn the local signal models from data via supervision).

We note that in this work, at each stage of our construction, we have chosen arguably the *simplest possible* construction to use. We can substitute each part of this construction, so long as the new part maintains the same conceptual role, and obtain another white-box architecture. Nevertheless, our such-constructed architecture, called CRATE (i.e., Coding RAte TransformEr), connects to existing transformer models, obtains competitive results on real-world datasets, and is fully mathematically interpretable.

3.3 Experiments

In this section, we conduct experiments to study the performance of our proposed white-box transformer CRATE on real-world datasets and tasks. As the analysis in Chapter 3.2 suggests, either the compression or the sparsification step can be achieved through various alternative design choices or strategies. CRATE arguably adopts the most basic choices and so our goal with the experiments is *not* simply to compete with other heavily engineered transformers while using such a rudimentary design. Rather, our goals are twofold. First, unlike any empirically designed black-box networks that are usually evaluated only on end-to-end performance, the white-box design of our network allows us to *look inside* the deep architecture and verify if layers of the learned network indeed perform their design objective—say performing incremental optimization for the objective Eq. (3.1). Second, despite their simplicity, our experiments will actually reveal the vast practical potential of our so-derived CRATE architectures since, as we will show, they already achieve very strong performance on large-scale real-world datasets and tasks. In the remainder of this section we highlight a selection of results; additional experimental details and results can be found in [289].

Model architecture. We implement the architecture that is described in Chapter 3.2.5, with minor modifications that are described in Chapter B.1.3. We consider different model sizes of CRATE by varying the token dimension d, number of heads K, and the number of layers L. We consider four model sizes in this work: CRATE-Tiny, CRATE-Small, CRATE-Base, and CRATE-Large. A PyTorch-style pseudocode can be found in Chapter B.1.3, which contains more implementation details. For training using supervised classification, we first take the `CLS` token $\overline{\boldsymbol{z}}_b = \boldsymbol{z}_{1,b}^{L+1}$ of for each sample, then apply a linear layer; the output of this linear layer $\boldsymbol{u}_b \doteq \boldsymbol{W}\overline{\boldsymbol{z}}_b$ is used as input to the standard cross-entropy loss. The overall loss averages over all samples $b \in [B]$.

Datasets and optimization. We mainly consider ImageNet-1K [59] as the testbed for our architecture. Specifically, we apply the Lion optimizer [42] to train CRATE models with different model sizes. Meanwhile, we also evaluate the transfer learning performance of CRATE: by considering the models trained on ImageNet-1K as pre-trained models, we fine-tune CRATE on several commonly used downstream datasets (CIFAR10/100, Oxford Flowers, Oxford-IIT-Pets). More details about the training and datasets can be found in Chapter B.1.3.

3.3.1 In-depth Layer-wise Analysis of CRATE

Do layers of CRATE achieve their design goals? As described in Chapter 3.2.3 and Chapter 3.2.4, the `MSSA` block is designed to optimize the compression term $R^c(\boldsymbol{Z})$ and the `ISTA` block to sparsify the token representations (corresponding to the sparsification term $\|\boldsymbol{Z}\|_0$). To understand whether CRATE indeed optimizes these terms, for each layer ℓ, we measure (i) the compression term $R^c(\boldsymbol{Z}^{\ell+1/2})$ on the `MSSA` block outputs $\boldsymbol{Z}^{\ell+1/2}$; and (ii) sparsity $\|\boldsymbol{Z}^{\ell+1}\|_0$ on the `ISTA` block outputs $\boldsymbol{Z}^{\ell+1}$. Specifically, we evaluate these two terms by using training/validation samples from ImageNet-1K. Both terms are evaluated at the per-sample level and averaged over $B = 10^3$ samples.

Chapter 3.4 shows the plots of these two key measures at all layers for the learned CRATE-small model. We find that as the layer index ℓ increases, both the compression and the sparsification terms improve in most cases. The increase in the sparsity measure of the last layer is caused by the extra linear layer for classification.[8] These results suggest that CRATE aligns well with the original design goals: once learned, it essentially learns to gradually compress and sparsity the representations through its layers. In addition, we also measure the compression and sparsification terms on CRATE models with different model sizes as well as intermediate model checkpoints and the results are shown by plots in Figure 3.3. The observations are very consistent across all different model sizes—both the compression and sparsification terms improve in most scenarios. Models with more layers tend to optimize the objectives more effectively, confirming our understanding of each layer's roles.

To see the effect of learning, we present the evaluations on CRATE-Small trained with different number of epochs in Chapter 3.5. When the model is not trained enough (e.g. untrained), the architecture does not optimize the objectives effectively. However, during training—learning better subspaces $\boldsymbol{U}_{[K]}^{\ell}$ and dictionaries $\boldsymbol{D}^{\ell}$—the designed blocks start to optimize the objectives much more effectively.

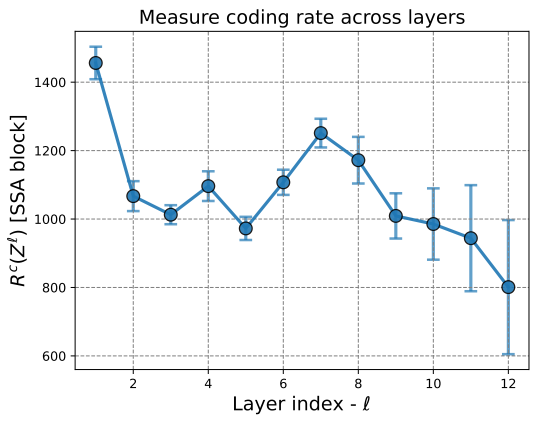
Visualizing layer-wise token representations. To gain a better understanding of the token representations of CRATE, we visualize the output of each `ISTA` block at layer ℓ in Figure 3.7. Specifically, we visualize the $\boldsymbol{Z}^{\ell+1}$ via heatmap plots. We observe that the output $\boldsymbol{Z}^{\ell+1}$ becomes more sparse as the layer increases. Moreover, besides the sparsity, we also find that $\boldsymbol{Z}^{\ell+1}$ becomes more structured (i.e., low-rank), which indicates that the set of token representations become closer to linear subspaces, confirming our mental picture of the geometry of each layer (as in Chapter 3.1).

Visualizing layer-wise subspaces in multi-head self-attention. We now visualize the $\boldsymbol{U}_{[K]}^{\ell}$ matrices used in the `MSSA` block. In Chapter 3.2.3, we assumed that $\boldsymbol{U}_{[K]}^{\ell}$ were incoherent to capture different "views" of the set of tokens. In Fig. 3.6, we first normalize the columns in each $\boldsymbol{U}_{k}^{\ell}$, then we visualize the $[\boldsymbol{U}_1^{\ell}, \ldots, \boldsymbol{U}_K^{\ell}]^{*}[\boldsymbol{U}_1^{\ell}, \ldots, \boldsymbol{U}_K^{\ell}] \in \mathbb{R}^{pK \times pK}$. The (i, j)-th block in each sub-figure corresponds to $(\boldsymbol{U}_i^{\ell})^{*}\boldsymbol{U}_j^{\ell}$ for $i, j \in [K]$ at a particular layer ℓ. We find that the learned $\boldsymbol{U}_{[K]}^{\ell}$ are approximately incoherent, which aligns well with our assumptions. One interesting observation is that the $\boldsymbol{U}_{[K]}^{\ell}$ becomes more incoherent when the layer index ℓ is larger, which suggests that the token representations are more separable. This mirrors the situation in other popular deep networks [91].

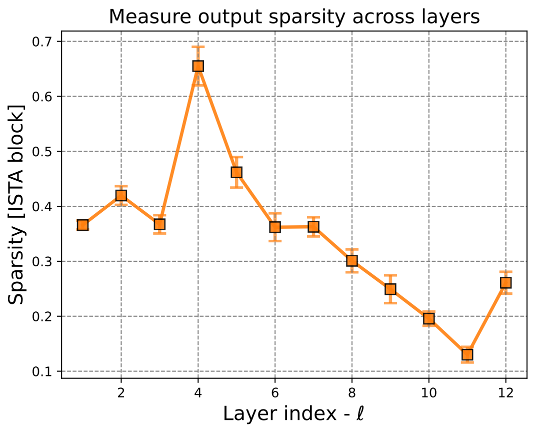
3.3.2 Evalutions of CRATE on Large Real-World Datasets and Tasks

We now study the empirical performance of the proposed networks by measuring their top-1 accuracy on ImageNet-1K as well as transfer learning performance on several widely used downstream datasets. We summarize the results in Chapter 3.1. As our designed architecture leverages parameter sharing in both the attention block (`MSSA`) and the MLP block (`ISTA`), our CRATE-Base model (22.08 million) has a similar number of parameters to the ViT-Small (22.05 million).

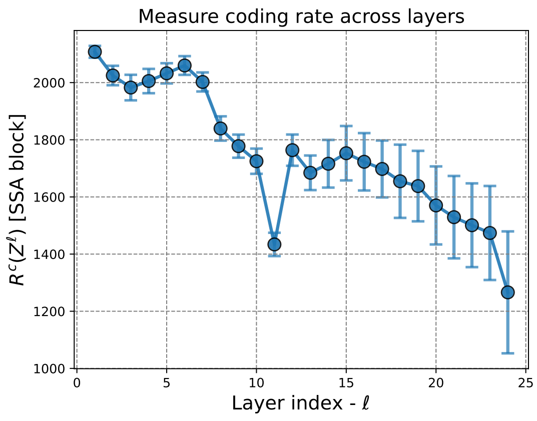
[8] Note that the learned sparse (tokens) features need to be mixed in the last layer for predicting the class. The phenomenon of increase in the sparsity measure at the last layer suggests that each class of objects may be associated with a number of features, and some of these features are likely to be shared across different classes.



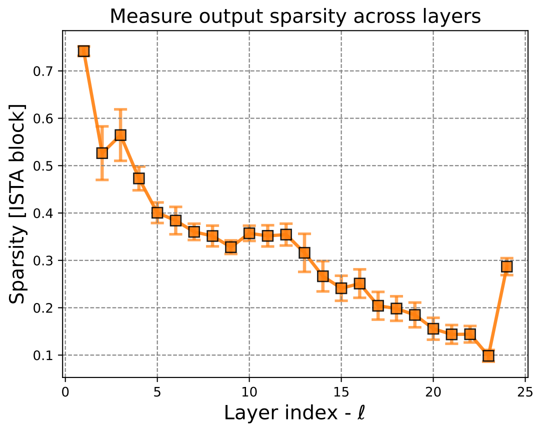
(a) Compression (Model: CRATE-Base).



(b) Sparsity (Model: CRATE-Base).



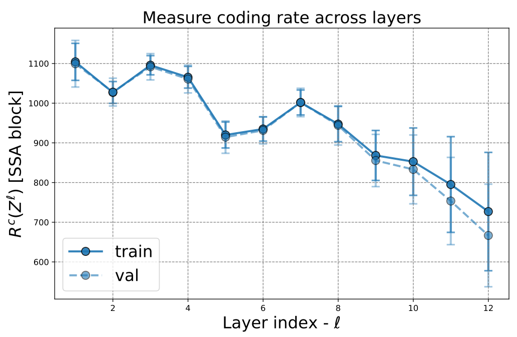
(c) Compression (Model: CRATE-Large).



(d) Sparsity (Model: CRATE-Large).

Figure 3.3: *Left*: The compression term $R^c(\boldsymbol{Z}^{\ell+1/2})$ of the MSSA outputs at different layers. *Right*: the sparsity of the ISTA output block, $\|\boldsymbol{Z}^{\ell+1}\|_0/(d \cdot N)$, at different layers.

From Chapter 3.1, we find that with a similar number of model parameters, our proposed network achieves similar ImageNet-1K and transfer learning performance as ViT, despite the simplicity and interpretability of our design. Moreover, with the same set of training hyperparameters, we observe promising scaling behavior in CRATE—we consistently improve the performance by scaling up the model size. For comparison, directly scaling ViT on ImageNet-1K does not always lead to consistent performance improvement measured by top-1 accuracy [63]. To summarize, we achieve promising performance on real-world large-scale datasets by directly implementing our principled architecture.



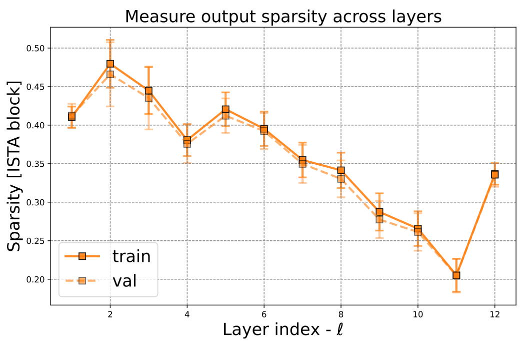
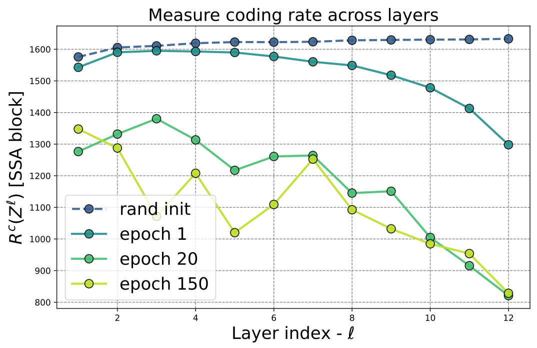


Figure 3.4: *Left*: The compression term $R^c(\boldsymbol{Z}^{\ell+1/2})$ of the MSSA outputs at different layers. *Right*: the sparsity of the ISTA output block, $\|\boldsymbol{Z}^{\ell+1}\|_0/(d \cdot N)$, at different layers. (Model: CRATE-Small).



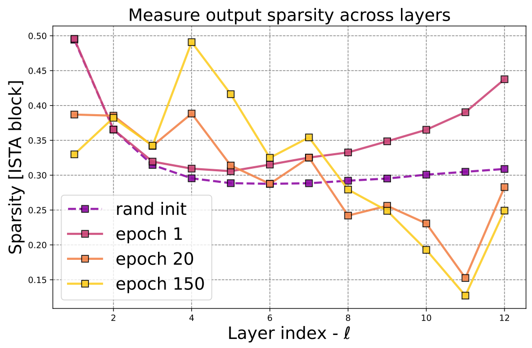
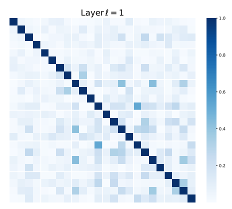


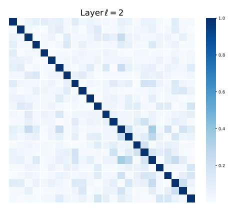
Figure 3.5: The compression term $R^c(\boldsymbol{Z})$ (*left*) and sparsification term $\|\boldsymbol{Z}\|_0/(d \cdot N)$ (*right*) across models trained with different numbers of epochs. (Model: CRATE-Base).

Table 3.1: Top 1 accuracy of CRATE on various datasets with different model scales when pre-trained on ImageNet. For ImageNet/ImageNetReaL, we directly evaluate the top-1 accuracy. For other datasets, we use models that are pre-trained on ImageNet as initialization and the evaluate the transfer learning performance via fine-tuning.

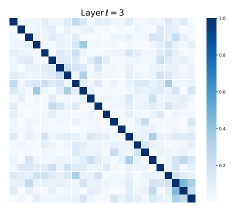
Datasets	CRATE-T	CRATE-S	CRATE-B	CRATE-L	ViT-T	ViT-S
# parameters	6.09M	13.12M	22.80M	77.64M	5.72M	22.05M
ImageNet	66.7	69.2	70.8	71.3	71.5	72.4
ImageNet ReaL	74.0	76.0	76.5	77.4	78.3	78.4
CIFAR10	95.5	96.0	96.8	97.2	96.6	97.2
CIFAR100	78.9	81.0	82.7	83.6	81.8	83.2
Oxford Flowers-102	84.6	87.1	88.7	88.3	85.1	88.5
Oxford-IIIT-Pets	81.4	84.9	85.3	87.4	88.5	88.6



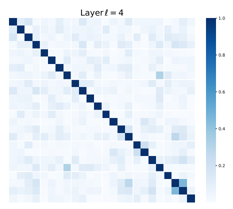
(a) $\ell = 1$.



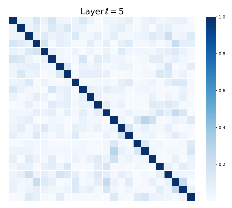
(b) $\ell = 2$.



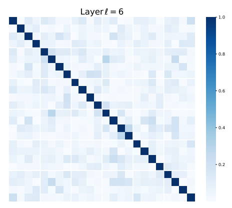
(c) $\ell = 3$.



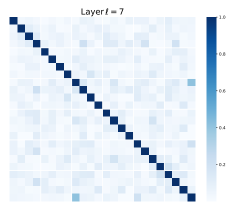
(d) $\ell = 4$.



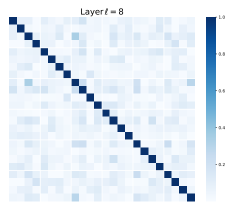
(e) $\ell = 5$.



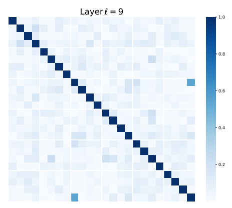
(f) $\ell = 6$.



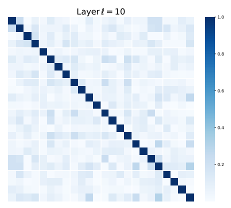
(g) $\ell = 7$.



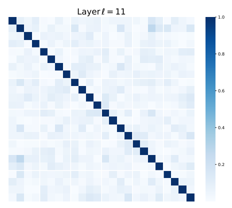
(h) $\ell = 8$.



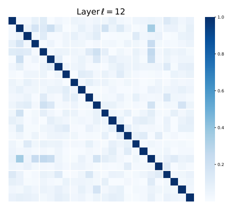
(i) $\ell = 9$.



(j) $\ell = 10$.



(k) $\ell = 11$.

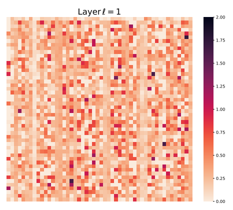


(l) $\ell = 12$.

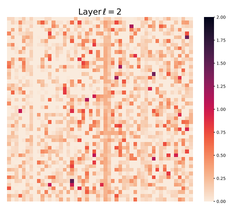
Figure 3.6: We visualize the $[\boldsymbol{U}_1^\ell, \dots, \boldsymbol{U}_K^\ell]^*[\boldsymbol{U}_1^\ell, \dots, \boldsymbol{U}_K^\ell] \in \mathbb{R}^{pK \times pK}$ at different layers. The (i, j)-th block in each sub-figure corresponds to $(\boldsymbol{U}_i^\ell)^* \boldsymbol{U}_j^\ell$ for $i, j \in [K]$ at a particular layer ℓ. To enhance the visual clarity, for each subspace $\boldsymbol{U}_i$, we randomly pick 4 directions for display purposes. (Model: CRATE-Tiny)

3.3.3 Emergence of Semantic Properties in Learned CRATE Attention Maps

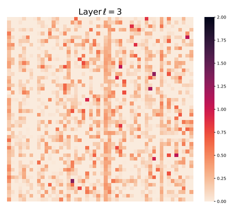
In this subsection, we analyze the attention maps within CRATE models trained on vision tasks. Previous work [32] use the self-attention in vision transformers to study semantic segmentation of the input image. [32] demonstrated that a specific self-supervised training method, named DINO, can lead to the emergence of segmentation in vision transformers. On the other hand, ViTs trained with supervised learning do not have such properties. In contrast, as we will present in this subsection, we find that *the white-box design of* CRATE *leads to the emergence of segmentation properties in the network's self-attention maps, solely through a minimalistic supervised training recipe—the supervised classification training used in vanilla supervised ViTs*. We quantify the segmentation properties of CRATE both qualitatively and quantitatively and compare the results with ViTs. Through extensive evaluations, we find that the self-attention maps in white-box



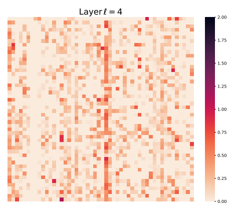
(a) $\ell = 1$.



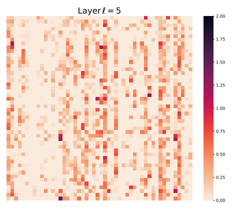
(b) $\ell = 2$.



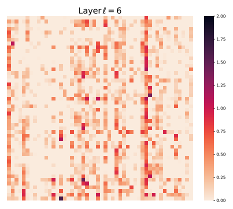
(c) $\ell = 3$.



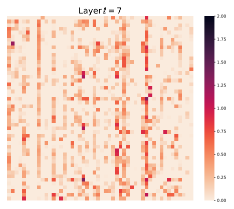
(d) $\ell = 4$.



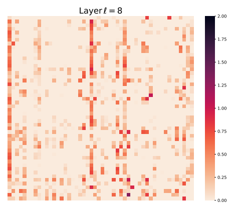
(e) $\ell = 5$.



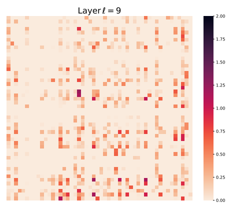
(f) $\ell = 6$.



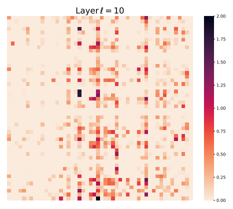
(g) $\ell = 7$.



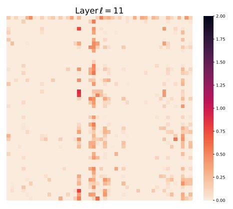
(h) $\ell = 8$.



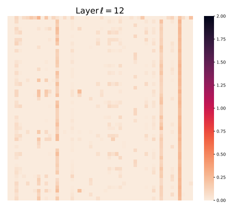
(i) $\ell = 9$.



(j) $\ell = 10$.



(k) $\ell = 11$.



(l) $\ell = 12$.

Figure 3.7: Visualizing layer-wise token $\boldsymbol{Z}^{\ell}$ representations at each layer ℓ. To enhance the visual clarity, we randomly extract a 50×50 sub-matrix from $\boldsymbol{Z}^{\ell}$ for display purposes. (Model: CRATE-Tiny)

transformers (CRATE) are much more interpretable than vanilla black-box vision transformers.

3.3.3.1 Experimental Setup

Model architecture We utilize the CRATE model at sizes -S/8 and -B/8 (that is, CRATE-Small and CRATE-Base with patch size 8×8). We similarly adopt the ViT model from [63] using the same scales (-S/8 and -B/8), ensuring consistent configurations between them. Refer to Table B.1 for detailed model performance evaluations on classification tasks.

Datasets. In this subsection, all visual models are trained for classification tasks, using the methodology described in the beginning of Chapter 3.3, on the complete ImageNet dataset [59], commonly referred to as ImageNet-21K. This dataset comprises 14,197,122 images distributed across 21,841 classes. We apply the MaskCut [266] pipeline on the COCO val2017 [166], which consists of 5,000 RGB images, and assess our models' performance for both object detection and

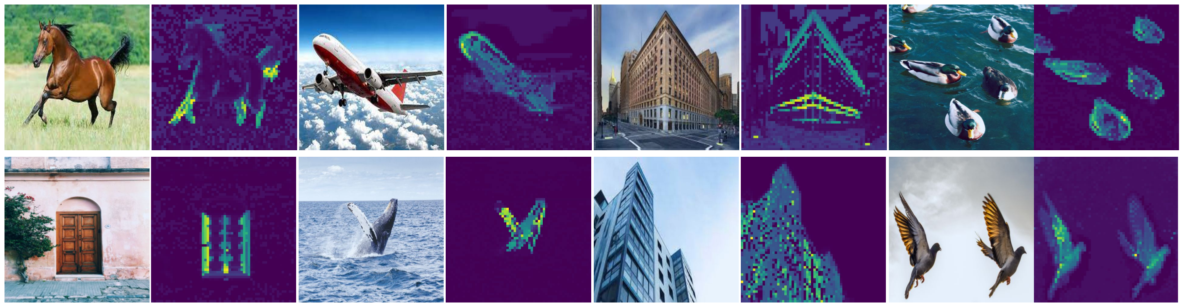


Figure 3.8: **Self-attention maps from a supervised CRATE with 8×8 patches** trained using classification. The CRATE architecture automatically learns to perform object segmentation without a complex self-supervised training recipe or any fine-tuning with segmentation-related annotations. For each image pair, we visualize the original image on the left and the self-attention map of the image on the right.

instance segmentation tasks. *All evaluation procedures are unsupervised, and we do not update the model weights during the detection and segmentation evaluation processes.*

3.3.3.2 Measuring the Emergence of Segmentation

Visualizing self-attention maps. To qualitatively measure the emergence phenomenon, we adopt the attention map approach based on the [CLS] token, which has been widely used as a way to interpret and visualize transformer-like architectures [2, 32]. Indeed, we use the same methodology as [2, 32], noting that in CRATE the query-key-value matrices are all the same; a more formal accounting is deferred to . The visualization results of self-attention maps can be found in Appendix B.1 of [291]. We observe that the self-attention maps of the CRATE model correspond to semantic regions in the input image. Our results suggest that the CRATE model encodes a clear semantic segmentation of each image in the network's internal representations, which is similar to the self-supervised method DINO [32]. In contrast, as shown in Chapter B.1 in the Appendices, the vanilla ViT trained on supervised classification does not exhibit similar segmentation properties.

Object detection and fine-grained segmentation. To further validate and evaluate the rich semantic information captured by CRATE, we employ MaskCut [266], a recent effective approach for object detection and segmentation that does not require human annotations. A more detailed methodological description can be found in [266]. This procedure allows us to extract more fine-grained segmentation for an image based on the token representations learned by different methods. In Figure 3.10, we visualize the fine-grained segmentation produced by MaskCut on features from ViT trained by classification, CRATE trained by classification, and CRATE trained via DINO [32], respectively. We compare the segmentation and detection performance in Table 3.2. Based on these results, we observe that MaskCut on features learned with supervised ViT typically fails to produce good segmentation masks, for example, the first image in Figure 3.10 and the ViT-S/8 row in Table 3.2. On the other hand, we notice that, regardless of the training task (supervised

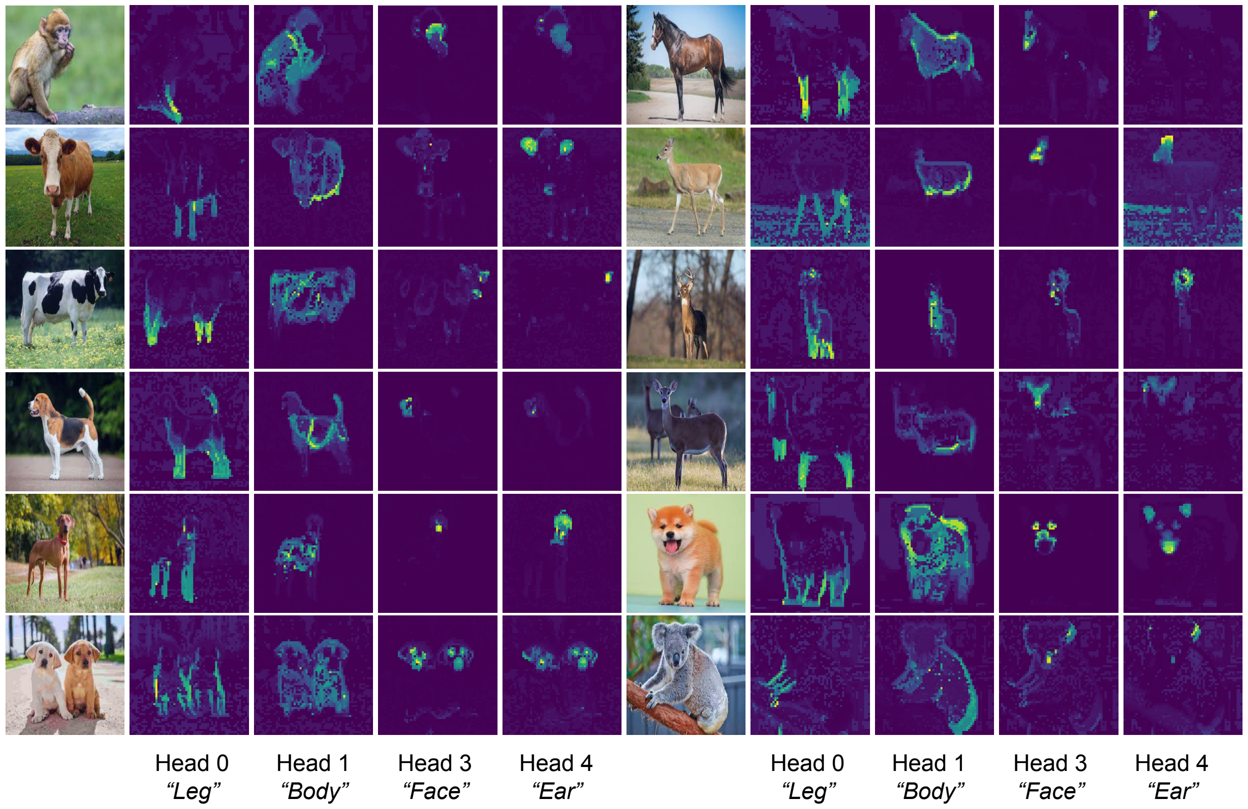


Figure 3.9: **Visualization of semantic heads.** We forward a mini-batch of images through a supervised CRATE and examine the attention maps from all the heads in the penultimate layer. We visualize a selection of attention heads to show that certain heads convey specific semantic meaning, i.e. *head 0* $\leftrightarrow$ *"Legs"*, *head 1* $\leftrightarrow$ *"Body"*, *head 3* $\leftrightarrow$ *"Face"*, *head 4* $\leftrightarrow$ *"Ear"*.

or unsupervised), CRATE is able to capture semantically meaningful boundaries of the main objects in an image. Compared to ViT, CRATE provides better internal representation tokens for both segmentation and detection.

3.3.3.3 Analysis of Segmentation in CRATE

Segmentation emerges through minimalistic design. Our empirical results demonstrate that self-supervised learning, as well as the specialized design options in DINO [32] (e.g., momentum encoder, student and teacher networks, self-distillation, etc.) are not necessary for the emergence of segmentation. We contend that an equally-promising approach to promote segmentation properties in transformer is to *design the transformer architecture with the structure of the input data in mind*. This finding of CRATE represents the *first supervised vision model with emergent segmentation properties*, and establishes white-box transformers as a promising direction for interpretable data-driven representation learning in foundation models.

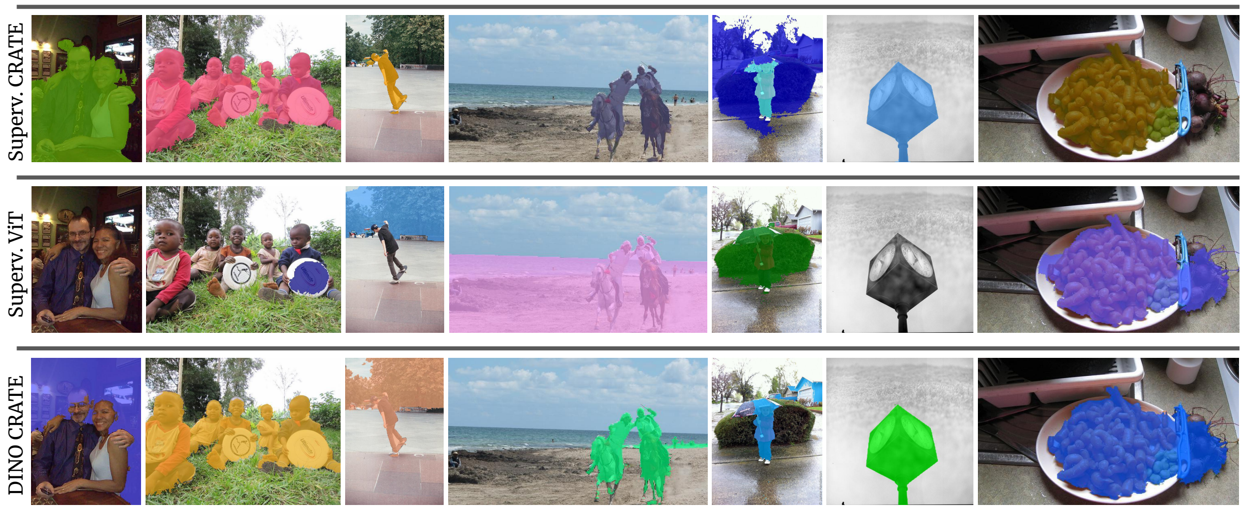


Figure 3.10: **Visualization of segmentation with MaskCut on COCO val2017 [166].** *Top and Bottom Rows*: CRATE architecture clearly detects main objects in the image when trained using either supervised classification or the DINO self-supervised technique [32]. *Middle Row*: Note that compared to CRATE, ViT trained via classification often fails to detect main objects in the images (columns 2, 3, 4).

Table 3.2: **Object detection and fine-grained segmentation via MaskCut on COCO val2017 [166]**. We consider models with different scales and evaluate the average precision measured by COCO's official evaluation metric. The first four models are pre-trained with image classification tasks under label supervision; the bottom three models are pre-trained via the DINO self-supervised technique [32]. CRATE conclusively performs better than the ViT at detection and segmentation metrics when both are trained using supervised classification.

		Detection			Segmentation		
Model	Train	AP_{50}	AP_{75}	AP	AP_{50}	AP_{75}	AP
CRATE-S/8	Supervised	2.9	1.0	1.1	1.8	0.7	0.8
CRATE-B/8	Supervised	2.9	1.0	1.3	2.2	0.7	1.0
ViT-S/8	Supervised	0.1	0.1	0.0	0.0	0.0	0.0
ViT-B/8	Supervised	0.8	0.2	0.4	0.7	0.5	0.4
CRATE-B/8	DINO	3.5	1.1	1.6	2.4	0.5	1.0
ViT-S/8	DINO	5.0	2.0	2.4	4.0	1.3	1.7
ViT-B/8	DINO	5.1	2.3	2.5	4.1	1.3	1.8

Identifying semantic properties of attention heads. We are interested in capturing the semantic meaning of certain attention *heads*; this is an important task for interpretability and is already studied for language transformers [188]. Intuitively, each head captures certain features of the data. Given a CRATE model, we first forward an input image (e.g. a horse image as in Chapter 3.9) and select four attention heads which seem to have semantic meaning by manual inspection. After identifying the attention heads, we visualize the self-attention map of these heads on other input images. Interestingly, we find that each of the selected attention heads captures a different part of the object, and even a different semantic meaning. For example, the attention head displayed in the first column of Chapter 3.9 captures the legs of different animals, and the attention head displayed in the last column captures the ears and head. This parsing of the visual input into a part-whole hierarchy has been a fundamental goal of learning-based recognition architectures since deformable part models [72, 73] and capsule networks [106, 213]—strikingly, it emerges from the white-box design of CRATE within our simple supervised training setup.

3.4 Conclusions and Future Work

In this work, we propose a new theoretical framework that allows us to derive deep transformer-like network architectures as incremental optimization schemes to learn compressed and sparse representation of the input data (or token sets). The so derived and learned deep architectures are not only fully mathematically interpretable, but also consistent on a layer-by-layer level with their design objective. Despite being arguably the simplest among all possible designs, these networks already demonstrate performance on large-scale real-world datasets and tasks close to seasoned transformers. We believe this work truly helps bridge the gap between theory and practice of deep neural networks as well as help unify seemingly separate approaches to learning and representing data distributions. Probably more importantly for practitioners, our framework provides theoretical guidelines to design and justify new, potentially more powerful, deep architectures for representation learning.

We believe this work helps *bridge the gap between theory and practice* of deep neural networks, as well as helps unify seemingly separate approaches to learning and representing data distributions through the perspective of data compression. These approaches include, but are not limited to: rate reduction, energy minimization, denoising-diffusion, and transformers. As we have seen, in these approaches, the role of each layer of the associated deep neural networks can be interpreted mathematically as operators to incrementally compress (denoise) and transform the data distribution according to its low-dimensional structures (modeled as Gaussian mixtures). To a large extent, our work suggests that a universal way to effectively and efficiently learn a data distribution with intrinsically low-dimensional structures in a high-dimensional space is through incremental compression, as already stated as a slogan by [270]:

We compress to learn, and we learn to compress.

This work and the previous work of [33] together strongly suggest that various deep neural networks are simply means to an end—compression. Different optimization schemes for compression are manifested as different network architectures and operators, e.g. LeNet, LISTA, ResNet, ReduNet, Transformers, which are now elucidated by CRATE.

From a practical standpoint, it is clear that this new framework can now provide us with theoretical guidelines to design and justify new, potentially more powerful, deep architectures for representation learning that are based on more advanced optimization techniques or strategies, instead of the basic gradient-based technique used to construct CRATE. Also, in this work, we have only used CRATE to learn deterministic encoding models. We believe that this framework can be extended to learn structured representations of the data distribution based on more advanced structured diffusion and denoising. This could potentially lead to more *controllable and consistent* generative methods, since improving the consistency and efficiency of existing diffusion-based generative models remains a challenging open problem [230].

Furthermore, as suggested by [52, 173], for any system to learn a low-dimensional data distribution *autonomously and continuously*, one needs to integrate the encoder and decoder not only as an autoencoder but as a *closed-loop transcription*:

$$\boldsymbol{x} \in \mathbb{R}^D \xrightarrow{f(\boldsymbol{x})} \boldsymbol{z} \in \mathbb{R}^d \xrightarrow{g(\boldsymbol{z})} \widehat{\boldsymbol{x}} \in \mathbb{R}^D \xrightarrow{f(\boldsymbol{x})} \widehat{\boldsymbol{z}} \in \mathbb{R}^d, \tag{3.19}$$

which allows the system to correct itself by minimizing the discrepancy between the internal representations $\boldsymbol{z}$ of the data and $\widehat{\boldsymbol{z}}$ of the generated approximation, without any external help (by human) to compare between the data $\boldsymbol{x}$ and $\widehat{\boldsymbol{x}}$. So far, the effectiveness of such a closed-loop transcription system has only been demonstrated with black-box deep architectures such as the ResNet [52, 246]. It would be possible now to build a closed-loop transcription with purely white-box encoder and decoder designs. This could ultimately enable us to develop *complete* learning systems that are capable of learning autonomously and continuously an internal data representation that is fully mathematically interpretable, controllable, and eventually self-consistent, or “aligned”, with the true external data distribution. The so-learned representation, just like our acquired memory, can support both discriminative and generative tasks, and serve both recognition and prediction purposes.

Part II

Privacy-preserving Representation Learning

Chapter 4

Reliable Federated Learning via NTK Representations

State-of-the-art federated learning methods can perform far worse than their centralized counterparts when clients have dissimilar data distributions. For neural networks, even when centralized SGD easily finds a solution that is simultaneously performant for all clients, current federated optimization methods fail to converge to a comparable solution. We show that this performance disparity can largely be attributed to optimization challenges presented by *nonconvexity*. Specifically, we find that the early layers of the network do learn useful features, but the final layers fail to make use of them. That is, federated optimization applied to this non-convex problem distorts the learning of the final layers. Leveraging this observation, we propose a ***Train**-**Convexify**-**Train*** (TCT) procedure to sidestep this issue: first, learn features using off-the-shelf methods (e.g., FedAvg); then, optimize a *convexified* problem obtained from the network's empirical neural tangent kernel approximation. Our technique yields accuracy improvements of up to $+36\%$ on FMNIST and $+37\%$ on CIFAR10 when clients have dissimilar data.

4.1 Introduction

Federated learning is a newly emerging paradigm for machine learning where multiple data holders (clients) collaborate to train a model on their combined dataset. Clients only share partially trained models and other statistics computed from their dataset, keeping their raw data local and private [176, 130]. By obviating the need for a third party to collect and store clients' data, federated learning has several advantages over the classical, centralized paradigm [56, 119, 82]: it ensures clients' consent is tied to the specific task at hand by requiring active participation of the clients in training, confers some basic level of privacy, and has the potential to make machine learning more participatory in general [148, 128]. Further, widespread legislation of data portability and privacy requirements (such as GDPR and CCPA) might even make federated learning a necessity [198].

Collaboration among clients is most attractive when clients have very different subsets of the combined dataset (*data heterogeneity*). For example, different autonomous driving companies may only be able to collect data in weather conditions specific to their location, whereas their vehicles would need to function under all conditions. In such a scenario, it would be mutually beneficial for companies in geographically diverse locations to collaborate and share data with each other. Further, in such settings, clients are physically separated and connected by ad-hoc networks with large latencies and limited bandwidth. This is especially true when clients are edge devices such as mobile phones, IoT sensors, etc. Thus, *communication efficiency* is crucial for practical federated learning. However, it is precisely under such circumstances (large data heterogeneity and low communication) that current algorithms fail dramatically [111, 158, 134, 207, 262, 3, 157, 5, 260, etc.]. This motivates our central question: *Why do current federated methods fail in the face of data heterogeneity—and how can we fix them?*

Our solution. We make two main observations: (i) We show that, even with data heterogeneity, linear models can be trained in a federated manner through gradient correction techniques such as SCAFFOLD [134]. While this observation is promising, it alone remains limited, as linear models are not rich enough to solve practical problems of interest (e.g., those that require feature learning). (ii) We shed light on why current federated algorithms struggle to train deep, nonconvex models. We observe that the failure of existing methods for neural networks is not uniform across the layers. The early layers of the network do in fact learn useful features, but the final layers fail to make use of them. Specifically, federated optimization applied to this nonconvex problem results in distorted final layers.

These observations suggest a *train-convexify-train* federated algorithm, which we call *TCT*: first, use any off-the-shelf federated algorithm [such as FedAvg, 176] to train a deep model to extract useful features; then, compute a convex approximation of the deep model using its empirical Neural Tangent Kernel (eNTK) [122, 152, 74, 169, 267], and use gradient correction methods such as SCAFFOLD to train the final model. Effectively, the second-stage features freeze the features learned in the first stage and fit a linear model over them. We show that this simple strategy is highly performant on a variety of tasks and models—we obtain accuracy gains up to 36% points on FMNIST with a CNN, 37% points on CIFAR10 with ResNet18-GN, and 16% points on CIFAR100 with ResNet18-GN. Further, its convergence remains unaffected even by extreme data heterogeneity. Finally, we show that given a pre-trained model, our method completely closes the gap between centralized and federated methods.

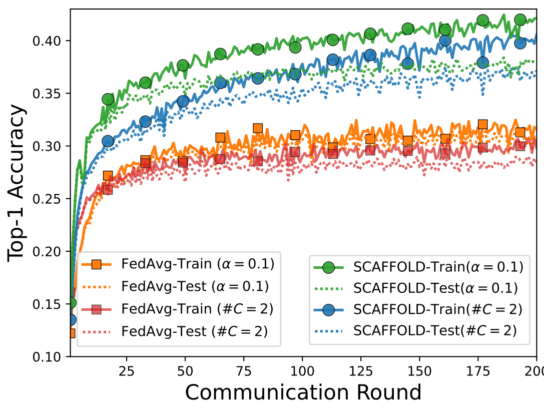
4.1.1 Related Work

Federated learning. There are two main motivating scenarios for federated learning (FL). The first is where internet service companies (e.g., Google, Facebook, Apple, etc.) want to train machine learning models over their users' data, but do not want to transmit raw personalized data away from user devices [204, 22]. This is the setting of *cross-device* federated learning and is characterized by an extremely large number of unreliable clients, each of whom has very little data and the collections of data are assumed to be homogeneous [130, 23, 133, 22]. The second motivating scenario is when valuable data is split across different organizations, each of whom is either protected by privacy regulation or is simply unwilling to share their raw data. Such "data islands" are common among hospital networks, financial institutions, autonomous-vehicle companies, etc. This is known as *cross-silo* federated learning and is characterized by a few highly reliable clients, who potentially have extremely diverse data. In this work, we focus on the latter scenario.

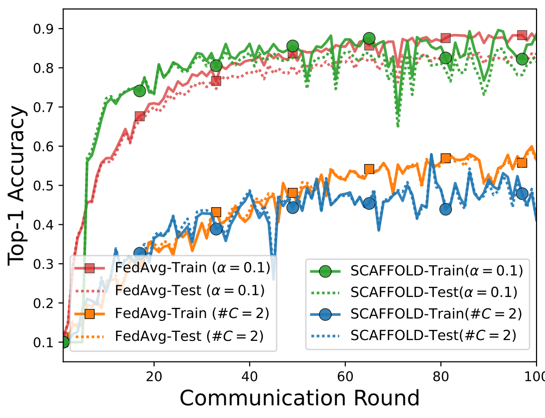
Metrics in FL. FL research considers numerous metrics, such as fairness across users [182, 159, 221], formal security and privacy guarantees [24, 204, 75, 183], robustness to corrupted agents and corrupted training data [19, 225, 69, 132, 101], preventing backdoors at test time [11, 238, 259, 171], etc. While these concerns are important, the main goal of FL (and our work) is to achieve high accuracy with minimal communication [176]. Clients are typically geographically separated yet need to communicate large deep learning models over unoptimized ad-hoc networks [130]. Finally, we focus on the setting where all users are interested in training the same model over the combined dataset. This is in contrast to model-agnostic protocols [165, 192, 5] or personalized federated learning [60, 68, 273, 48, 147, 36]. Finally, we focus on minimizing the number of rounds required. Our approach can be combined with communication compression, which reduces bits sent per round [239, 6, 88, 235].

Federated optimization. Algorithms for FL proceed in rounds. In each round, the server sends a model to the clients, who partially train this model using their local compute and data. The clients send these partially trained models back to the server who then aggregates them, finishing a round. FedAvg [176], which is the de facto standard FL algorithm, uses SGD to perform local updates on the clients and aggregates the client models by simply averaging their parameters. Unfortunately, however, FedAvg has been observed to perform poorly when faced with data heterogeneity across the clients [111, 158, 134, 207, 262, 3, 157, 5, 260, 242, etc.]. Theoretical investigations of this phenomenon [134, 268] showed that this was a result of *gradient heterogeneity* across the clients. Consider FedAvg initialized with the globally optimal model. If this model is not also optimal for each of the clients as well, the local updates will push it away from the global optimum. Thus, convergence would require a careful tuning of hyper-parameters. To overcome this issue, SCAFFOLD [134] and FedDyn [3] propose to use control variates to correct for the biases of the individual clients akin to variance reduction [126, 57]. This *gradient correction* is applied in every local update by the client and provably nullifies the effect of gradient heterogeneity [134, 181, 36]. However, as we show here, such methods are insufficient to overcome high data heterogeneity especially for deep learning. Other, more heuristic approaches to combat gradient heterogeneity include using a regularizer [158] and sophisticated server aggregation strategies such as momentum [112, 263, 164] or adaptivity [207, 133, 35].

A second line of work pins the blame on performance loss due to averaging nonconvex models. To overcome this, [224, 288] propose to learn a mapping between the weights of the client models before averaging, [5] advocates a functional perspective and replaces the averaging step with knowledge distillation, and [265, 157, 241] attempt to align the internal representations of the client models. However, averaging is unlikely to be the only culprit since FedAvg does succeed under low heterogeneity, and averaging nonconvex models can lead to improved performance [121, 269].



(a) Using linear model.



(b) Using ResNet-18.

Figure 4.1: Performance of FedAvg and SCAFFOLD on CIFAR10 when data are split among ten clients in two ways (#C=2 and α=0.1). The #C=2 split is more non-i.i.d. than the α=0.1 split. For convex problems (left), gradient correction methods such as SCAFFOLD are relatively unaffected by data heterogeneity, and consistently outperform FedAvg. However, for nonconvex problems (right), FedAvg and SCAFFOLD perform very similarly and both are strongly negatively affected by data heterogeneity.

Neural Tangent Kernels (NTK) and neural network linearization. NTK was first proposed to analyze the limiting behavior of infinitely wide networks [122, 152]. While NTK with MSE may be a bad approximation of real-world finite networks in general [78], it approximates the fine-tuning of a pre-trained network well [184], especially with some minor modifications [4]. That is, NTK cannot capture feature learning but does capture how a model utilizes learnt features better than last/mid layer activations.

4.2 The Effect of Nonconvexity in Federated Learning

In this section, we investigate the poor performance of FedAvg [176] and SCAFFOLD [134] empirically in the setting of deep neural networks, focusing on image classification with a ResNet-18. To construct our federated learning setup, we split the CIFAR-10 dataset in a highly heterogeneous manner among ten clients. We either assign each client two classes (denoted by #C=2) or distribute samples according to a Dirichlet distribution with $\alpha = 0.1$ (denoted by α=0.1). For more details, see Section 4.4.1.

Insufficiency of gradient correction methods. Current theoretical work [e.g., 134, 207, 3, 261] attributes the slowdown from data heterogeneity to the individual clients having varying local optima. If no single model is simultaneously optimal for all clients, then the updates of different clients can compete with and distort each other, leading to slow convergence. This tension is captured by the variance of the updates across the clients [client gradient heterogeneity, see 260].

Table 4.1: Feature learning by FedAvg. We report the test accuracy of a ResNet-18 after (centralized) retraining of the last ℓ layers on CIFAR10. The earlier $(7-\ell)$ layers are frozen to either random initialization or to the weights of a FedAvg-trained model. The difference measures utility of the $(7-\ell)$ layers learnt by FedAvg. The baseline FedAvg model without additional training gets 56.9% accuracy. We see that all layers of the FedAvg model contain useful information.

Layers retrained	Accuracy (%) Random init	Accuracy (%) FedAvg init	Improvement (%) (FedAvg - Random)
1/7 last layer	35.37	77.93	42.56
2/7 last layers	67.33	87.04	19.71
3/7 last layers	80.18	89.28	9.10
4/7 last layers	88.03	90.57	2.54
5/7 last layers	91.34	91.61	0.27
6/7 last layers	91.78	91.91	0.13

Gradient correction methods such as SCAFFOLD [134] and FedDyn [3] explicitly correct for this and are provably unaffected by gradient heterogeneity for both convex and nonconvex losses.

These theoretical predictions are aligned with the results of Figure 4.1a, where the loss landscape is convex: SCAFFOLD is relatively unaffected by the level of heterogeneity and consistently outperforms FedAvg. In particular, performance is largely dictated by the algorithm and not the data distributions. This shows that client gradient heterogeneity captures the difficulty of the problem well. On the other hand, when training a ResNet-18 model with nonconvex loss landscape, Figure 4.1b shows that both FedAvg and SCAFFOLD suffer from data heterogeneity. This is despite the theory of gradient correction applying to both convex and nonconvex losses. Further, the train and test accuracies in Figure 4.1b match quite closely, suggesting that the failure lies in optimization (not fitting the training data) rather than generalization. Thus, while the current theory makes no qualitative distinctions between convex and nonconvex convergence, the practical behavior of algorithms in these settings is very different. Such differences between theoretical predictions and practical reality suggests that black-box notions such as gradient heterogeneity are insufficient for capturing the difficulty of training deep models.

Ease of feature learning. We now dive into how a ResNet-18 trained with FedAvg (56.9% accuracy) differs from the centralized baseline (91.9% accuracy). We first apply linear probing to the FedAvg model (i.e., retraining with all but the output layer frozen). Note that this is equivalent to (convex) logistic regression over the last-layer activations. This simple procedure produces a striking jump from 56.9% to 77.9% accuracy. Thus, of the 35% gap in accuracy between the FedAvg and centralized models, 21% may be attributed to a failure to optimize the linear output layer. We next extend this experiment towards probing the information content of other layers.

Given a FedAvg-trained model, we can use centralized training to retrain only the last ℓ layers while keeping the rest of the $(7-\ell)$ layers (or ResNet blocks) frozen. We can also perform this procedure starting from a randomly initialized model. The performance difference between these two models can be attributed to the information content of the frozen $(7 - \ell)$ layers of the FedAvg model. Table 4.1 summarizes the results of this experiment. The large difference in accuracy (up to 42.6%) indicates the initial layers of the FedAvg model have learned useful features. There continues to be a gap between the FedAvg features and random features in the earlier layers as well,[1] meaning that all layers of the FedAvg model learn useful features. We conjecture this is because from the perspective of earlier layers which perform simple edge detection, the tasks are independent of labels and the clients are i.i.d. However, the higher layers are more specialized and the effect of the heterogeneity is stronger.

4.3 Method

Based on the observations in Section 4.2, we propose *train-convexify-train* (TCT) as a method for overcoming data heterogeneity when training deep models in a federated setting. Our high-level intuition is that we want to leverage both the features learned from applying FedAvg to neural networks and the effectiveness of *convex* federated optimization. More specifically, we perform several rounds of "*bootstrap*" FedAvg to learn features before solving a convexified version of the original optimization problem.

4.3.1 Computing the Empirical Neural Tangent Kernel

To sidestep the challenges presented by nonconvexity, we describe how we approximate a neural network by its "linearization." Given a neural network $f(\,\cdot\,;\theta_0)$ with weights $\theta_0 \in \mathbb{R}^P$ mapping inputs $x \in \mathbb{R}^D$ to $\mathbb{R}^C$, we replace it by its *empirical neural tangent kernel (eNTK)* approximation at θ_0 given by

$$f(x;\theta) \approx f(x;\theta_0) + (\theta - \theta_0)^\top \frac{\partial}{\partial\theta} f(x;\theta_0),$$

at each $x \in \mathbb{R}^D$. Under this approximation, $f(x;\theta)$ is a linear function of the "feature vector" $(f(x;\theta_0), \frac{\partial}{\partial\theta} f(x;\theta_0))$ and the original nonconvex optimization problem becomes (convex) linear regression with respect to these features.[2] Leveraging NTK for solving federated optimization problems has also been studied in previous work [115, 294].

To reduce the computational burden of working with the eNTK approximation, we make two further approximations: First, we randomly reinitialize the last layer of θ_0 and only consider $\frac{\partial}{\partial\theta} f(x;\theta_0)$ with respect to a single output logit. Over the randomness of this reinitialization, $\mathbb{E}[f(x;\theta_0)] = 0$. Moreover, given the random reinitialization, all the output logits of $f(x;\theta_0)$ are symmetric. These observations mean each data point x can be represented by a P-dimensional feature vector $\frac{\partial}{\partial\theta} f_1(x;\theta_0)$, where $f_1(\,\cdot\,;\theta_0)$ refers to the first output logit. Then, we apply a dimen-

[1]The significant decrease in the gap as we go down the layers may be because of the skip connections in the lower ResNet blocks which allow the random frozen layers to be sidestepped. This underestimates the true utility and information content in the earlier FedAvg layers.

[2]For classification problems, we one-hot encoded labels and fit a linear model using squared loss.

sionality reduction by subsampling p random coordinates from this P-dimensional featurization.[3] In our setting, this sub-sampling has the added benefit of reducing the number of bits communicated per round.

In summary, we transform our original (nonconvex) optimization problem over a neural network initialized at θ_0 into a convex optimization problem in three steps: (i) reinitialize the last layer of θ_0; (ii) for each data point x, compute the gradient $\phi_{\mathrm{eNTK}}(x;\theta_0) := \frac{\partial}{\partial\theta} f_1(x;\theta_0)$; (iii) subsample the coordinates of $\phi_{\mathrm{eNTK}}(x;\theta_0)$ for each x to obtain a reduced-dimensionality eNTK representation. Let $\mathscr{S}\colon \mathbb{R}^P \to \mathbb{R}^p$ denote this subsampling operation. Finally, we solve the resulting linear regression problem over these eNTK representations.[4]

4.3.2 Convexifying Federated Learning via eNTK Representations

The eNTK approximation lets us convexify the neural net optimization problem: following Section 4.3.1, we may extract (from a model trained with FedAvg) eNTK representations of inputs from each client. It remains to fit an overparameterized linear model using these eNTK features in a federated manner. For ease of presentation, we denote the subsampled eNTK representation of input x by $z \in \mathbb{R}^p$, where p is the eNTK feature dimension after subsampling. We use z_i^k to represent the eNTK feature of the i-th sample from the k-th client. Then, for K the number of clients, Y_i^k the one-hot encoded labels, n_k the number of data points of the k-th client, $n := \sum_{k\in[K]} n_k$ the number of data points across all clients, and $p_k := n_k/n$, we can approximate the nonconvex neural net optimization problem by the convex linear regression problem

$$\min_W L(W) := \sum_{k=1}^{K} p_k \cdot L_k(W), \qquad \text{where} \quad L_k(W) := \frac{1}{n_k} \sum_{i=1}^{n_k} \|W^\top z_i^k - Y_i^k\|_2^2. \tag{4.1}$$

To obtain the eNTK representation z of an input x, we take θ_0 in Section 4.3.1 to be the weights of a model trained with FedAvg. As we will show in Section 4.4, the convex reformulation in Eq. (4.1) significantly reduces the number of communication rounds needed to find an optimal solution.

4.3.3 Train-Convexify-Train (TCT)

We now present our algorithm train-convexify-train (TCT), with convexification done via the neural tangent kernel, for federated optimization.

[3]That such representations empirically have low effective dimension due to fast eigenvalue decay [see, e.g., 267] means that such a random projection approximately preserves the geometry of the data points [10, 295]. For all of our experiments, we set $p = 100,000$.

[4]Given a fitted linear model with weights $W \in \mathbb{R}^{p\times C}$, the prediction at x is $\arg\max_j [W^\top \mathscr{S}(\phi_{\mathrm{eNTK}}(x))]_j$.

TCT — train-convexify-train with eNTK representations

- **Stage 1:** *Extract eNTK features from a FedAvg-trained model.* FedAvg is first used to train the model for T_1 communication rounds. Let θ_{T_1} denote the model weights after these T_1 rounds. Then, each client locally computes subsampled eNTK features, i.e., $z_i^k = \mathscr{S}(\phi_{\text{eNTK}}(x_i^k; \theta_{T_1}))$ for $k \in [K]$ and $i \in [n_k]$.
- **Stage 2:** *Decentralized linear regression with gradient correction.* Given samples $\{(z_i^k, Y_i^k)\}_{i=1}^{n_k}$ on each client k, first normalize the eNTK inputs of all clients with a single communication round.[a] Then, solve the linear regression problem defined in Eq. (4.1) by SCAFFOLD with local learning rate η and local steps M.[b]

[a]For every feature in the eNTK representation, subtract the mean and scale to unit variance.

[b]The detailed description of SCAFFOLD for solving linear regression problems can be found in Algorithm 1, Appendix C.1. It has the same communication and computation cost as FedAvg.

To motivate TCT, recall that in Section 4.2 we found that FedAvg learns "useful" features despite its poor performance, especially in the earlier layers. By taking an eNTK approximation, TCT optimizes a convex approximation while using information from *all* layers of the model. Empirically, we find that these extracted eNTK features significantly reduce the number of communication rounds needed to learn a performant model, even with data heterogeneity.

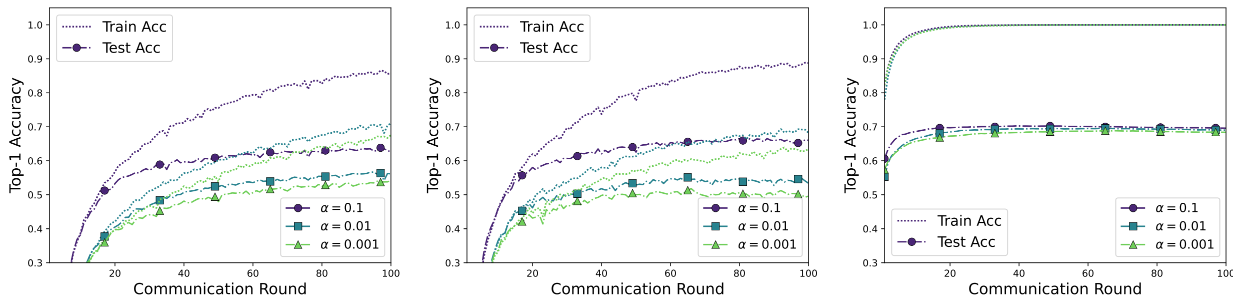
4.4 Experiments

We now study the performance of TCT for the decentralized training of deep neural networks in the presence of data heterogeneity. We compare TCT to state-of-the-art federated learning algorithms on three benchmark tasks in federated learning. For each task, we apply these algorithms on client data distributions with varying degrees of data heterogeneity. We find that our proposed approach significantly outperforms existing algorithms when clients have highly heterogeneous data across all tasks. For additional experimental results and implementation details, see Appendix C.2. Our code is available at `https://github.com/yaodongyu/TCT`.

4.4.1 Experimental Setup

Datasets and degrees of data heterogeneity. We assess the performance of federated learning algorithms on the image classification tasks FMNIST [277], CIFAR10, and CIFAR100 [144]. FMNIST and CIFAR10 each consist of 10 classes, while CIFAR100 includes images from 100 classes. There are 60,000 training images in FMNIST, and 50,000 training images in CIFAR10/100.

To vary the degree of data heterogeneity, we follow the setup of [156]. We consider two types of non-i.i.d. data distribution: *(i) Data heterogeneity sampled from a symmetric Dirichlet distribution with parameter* α [165, 262]. That is, we sample $p_c \sim \text{Dir}_K(\alpha)$ from a K-dimensional symmetric Dirichlet distribution and assign a p_c^k-fraction of the class c samples to client k. (Smaller



(a) FedAvg. (b) SCAFFOLD. (c) TCT.

Figure 4.2: Training/test accuracy vs. communication round for FedAvg (left), SCAFFOLD (middle), and our algorithm TCT (right) on the CIFAR100 dataset with various degrees of non-iid-ness ($\mathrm{Dir}_K(\alpha)$ with $\alpha \in \{0.1, 0.01, 0.001\}$). Dotted lines represent the training accuracy, and dashdot lines with markers represent the test accuracy.

α corresponds to more heterogeneity.) *(ii) Clients get samples from a fixed subset of classes* [176]. That is, each client is allocated a subset of classes; then, the samples of each class are split into non-overlapping subsets and assigned to clients that were allocated this class. We use `#C` to denote the number of classes allocated to each client. For example, `#C`$=2$ means each client has samples from 2 classes. To allow for consistent comparisons, all of our experiments are run with 10 clients.

Models. For FMNIST, we use a convolutional neural network with ReLU activations consisting of two convolutional layers with max pooling followed by two fully connected layers (SimpleCNN). For CIFAR10 and CIFAR100, we mainly consider an 18-layer residual network [100] with 4 basic residual blocks (ResNet-18). In Appendix C.2.2, we present experimental results for other architectures.

Algorithms and training schemes. We compare TCT to state-of-the-art federated learning algorithms, focusing on the widely-used algorithms FedAvg [176], FedProx [158], and SCAFFOLD [134]. (For comparisons to additional algorithms, see Appendix C.2.1.) Each client uses SGD with weight decay 10^{-5} and batch size 64 by default. For each baseline method, we run it for 200 total communication rounds using 5 local training epochs with local learning rate selected from $\{0.1, 0.01, 0.001\}$ by grid search. For TCT, we run 100 rounds of FedAvg in Stage 1 following the above and use 100 communication rounds in Stage 2 with $M = 500$ local steps and local learning rate $\eta = 5 \cdot 10^{-5}$.

4.4.2 Main Results

Table 4.2 displays the top-1 accuracy of all algorithm on the three tasks with varying degrees of data heterogeneity. We evaluated each algorithms on each task under four degrees of data heterogeneity. Smaller `#C` and α in Table 4.2 correspond to higher heterogeneity.

We find that the existing federated algorithms all suffer when data heterogeneity is high across all three tasks. For example, the top-1 accuracy of FedAvg on CIFAR-10 is 56.86% when `#C`$=2$,

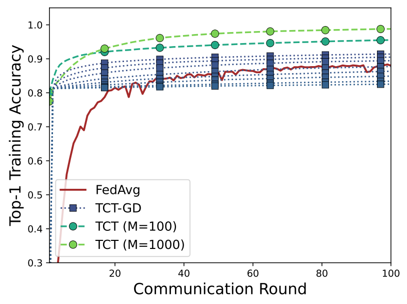
Table 4.2: The top-1 accuracy $(\%)$ of our algorithm (TCT) vs. state-of-the-art federated learning algorithms evaluated on FMNIST, CIFAR10, and CIFAR100. We vary the degree of data heterogeneity by controlling the α parameter of the symmetric Dirichlet distribution $\mathrm{Dir}_K(\alpha)$ and the `#C` parameter for assigning how many labels each client owns. Higher accuracy is better. The highest top-1 accuracy in each setting is highlighted in **bold**.

Datasets	**Architectures**	**Methods**	**Non-i.i.d. degree**			
			$\#\mathtt{C}=1$	$\#\mathtt{C}=2$	$\alpha=0.1$	$\alpha=0.5$
FMNIST	SimpleCNN	FedAvg	35.10%	85.18%	86.18%	90.09%
		FedProx	50.04%	84.91%	86.31%	89.77%
		SCAFFOLD	12.80%	42.80%	83.87%	89.40%
		TCT	**86.32%**	**90.33%**	**90.78%**	**91.13%**
		Centralized	91.40%			
			$\#\mathtt{C}=1$	$\#\mathtt{C}=2$	$\alpha=0.1$	$\alpha=0.5$
CIFAR-10	ResNet-18	FedAvg	11.27%	56.86%	82.60%	90.43%
		FedProx	12.30%	56.87%	83.31%	90.68%
		SCAFFOLD	10.00%	46.75%	80.46%	90.72%
		TCT	**49.92%**	**83.02%**	**89.21%**	**91.10%**
		Centralized	91.90%			
			$\alpha=0.001$	$\alpha=0.01$	$\alpha=0.1$	$\alpha=0.5$
CIFAR-100	ResNet-18	FedAvg	53.89%	54.22%	63.49%	67.65%
		FedProx	52.87%	54.32%	63.47%	67.54%
		SCAFFOLD	49.86%	54.07%	65.67%	**71.07%**
		TCT	**68.42%**	**69.07%**	**69.66%**	69.68%
		Centralized	73.61%			

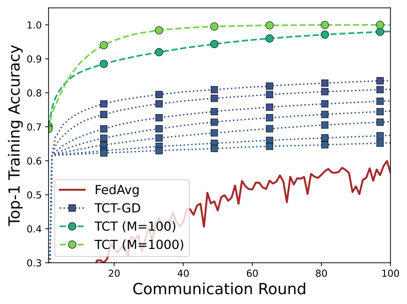
which is much worse than the 90.43% achieved in a more homogeneous setting (e.g. $\alpha = 0.5$). In contrast, TCT achieves consistently strong performance, even in the face of high data heterogeneity. More specifically, TCT achieves the best top-1 accuracy performance across all settings except CIFAR-100 with $\alpha = 0.5$, where TCT does only slightly worse than SCAFFOLD.

In absolute terms, we find that TCT is not affected much by data heterogeneity, with performance dropping by less than 1.5% on CIFAR100 as α goes from 0.5 to 0.001. Moreover, our algorithm improves over existing methods by at least 15% in the challenging cases, including FMNIST with `#C=1`, CIFAR-10 with `#C=1` and `#C=2`, and CIFAR-100 with $\alpha = 0.01$ and $\alpha = 0.001$. And, perhaps surprisingly, our algorithm still performs relatively well in the extreme non-i.i.d. setting where each client sees only a single class.

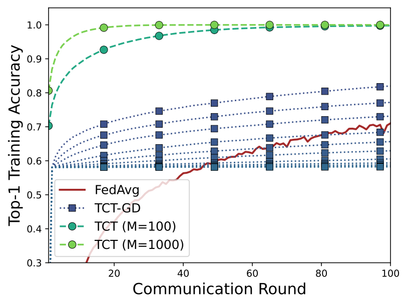
Figure 4.2 compares the performances of FedAvg, SCAFFOLD, and TCT in more detail on CIFAR100 dataset with different degrees of data heterogeneity. We consider the Dirichlet distribution with parameter $\alpha \in \{0.1, 0.01, 0.001\}$ and compare the training and test accuracy of these



(a) FMNIST.



(b) CIFAR10.



(c) CIFAR100.

Figure 4.3: Training accuracy vs. communication round for full batch gradient descent (GD) and TCT on FMNIST-[#C=2] **(a)**, CIFAR10-[#C=2] **(b)**, and CIFAR100-[$\alpha = 0.01$] **(c)**. Each dotted line with square markers represents the training accuracy of GD with some learning rate. Dashed lines with circle markers represent the training accuracy of TCT with different numbers of local steps. We also include the training accuracy results of FedAvg with learning rate $\eta = 0.1$. We use TCT-GD to denote the variant of TCT which replaces SCAFFOLD with GD in Stage 2.

three algorithms. As shown in Figures 4.2a and 4.2b, both FedAvg and SCAFFOLD struggle when data heterogeneity is high: for both algorithms, test accuracy drops significantly when α decreases. In contrast, we see from Figure 4.2c that TCT maintains almost the same test accuracy for different α. Furthermore, the same set of default parameters for our algorithm, including local learning rate and the number of local steps, is relatively robust to different levels of data heterogeneity.

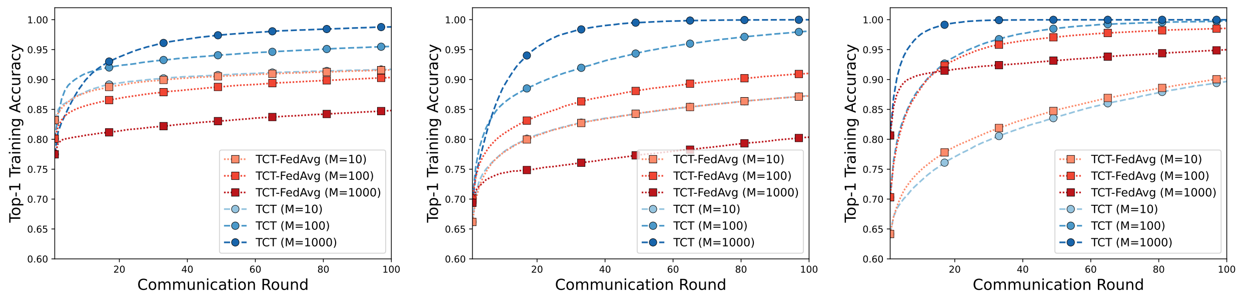
4.4.3 Communication Efficiency

To understand the effectiveness of the local steps in our algorithm, we compare SCAFFOLD (used in TCT-Stage 2) to full batch gradient descent (GD) applied to the overparameterized linear regression problem in Stage 2 of TCT on these datasets. For our algorithm, we set local steps $M \in \{10^2, 10^3\}$ and use the default local learning rate. For full batch GD, we vary the learning rate from 10^{-5} to 10^{-1} and visualize the ones that do not diverge.

The results are summarized in Figure 4.3. Each dotted line with square markers in Figure 4.3 corresponds to full batch GD with some learning rate. Across all three datasets, our proposed algorithm consistently outperforms full batch GD. Meanwhile, we find that more local steps for our algorithms lead to faster convergence across all settings. In particular, our algorithm converges within 20 communication rounds on CIFAR100 (as shown in Figure 4.3c). These results suggest that our proposed algorithm can largely leverage the local computation and improve communication efficiency.

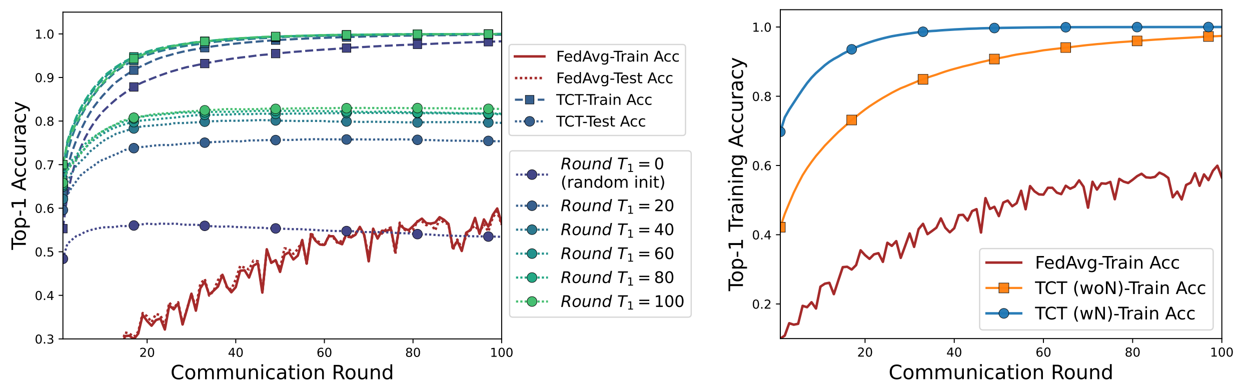
4.4.4 Ablations

Gradient correction. We investigate the role of gradient correction when solving overparameterized linear regression with eNTK features in TCT. We compare SCAFFOLD (used in TCT) to FedAvg on solving the regression problems and summarize the results in Figure 4.4. We use the



(a) FMNIST. (b) CIFAR10. (c) CIFAR100.

Figure 4.4: Comparing TCT to TCT-FedAvg for solving the overparameterized linear regression problem on **(a)** FMNIST-[#C=2], **(b)** CIFAR10-[#C=2], and **(c)** CIFAR100-[$\alpha = 0.01$]. We use TCT-FedAvg to denote a variant of TCT that uses FedAvg instead of SCAFFOLD to perform linear regression in TCT-Stage 2. Dotted red lines with square markers represent the training accuracy of TCT-FedAvg with different numbers of local steps. Dashed blue red lines with circle markers represent the training accuracy of TCT with different numbers of local steps. A darker color means more local steps.



(a) Effect of FedAvg communication rounds in Stage 1. (b) Effect of normalization.

Figure 4.5: **(a).** We evaluate TCT on using checkpoints save at different communication rounds T_1 in Stage 1. $T_1 = 0$ corresponds to the randmon initialized model weights scenario (without FedAvg training). Dash lines with square markers represent the training accuracy, and dotted lines with circle makers represent the test accuracy. **(b).** We study the effect of pre-conditioning on TCT. TCT (wN) corresponds to the setting where eNTK features are normalized, and TCT (woN) corresponds to the without normalization step setting.

default local learning rate and consider three different numbers of local steps for both algorithms, i.e., $M \in \{10, 100, 1000\}$. As shown in Figure 4.4, our approach largely outperforms FedAvg when the number of local steps is large ($M \geq 100$) across three datasets. We also find that the performance of FedAvg can even degrade when the number of local steps increases. For example, FedAvg with $M = 1000$ performs the worst across all three datasets. In contrast to FedAvg, SCAF-

FOLD converges faster when the number of local steps increases. These observations highlight the importance of gradient correction in our algorithm.

Model weights for computing eNTK features. To understand the impact of the model weights trained in Stage 1 of TCT, we evaluate TCT run with different T_1 parameters. We consider $T_1 \in \{0, 20, 40, 60, 80, 100\}$, where $T_1 = 0$ corresponds to randomly initialized weights. From Figure 4.5a, we find that weights after FedAvg training are much more effective than weights at random initialization. Specifically, without FedAvg training, the eNTK (at random initialization) performs worse than standard FedAvg. In contrast, TCT significantly outperforms FedAvg by a large margin (roughly 20% in test accuracy) when eNTK features are extracted from a FedAvg-trained model. Also, we find that TCT is stable with respect to the choice of communication rounds T_1 in Stage 1. For example, models trained by TCT with $T_1 \geq 60$ achieve similar performance.

Effect of normalization. In Figure 4.5b, we investigate the role of normalization on TCT by comparing TCT run with normalized and unnormalized eNTK features. The same number of local steps ($M = 500$) is applied for both settings. We tune the learning rate η for each setting and plot the run that performs best (as measured in training accuracy). The results in Figure 4.5b suggest that the normalization step in TCT significantly improves the communication efficiency by increasing convergence speed. In particular, TCT with normalization converges to nearly 100% training accuracy in approximately 40 communication rounds, which is much faster than TCT without normalization.

Pre-training vs. Bootstrapping. In Appendix C.2.4, we explore the effect of starting from a pre-trained model instead of relying on bootstrapping to learn the features. We find that pre-training further improves the performance of TCT and completely erases the gap between centralized and federated learning.

Additionally, we conduct experiments on investigating the role of training loss function and subsampling approximation in TCT-Stage 2. For TCT-Stage 2, we find that neither using the cross-entropy loss as the training objective nor applying full eNTK representations significantly improves the performance of TCT. On the other hand, applying subsampling approximation in TCT-Stage 2 can largely improve the communication efficiency compared to the full eNTK representations approach. See Appendix C.2.7 for detailed experimental results.

4.5 Conclusion

We have argued that nonconvexity poses a significant challenge for federated learning algorithms. We found that a neural network trained in such a manner does learn useful features, but fails to use them and thus has poor overall accuracy. To sidestep this issue, we proposed a *train-convexify-train* procedure: first, train the neural network using FedAvg; then, optimize (using SCAFFOLD) a convex approximation of the model obtained using its empirical neural tangent kernel. We showed that the first stage extracts meaningful features, whereas the second stage learns to utilize these features to obtain a highly performant model. The resulting algorithm is significantly faster and more stable to hyper-parameters than previous federated learning methods. Finally, we also showed that given a good pre-pretrained feature extractor, our convexify-train procedure fully closes the gap between centralized and federated learning.

Our algorithm adds to the growing body of work using eNTK to *linearize* neural networks and obtain tractable convex approximations. However, unlike most of these past works which only work with pre-trained models, our bootstrapping allows training models from scratch. Finally, we stress that the success of our approach underscores the need to revisit theoretical understanding of heterogeneous federated learning. Nonconvexity seems to play an outsized role but its effect in FL has hitherto been unexplored. In particular, black-box notions of difficulty such as gradient dissimilarity or distances between client optima seem insufficient to capture practical performance. It is likely that further progress in the field (e.g. federated pre-training of foundational models), will require tackling the issue of nonconvexity head on.

Chapter 5

Differentially Private Representation Learning

Artificial intelligence (AI) has seen a tremendous surge in capabilities thanks to the use of *foundation models* trained on internet-scale data. On the flip side, the uncurated nature of internet-scale data also poses significant privacy and legal risks, as they often contain personal information or copyrighted material that should not be trained on without permission. In this work, we propose as a mitigation measure a recipe to train foundation vision models via self-supervised learning with differential privacy (DP) guarantee. We identify masked autoencoders as a suitable learning algorithm that aligns well with DP-SGD, and train *ViP*—a **Vi**sion transformer with differential **P**rivacy—under a strict privacy budget of $\epsilon = 8$ on the LAION400M dataset. We evaluate the quality of representation learned by ViP using standard downstream vision tasks; in particular, ViP achieves a (non-private) linear probing accuracy of 55.7% on ImageNet, comparable to that of end-to-end trained AlexNet (trained and evaluated on ImageNet). Our result suggests that scaling to internet-scale data can be practical for private learning.

5.1 Introduction

Foundation models (*e.g.*, GPT-3, SimCLR, CLIP, *etc.* [26, 39, 201]) pre-trained on vast amounts of diverse unlabeled data through self-supervised learning (SSL) have emerged as an important building block for artificial intelligence (AI) systems [20]. These foundation models enable downstream applications via fine-tuning, prompting, or training a simpler model on top of the learned representations to perform more specialized tasks, and have performed tremendously well on challenging benchmarks in both language and vision domains [26, 201, 247].

Despite the widespread deployment of foundation models, there are significant privacy and legal risks of training these models on uncurated data that often contain personal information or copyrighted material. Although the training data for these models are considered *public* in most cases, some of the data may be sensitive; additionally, there are certain privacy and copyright laws that apply to model training even on such *public* data [102]. In addition, studies have shown that

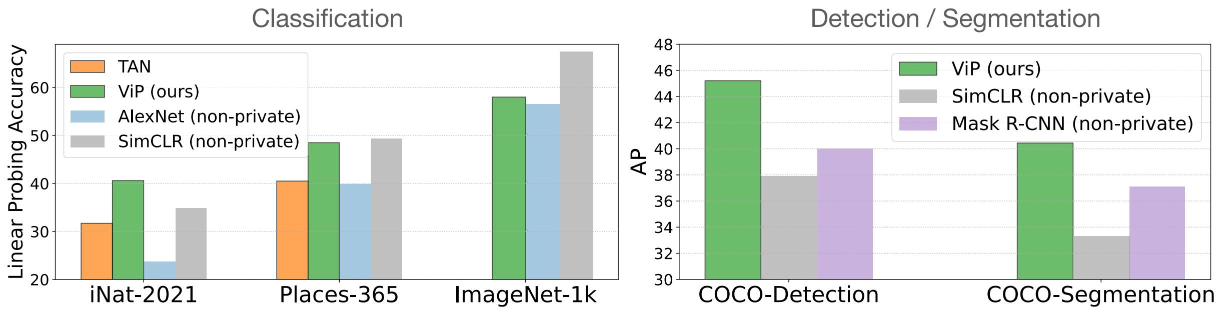


Figure 5.1: (**left**) Linear probing accuracies of TAN [215] (state-of-the-art DP training method), AlexNet [145], SimCLR [39] and ViP—our DP-trained model with $\epsilon = 8$. ViP can achieve similar transfer learning result as SimCLR on iNat-2021 and Places-365, and achieves similar accuracy on ImageNet as end-to-end trained AlexNet. (**right**) Average precision (AP) evaluations of SimCLR [39], Mask R-CNN [98] and ViP on MS-COCO. Our DP-trained model outperforms both SimCLR and Mask R-CNN.

generative foundation models such as GPT-3 can sometimes regurgitate memorized information about individuals and licensed content from its training data when prompted to do so [31]. More recently, [177] showed that non-generative vision SSL models can also be probed to reveal sensitive information about individual samples in its training data when given partial information.

Given these risks, there is an urgent need to train foundation models that can adhere to relevant privacy and copyright laws. To this end, differential privacy (DP; [64]) seeks to limit the influence of individual training data points on the trained model, and hence has the potential to mitigate both privacy and copyright risks for sensitive information that is confined to a single or a few training examples [102]. For any model that can be trained using gradient-based optimization, DP-SGD [229, 1] can be applied instead to ensure that the trained model satisfies the rigorous definition of DP. However, there are still significant technical challenges in DP-SGD training of large-scale foundation vision models:

1. Differentially private representation learning in general is a difficult problem. [248] showed that even handcrafted features can outperform feature learned by state-of-the-art DP-trained models, and attaining high-utility learned representations requires significantly more training data—much more than what is provided in typical supervised/curated datasets.

2. Combining self-supervised learning (SSL) with internet-scale *uncurated* datasets may seem like a natural approach to gain access to the large amount of data needed for DP training. However, most vision SSL training algorithms are based on *contrastive learning*, where the objective function depends on multiple samples in an entangled manner. This makes it difficult to perform the per-sample gradient computation needed in DP-SGD.

3. SSL training requires a much larger number of training epochs compared to supervised learning, which sharply increases the DP parameter ϵ, leading to meaningless privacy guarantees.

In this work, we describe a successful recipe for training differentially private large-scale foundation models via SSL. Firstly, we identify masked autoencoder (MAE; [93]) as a promising SSL training algorithm that is amenable to DP-SGD. MAE uses an instance-separable loss function

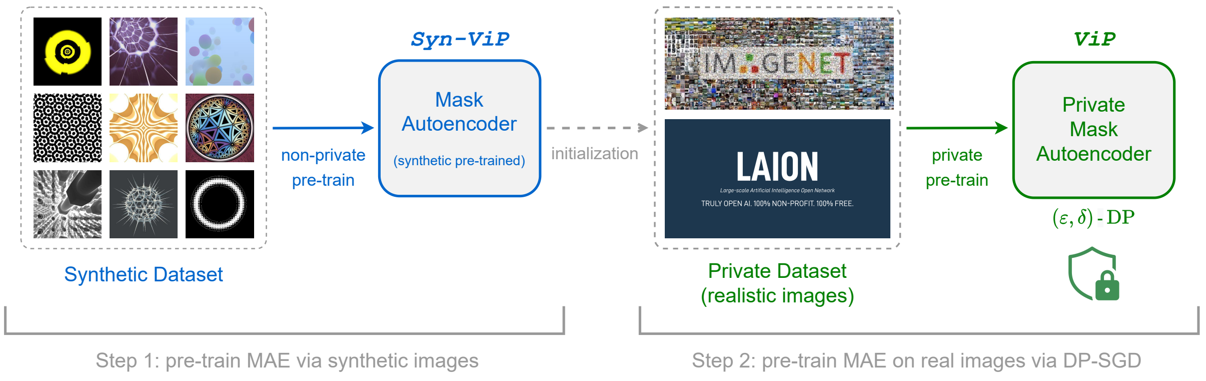


Figure 5.2: **How to pre-train differentially private transformers (*ViP*) with synthetic data?** In Step 1, we first pre-train a MAE model on synthetic images with standard optimizers (*e.g.*, SGD, AdamW). We denote this model by *(Syn)-ViP*. In Step 2, we use the MAE model pre-trained on synthetic images as initialization, and then apply differential private optimizers (*e.g.*, DP-SGD, DP-AdamW) to train a *ViP* model that satisfies (ϵ, δ)-DP.

and does not require batch normalization, and hence per-sample gradients can be easily computed. We also show that it is tolerant to the large amount of Gaussian noise added in DP-SGD. Next, we demonstrate that MAE can effectively leverage synthetic datasets containing only programmatically-generated synthesized textures [16] to warm-start the DP training process, significantly reducing the number of training epochs required to reach a high-utility model. The combination of these two ingredients forms a powerful DP training recipe for obtaining high-utility differentially private foundation vision models.

We implement this training recipe on the LAION400M dataset [217]. We show that the resulting model, which we call *ViP* (**Vi**sion transformer with differential **P**rivacy), learns highly useful and transferable representations—*rivaling that of representation learned by SimCLR on ImageNet*—while providing a strong DP guarantee with $\epsilon = 8$. In Figure 5.1, we compare ViP with other private and non-private models in terms of downstream linear probing accuracy and fine-tuning accuracy for different image datasets:

- For iNat-2021 and Places-365 classification, ViP outperforms both TAN [215]—the previous SOTA for DP supervised training—and AlexNet [145], while matching or exceeding the performance of SimCLR pre-trained on ImageNet.
- On ImageNet, the linear probing accuracy of ViP matches that of end-to-end trained AlexNet[1].
- On MS-COCO detection and segmentation, ViP outperforms both SimCLR pre-trained on ImageNet and Mask R-CNN.

Our experiments demonstrate that by scaling DP-SGD training to vast amounts of unlabeled data and using synthetic data to warm-start the model, we can attain high-utility foundation vision models under stringent privacy guarantees. Consequently, we hope that future work can continue

[1]The model is sourced from the PyTorch website and is end-to-end trained with supervised learning.

to build on our successful recipe and further push the performance boundary of large-scale DP training.

5.2 Background

Differential privacy [65] is a mathematical framework for formal reasoning about information leakage through a private mechanism. A learning algorithm $\mathcal{A}$ is said to be (ϵ, δ)-*differentially private* (denoted (ϵ, δ)-DP) if for all training datasets $\mathcal{D}, \mathcal{D}'$ that differ[2] in a single training sample, we have:

$$P(\mathcal{A}(\mathcal{D}) \in S) \leq e^\epsilon P(\mathcal{A}(\mathcal{D}') \in S) + \delta \tag{5.1}$$

for all outcome sets S. More generally, (5.1) can be expressed as a statistical divergence

$$D(\mathcal{A}(\mathcal{D}) \| \mathcal{A}(\mathcal{D}'))$$

between the distribution of models trained on $\mathcal{D}$ vs. $\mathcal{D}'$, with (ϵ, δ)-DP corresponding to the "hockey-stick" divergence [219]. Another useful variant is *Rényi differential privacy* (RDP; [179]), which uses the Rényi divergence D_α [208]: $\mathcal{A}$ is said to be (α, ϵ)-RDP if $D_\alpha(\mathcal{A}(\mathcal{D}) || \mathcal{A}(\mathcal{D}')) \leq \epsilon$. Moreover, RDP can be converted to DP via the following [14]: if $\mathcal{A}$ is $(\alpha, \epsilon_\alpha)$-RDP then it is also (ϵ, δ)-DP with

$$\epsilon = \epsilon_\alpha + \log\left(\frac{\alpha - 1}{\alpha}\right) - \frac{\log \delta + \log \alpha}{\alpha - 1}. \tag{5.2}$$

DP-SGD training. [1] showed that stochastic gradient descent (SGD)—the quintessential learning algorithm—can be made differentially private by perturbing the per-iteration gradient with Gaussian noise. The modified SGD update with gradient perturbation (often referred to as *DP-SGD*) is given by $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \widetilde{\boldsymbol{g}}_t$, and

$$\widetilde{\boldsymbol{g}}_t = \frac{1}{B}\left(\sum_{\mathbf{x} \in \mathcal{B}_t} \mathsf{clip}_C(\nabla_{\boldsymbol{\theta}} \ell(\mathbf{x}; \boldsymbol{\theta}_t) + \mathcal{N}(0, \sigma^2 C^2 \boldsymbol{I})\right), \tag{5.3}$$

where η_t is the learning rate, $\mathcal{B}_t$ is the sampled batch, B is the average batch size, $\sigma > 0$ is the noise multiplier, and $\mathsf{clip}_C(\cdot)$ is the operation that clips the per-sample gradient norm to at most $C > 0$. It can be shown that this update procedure is $(\alpha, \epsilon_\alpha)$-RDP for some computable ϵ_α [180]. The end-to-end learning algorithm by running T iterations of SGD is thus $(\alpha, T\epsilon_\alpha)$-RDP via composition [179], and a conversion to (ϵ, δ)-DP can be obtained using (5.2). Such privatization mechanism—per-sample clipping and injecting noise—can be easily integrated with other first-order optimization algorithms such as Adam [139] and AdamW [170].

[2]We adopt the removal notion of adjacency, *i.e.*, $\mathcal{D}' = \mathcal{D} \cup \mathbf{z}$ for some $\mathbf{z}$ and vice versa.

Self-supervised learning (SSL) has emerged as a prominent approach for scaling up the training of machine learning models to large-scale unlabeled datasets. Restricting our attention to the vision domain, SSL pre-trained models generalize effectively across a wide range of transfer learning downstream tasks such as classification, instance segmentation and object detection [41, 20], especially under the scenario of limited downstream training data. Vision SSL methods can be broadly categorized as either *joint embedding-based learning* (JE) [39, 95, 85, 297, 43] or *reconstruction-based learning* (REC) [15, 281, 93]. JE-based approaches design objective functions so that all views (or image augmentations) of the same sample have similar embeddings, while views of different samples have different embeddings. As a result, most JE-based approaches *require* a batch containing multiple samples in order to define the objective function. On the other hand, REC-based approaches aim to optimize models to reconstruct image inputs in the pixel space based on partially masked inputs, which promotes the model to learn compressed representations that can generalize well.

Related work. Recently, an expanding body of literature has emerged on scaling DP training to large-scale datasets and models in both NLP and vision domains. In NLP, a series of works [7, 287, 160] showed that by combining public pre-training and scaling up the training batch size, it is possible to fine-tune the pre-trained language model to achieve reasonable downstream performance. In computer vision, [149] first attempted to scale DP training of convolutional neural networks (ResNets) to ImageNet. [54] further improved the performance of [149] with a Normalizer-Free ResNet architecture and an improved training recipe. More recently, [215] proposed a more efficient hyperparameter tuning method for DP training that led to state-of-the-art performance on ImageNet. It is worth noting that all these works on DP-trained computer vision models focus on training supervised models.

5.3 Recipe for Training DP Foundation Vision Models

In this work, we identify a successful recipe for training differentially private foundation vision models. Training DP foundation models, or in general any deep learning model with a large number of parameters, poses a significant challenge due to the large amount of injected noise—$\mathcal{N}(0, \sigma^2 C^2 \boldsymbol{I})$ in (5.3). Indeed, current state-of-the-art differentially private deep learning models even under-perform linear models with handcrafted features when ϵ is small [54, 248]. We propose two effective techniques that reduce the magnitude of noise injected during training while attaining strong (ϵ, δ)-DP guarantees: **1.** Scaling up the number of training samples via self-supervised learning with masked autoencoder; and **2.** Facilitating faster training by warm-starting the model with weights pre-trained on synthetic samples.

5.3.1 Differential Private SSL with Mask Autoencoder

Most existing works on differentially private training [54, 215, 28] focus on supervised learning, which inherently restricts the quantity of training samples that can be utilized. In contrast, self-supervised learning approaches unlock the use of (albeit uncurated) internet-scale training data that can be on the order of billions of samples, which can potentially satisfy the amount of data needed for DP training of high-utility models [248].

On the other hand, most existing SSL training approaches do not align with requirements in DP-SGD training. For example, SimCLR [39] requires a mini-batch of samples in order to compute the contrastive loss; BYOL [85] computes per-sample loss but it utilizes batch normalization (BN) [120] in the model architecture, resulting in each loss depending on a mini-batch of training samples.[3] Therefore, it is challenging to perform the per-sample gradient clipping as described in (5.3). Among various types of SSL methods, we identify reconstruction-base learning with masked autoencoders (MAE) [93] as one of the most suitable SSL approaches for training DP foundation vision models. The training objective $L_{\text{MAE}}(\boldsymbol{\theta})$ is defined as:

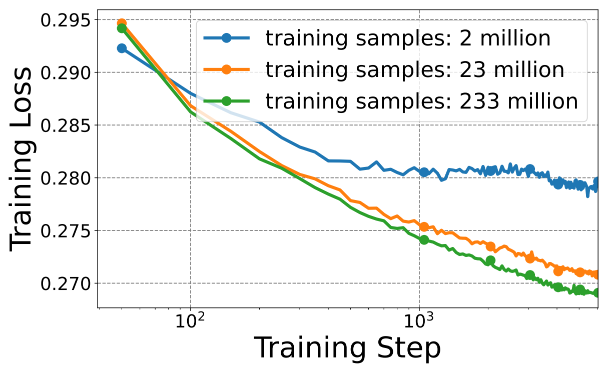
$$L_{\text{MAE}}(\boldsymbol{\theta}) := \frac{1}{n}\sum_{i=1}^{n} \underbrace{\ell_{\text{MSE}}(g \circ \psi(\text{mask}(\mathbf{x}_i); \boldsymbol{\theta}), \mathbf{x}_i)}_{\ell(\mathbf{x}_i;\boldsymbol{\theta})}, \tag{5.4}$$

where n is the number of training samples, $\mathbf{x}_i \in \mathbb{R}^{C\times H\times W}$ is the input of the i-th training image (C-number of channels, H-height, W-width), $\text{mask}(\cdot)$ is a function that mask out a fraction of the image, $\psi : \mathbb{R}^{C\times H\times W} \to \mathbb{R}^d$ is the encoder and $g : \mathbb{R}^d \to \mathbb{R}^{C\times H\times W}$ is the decoder. We use $\boldsymbol{\theta}$ to denote the trainable parameters of the ψ and g, and use ℓ_{MSE} to denote the mean squared error (MSE) loss defined on the pixel space, *i.e.*, $\ell_{\text{MSE}}(\mathbf{x}_1, \mathbf{x}_2) = \|\mathbf{x}_1 - \mathbf{x}_2\|_F^2$. Similar to [93], we apply vision transformers [63] to instantiate the encoder and decoder maps. As shown in (5.4), the training objective can be decomposed into n individual losses, and each individual loss $\ell(\mathbf{x}_i; \boldsymbol{\theta})$ only depends on the i-th training sample $\mathbf{x}_i$ and does not require the label of $\mathbf{x}_i$. Therefore, we can compute per-sample gradient $\nabla_{\boldsymbol{\theta}}\ell(\mathbf{x}_i; \boldsymbol{\theta})$ and perform per-sample gradient clipping without modifying the MAE training.

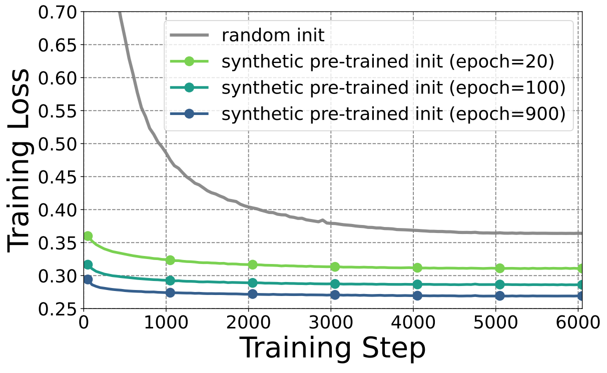
By leveraging the self-supervised MAE training paradigm, we can now significantly scale up the training data size for DP SSL pre-training. Dataset scaling can effectively reduce the magnitude of noise in DP-SGD while maintaining the same (ϵ, δ_n)-DP guarantee, where $\delta_n = 1/2n$. As shown in Figure 5.3a, we investigate the impact of injected noise in ViP training by keeping all training hyperparameters the same except for the number of training samples[4]. With more training samples, the magnitude of the injected noise σ becomes smaller. We find that when the noise magnitude is large, the training loss cannot be further optimized after certain number of training steps. In contrast, smaller magnitude of noise (as a result of larger training dataset) facilitates faster optimization of the training loss in comparison to larger noise scenarios. Importantly, the optimization trajectory is stable despite the presence of noise, allowing the MAE model to learn useful features.

[3]Subsequent work by [209] demonstrated that BN can be substituted with group normalization by carefully modifying the model architecture. However, we have observed that the design of exponential moving averaged online network in BYOL can result in dynamic instability during training, which poses challenges in the context of DP training.

[4]We maintain the same batch size across various data size settings while modifying the noise multiplier σ. Consequently, as the data size increases, the corresponding σ values decrease.



(a) Role of dataset size.



(b) Effectiveness of synthetic pre-training.

Figure 5.3: (a). We vary the number of training samples n with the (ϵ, δ_n)-DP guarantee ($\delta_n = 1/2n$), and compare the training losses of MAE-DP. By scaling up the training dataset size, we can consistently improve the ViP training under the same ϵ-DP budget. (b). Compared to ViP training from random initialization, we can significantly speed up the ViP training by leveraging the synthetic pre-trained MAE model as initialization.

5.3.2 Synthetic Pre-training Enables Faster DP Training for ViP

Non-private training of SSL models often require a significant number of training epochs, much larger than what is required in supervised learning [39, 93, 13]. This creates an additional challenge for DP training since the number of training iterations T directly impacts the privacy guarantee. Indeed, as mentioned in Section 5.2, DP-SGD with T iterations is $(\alpha, T\epsilon_\alpha)$-RDP. Consequently, naively applying DP-SGD to MAE training results in an unfavorable privacy-utility trade-off.

Fortunately, [96] demonstrated that using pre-trained initialization enables much faster model convergence compared to random initialization. However, in light of our discussion in Section 5.1, it is critical that the pre-training data does not contain any private information, even if the data is deemed “public”. One promising alternative is pre-training on programmatically-generated synthetic images [137, 16], which was shown to achieve competitive downstream performance compared to pre-training on natural images. Doing so allows the MAE to learn spatial structure in the transformer modules [123] without expending any privacy budget for the natural image data. More importantly, synthetic pre-training does not carry any privacy risk, and legal risk is limited to obtaining proper license for the synthetic image generation code.

Thus, to accelerate ViP training, we pre-train the model on synthetic images generated using the Shaders21k tool developed in [16]. Figure 5.2 shows samples of synthetic images generated by the tool. In Figure 5.3b, we compare the ViP training with and without synthetic pre-trained initialization. Notably, training ViP with synthetic pre-trained weights converges significantly faster than those with random initialized weights. Increasing the synthetic pre-training from 20 to 900 epochs further improves convergence for ViP training. Interestingly, as shown in Figure 5.1, MAE trained on the synthetic dataset already outperforms existing state-of-the-art DP-trained models [54, 215] under our transfer learning evaluation, which shows that DP training on datasets even as large as ImageNet does not learn sufficiently expressive features (see Table 5.1).

5.3.3 Our Proposed Approach

We now summarize our approach for DP foundation vision model training (also see Figure 5.2). It is worth mentioning that our proposed approach offers flexibility in the selection of both SSL training methods and synthetic datasets. For example, developing better synthetic datasets or more effective SSL learning method can further push the performance of the final DP foundation model.

DP-MAE with Synthetic Pre-training

- **Step 1:** *Synthetic pre-training for initialization.* Pre-train mask autoencoder on the synthetic dataset with non-private optimizers.
- **Step 2:** *DP training with synthetic initialization.* Apply the synthetic pre-trained model as initialization and train mask autoencoder on a large-scale natural image dataset (*e.g.*, LAION400M) with DP-SGD. The DP guarantee then applies to the natural image dataset.

5.4 Evaluation

We evaluate the effectiveness of our training recipe by applying it to the LAION400M dataset to train our private foundation vision model: **ViP**. We consider various downstream tasks in order to demonstrate the quality and transferability of its learned representation. Furthermore, we compare ViP to previous state-of-the-art DP-trained models as well as widely adopted non-privately trained models, and find that ViP significantly improves SOTA for DP training on downstream transfer tasks (Section 5.4.2) and even outperforms non-private models on several challenging datasets. In addition to assessing the performance of ViP on non-private downstream tasks, in Section D.2.3, we also evaluate the ViP model via DP fine-tuning on ImageNet-1K, which shows a notable improvement of 10%+ absolute top-1 accuracy compared to previous SOTA [215]. For additional experimental results on ViP, see Appendix D.2.

5.4.1 Evaluation Setup

Our implementation uses PyTorch, along with the functorch package [110] for computation of per-sample gradients and the opacus package [286] for privacy accounting. See Appendix D.1 for additional implementation details.

Datasets. We use 1.05 million samples generated using the Shader21k [16] tool as our synthetic pre-training dataset, and the LAION400M [217] as our private pre-training dataset for the ViP model[5]. We evaluate ViP and baseline models via *non-private* linear probing and fine-tuning

[5]Some of the links in LAION400M are now broken since its initial release, and the version we use contains $\sim$233 million real images. We use LAION233M to denote this subsampled version of LAION400M.

Table 5.1: Linear probing evaluation on downstream classification. We compare *ViP* with both private pre-training (DP-NFNet and TAN) and non-private pre-training (AlexNet and SimCLR) baselines, as well as the synthetically pre-trained MAE model: *(Syn)-ViP*. *ViP* consistently outperforms all private baselines, and has similar transfer learning performance as non-private SimCLR pre-trained on ImageNet-1K. (‡All models except for *(Syn)-ViP* and *ViP* are pre-trained on ImageNet-1K, giving them an unfair advantage for the linear probing evaluation on ImageNet-1K.)

Model	DP?	SSL?	Pre-train dataset	# pre-train samples	ImageNet-1K‡	Places-365	Places-205	iNat-2021
DP-NFNet	✓	✗	ImageNet-1k	~1 million	45.3%	40.1%	39.2%	28.2%
TAN	✓	✗	ImageNet-1k	~1 million	49.0%	40.5%	38.2%	31.7%
AlexNet	✗	✗	ImageNet-1k	~1 million	56.5%	39.8%	35.1%	23.7%
SimCLR	✗	✓	ImageNet-1k	~1 million	67.5%	46.8%	49.3%	34.8%
(Syn)-ViP	✓	✓	Shaders21k	~1 million	49.8%	43.2%	45.8%	32.4%
ViP-LAION	✓	✓	LAION	~233 million	**55.7%**	**46.1%**	**48.5%**	**38.1%**
ViP-ImageNet	✓	✓	ImageNet-1k	~1 million	52.6%	44.3%	46.5%	34.2%

Table 5.2: Fine-tuning evaluation on few-shot downstream classification. ViP consistently outperforms both TAN (private) and AlexNet (non-private), as well as (Syn)-ViP by a large margin. Performance does fall short compared to non-private SimCLR pre-trained on ImageNet-1K despite having access to more than $100\times$ more data, suggesting that there is much room for improvement for private learning.

Model	Aircraft			Caltech-101			CIFAR-100		
	10-shot	20-shot	30-shot	5-shot	10-shot	30-shot	5-shot	10-shot	30-shot
AlexNet	23.27%	34.47%	41.35%	64.70%	73.57%	81.40%	29.74%	36.31%	49.28%
SimCLR	38.79%	56.90%	64.90%	81.70%	89.11%	94.51%	49.93%	60.18%	71.84%
TAN	22.84%	37.93%	46.01%	49.32%	66.42%	77.87%	21.28%	27.78%	42.35%
(Syn)-ViP	21.79%	46.85%	58.45%	60.51%	76.21%	88.48%	27.62%	38.96%	55.84%
ViP	**31.62%**	**53.05%**	**64.26%**	**68.05%**	**79.03%**	**88.90%**	**30.73%**	**40.95%**	**57.52%**

on the following downstream classification datasets: ImageNet-1K [59], Places-365 and Places-205 [304], iNaturalist-2021 [249], CIFAR-100 [144], Caltech101 [71], and Aircraft [175]. The input images are resized and center-cropped to 224×224 resolution. We also evaluate using MS-COCO instance segmentation and object detection [166], and semantic segmentation with the ADE20K dataset [305] (in Appendix D.2.1).

Model architecture. Following [93], we use vision transformer (ViT) [63] to instantiate the masked autoencoder models. The default MAE-encoder has 12 transformer blocks and width 768, and the default MAE-decoder has 4 transformer blocks and width 512. We denote this MAE model as MAE-base. We also consider MAE models with different model sizes, including MAE-Nano, MAE-Tiny, MAE-Small and MAE-Large in Section 5.4.3.

Optimization and hyperparameters for (DP-)MAE training. We use AdamW [170] for training MAE – both for synthetic pre-training and differentially private MAE pre-training. When evaluating pre-trained models in downstream tasks, we apply LARS [285] for linear probing and AdamW for fine-tuning. For MAE training, we set the masking ratio to 75%. In terms of DP training, we set $\epsilon = 8.0$ and $\delta = 1/2n$ by default for training (ϵ, δ)-DP model. We set the clipping parameter $C = 0.1$, sampling ratio $q = 81920/n$, and noise parameter $\sigma = 0.5$.

Existing methods for comparison. We compare with existing state-of-the-art DP-trained models: DP-NFNet [54] and TAN [215]), both of which are trained differentially privately on ImageNet-1K using supervised learning. In addition, we present the results of several widely used *non-private* models that are pre-trained on ImageNet-1K including AlexNet [145] (supervised learning-based) and SimCLR [39] (SSL-based) for reference. To measure the effectiveness of DP pre-training compared to synthetic pre-training, we also evaluate the model pre-trained on synthetically generated Shader21k data, denoted **(Syn)-ViP**. We also compare ViP with the non-private MAE model pre-trained on the same datasets and summarize the results in Table D.4 (Appendix D.2.4).

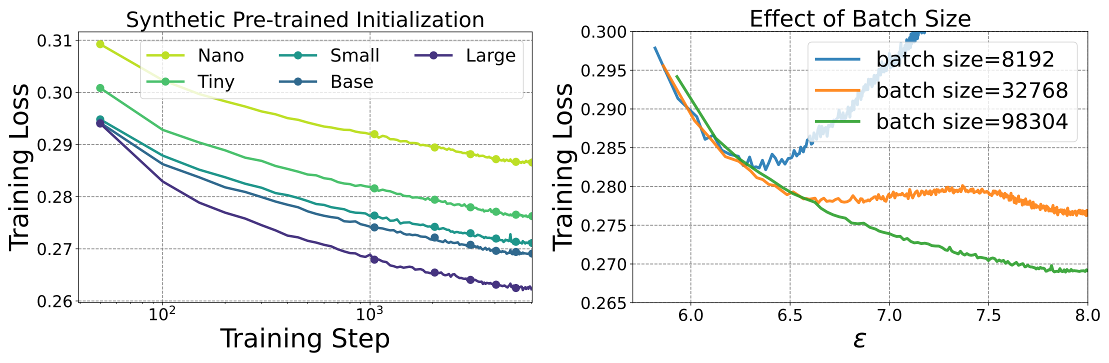
5.4.2 Transfer Learning Evaluation

To show that ViP learns high-quality representations from its training data, we evaluate its transfer learning performance on a suite of image classification tasks using both linear probing and few-shot fine-tuning. For linear probing, we use all the training samples in the downstream task training set to learn the linear classifier, while freezing all layers except for the final linear layer. For few-shot fine-tuning, we randomly select K training samples from each class and fine-tune the entire model. It is worth noting that both linear probing and fine-tuning evaluations are done using *non-private* training; our pre-trained ViP model only satisfies (ϵ, δ)-DP on the LAION233M dataset.

Linear probing. Table 5.1 shows the linear probing results on four large-scale image classification datasets: ImageNet-1K, Places-365/205 and iNat-2021. The most suitable baselines in this setting are DP-NFNet and TAN, both of which are DP-trained on ImageNet-1K with $\epsilon = 8$ and represent previous state-of-the-art in large-scale DP pre-training. First of all, we find that MAE pre-training only on synthetic images (*i.e.*, (Syn)-ViP) is already comparable or even outperforms SOTA DP pre-trained models. After differentially privately pre-training on LAION233M, ViP effectively improves the performance of (Syn)-ViP on all datasets by a large margin.

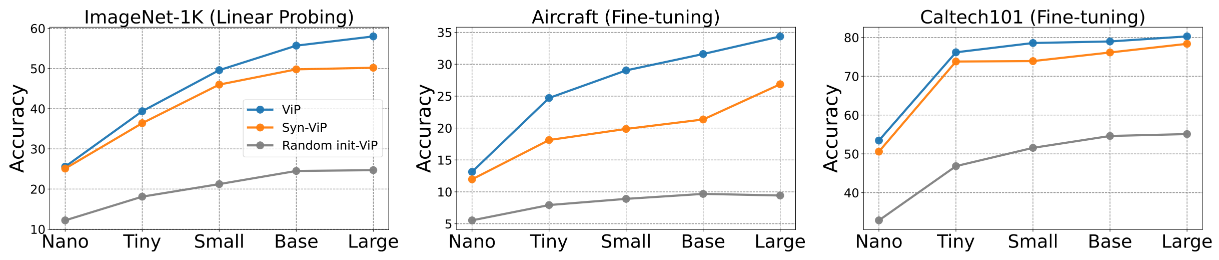
Importantly, ViP even outperforms *non-private* SimCLR pre-trained on ImageNet-1K on all datasets (except ImageNet-1k itself because SimCLR does not need to transfer), and achieves similar performance as end-to-end non-privately trained AlexNet. To the best of our knowledge, this is the first time a DP-trained model can achieve similar performance on vision benchmark datasets as that of a mainstream (albeit older) model, which demonstrates the potential of our training recipe.

Few-shot fine-tuning. Table 5.2 shows the few-shot fine-tuning results on Aircraft, Caltech-101 and CIFAR-100. Similar to the linear probing result, (Syn)-ViP already outperforms TAN—the previous SOTA DP-trained model—across all evaluation settings except for 10-shot classification



(a) Scaling up model. (b) Scaling up batch size.

Figure 5.4: (**Left**) Effect of scaling up model size on MAE training loss. Larger models attain lower training loss despite the larger magnitude of noise added during DP-SGD. (**Right**) Effect of batch size on MAE training loss while fixing ϵ. A large batch size is necessary for convergence.



(a) ImageNet-1K (linear probing) (b) Aircraft (fine-tuning) (c) Caltech (fine-tuning)

Figure 5.5: Effect of scaling up model size on downstream performance. ViP with synthetic pre-training (blue line) benefits substantially from larger model size. In comparison, ViP with random initialization (gray line) does not benefit as much from model scaling, as the difference in performance between MAE-Large and MAE-Nano is considerably smaller.

on Aircraft. Next, we find that ViP can largely improve upon (Syn)-ViP when the number of samples per class is small, attaining SOTA performance in all evaluation settings. ViP also achieves better performance than non-privately pre-trained AlexNet by a large margin, but falls short against non-private SimCLR despite having access to more than $100\times$ training data. Thus, our result can be viewed as both a positive and a negative result, showing that there is still a long way to go for private learning before matching the performance of mainstream vision models across the board.

5.4.3 Scaling Properties

We now study scaling properties of our training recipe, including scaling up (1) the model size, (2) the training set size, and (3) the previously known successful recipe of scaling up batch size.

Scaling up model size. DP-SGD training is generally unfavorable to large models because the noise magnitude increases with model size. Interestingly, we show that model performance in fact improves by scaling up model size using our training recipe. Specifically, we change the MAE-encoder size while fixing the MAE-decoder size, resulting in five different model sizes from MAE-Nano to MAE-Large; Table D.1 in Appendix D.1.1) gives architecture details including number of parameters. All models are trained to satisfy the same (ϵ, δ)-DP guarantee with $\epsilon = 8$.

Figure 5.4a plots the training curve for the different-sized models. At the beginning of DP training, due to synthetic pre-training, a larger MAE model can learn more expressive features and hence the MAE training loss on LAION233M decreases as model size increases. Intriguingly, the training losses of MAE-Small/Base/Large are similar at the beginning, but larger ViT models achieve faster convergence *despite the large amount of DP noise*. Although similar observations on larger models converge faster have also been described in the context of non-private learning [162], the fact that we observe the same phenomenon in Figure 5.4a suggests that model scaling can be effective even for *private* learning under our training recipe.

Figure 5.5 shows the effect of model scaling on downstream linear probing and fine-tuning performance. In particular, the effective reduction in training loss shown in Figure 5.4a indeed translates to better downstream performance, with larger ViP model consistently achieving better accuracy without modifications to the training process. Moreover, comparing ViP with synthetic pre-training (blue line) vs. random initialization (gray line) shows that synthetic pre-training is crucial for unlocking this scaling behavior: the difference in performance between MAE-Large and MAE-Nano is much smaller when the model is randomly initialized.

Scaling up dataset size. Next, we investigate the effect of scaling up the number of training samples in ViP training. We vary the training dataset size from 2M to 23M to 233M while choosing the magnitude of injected noise σ so that models trained on different dataset sizes satisfy (ϵ, δ_n)-DP guarantee with $\epsilon = 8$ and $\delta_n = 1/2n$, where n is the number of training samples. Table D.8 shows downstream evaluation results. The first row corresponds to the synthetically pre-trained ViP model and rows 2-4 correspond to DP-trained ViP models with different dataset sizes. As expected, a larger pre-training dataset size results in a higher-utility ViP model. For example, scaling from 2M to 233M gives 3.1% linear probing accuracy gain on ImageNet-1K (from 52.6% to 55.7%). Given that the collection of large labeled datasets is very costly in practice, these results highlight the significance of self-supervised learning in DP training.

Scaling up batch size. Scaling up the training batch size is a known effective way to achieve strong performance in DP supervised learning [160]. We analyze the effect of batch size in training ViP models and show that the same observation holds for DP self-supervised learning. We consider three different batch size $B \in \{8192, 32768, 98304\}$, and keep the computational budget—number of per-sample gradient computation—the same for all batch sizes. We then select the noise σ such that models trained with different batch size satisfy the same (ϵ, δ)-DP. As shown in Figure 5.4b, we find that larger batch size leads to better stability in the training process as well as faster convergence under the same computational budget. Rows 5-7 in Table D.8 demonstrate that larger batch size also translates to a substantial improvement in ViP's transfer learning performance.

5.5 Discussion and Future Work

We developed a recipe for DP self-supervised learning of foundation vision models, and showed that the resulting model—ViP—can achieve downstream performance matching or exceeding that of mainstream non-private models such as SimCLR (with ImageNet-1K pre-training). Our work shows the potential of scaling DP training to internet-scale unlabeled datasets and presents several opportunities for future work. **1.** Our recipe adapted MAE to DP-SGD training with minimal modifications. It may be possible to design more specialized SSL training algorithms that conform to the requirements of DP-SGD and are more effective at learning useful representations. **2.** Multi-modal SSL is generally more effective than single-modality pre-training due to the additional supervision from cross-modal alignment [185]. However, existing multi-modal SSL methods are mostly based on contrastive learning (*e.g.*, CLIP [201], SLIP [185] and FLIP [161]) and do not admit per-sample gradient computation. Recent work [114] investigated how to fine-tune CLIP on vision-language tasks with DP guarantee. Additional work may be needed to adapt these methods to DP-SGD training.

Bibliography

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. “Deep learning with differential privacy”. In: *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 2016, pp. 308–318.

[2] Samira Abnar and Willem H. Zuidema. “Quantifying Attention Flow in Transformers”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*. Ed. by Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault. Association for Computational Linguistics, 2020, pp. 4190–4197. DOI: 10.18653/V1/2020.ACL-MAIN.385. URL: https://doi.org/10.18653/v1/2020.acl-main.385.

[3] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas, Matthew Mattina, Paul Whatmough, and Venkatesh Saligrama. “Federated Learning Based on Dynamic Regularization”. In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=B7v4QMR6Z9w.

[4] Alessandro Achille, Aditya Golatkar, Avinash Ravichandran, Marzia Polito, and Stefano Soatto. “Lqf: Linear quadratic fine-tuning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 15729–15739.

[5] Andrei Afonin and Sai Praneeth Karimireddy. “Towards Model Agnostic Federated Learning Using Knowledge Distillation”. In: *arXiv preprint arXiv:2110.15210* (2021).

[6] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. “QSGD: Communication-efficient SGD via gradient quantization and encoding”. In: *Advances in Neural Information Processing Systems* 30 (2017).

[7] Rohan Anil, Badih Ghazi, Vineet Gupta, Ravi Kumar, and Pasin Manurangsi. “Large-scale differentially private BERT”. In: *arXiv preprint arXiv:2108.01624* (2021).

[8] Martin Arjovsky, Soumith Chintala, and Léon Bottou. “Wasserstein Generative Adversarial Networks”. In: *International Conference on Machine Learning*. 2017, pp. 214–223.

[9] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lucic, and Cordelia Schmid. “ViViT: A Video Vision Transformer”. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE Computer Society. 2021, pp. 6816–6826.

[10] Haim Avron, Kenneth L. Clarkson, and David P. Woodruff. "Sharper Bounds for Regularized Data Fitting". In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques,*

[11] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. "How to backdoor federated learning". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020, pp. 2938–2948.

[12] Pierre Baldi and Kurt Hornik. "Neural networks and principal component analysis: Learning from examples without local minima". In: *Neural networks* 2.1 (1989), pp. 53–58.

[13] Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari S. Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Grégoire Mialon, Yuandong Tian, Avi Schwarzschild, Andrew Gordon Wilson, Jonas Geiping, Quentin Garrido, Pierre Fernandez, Amir Bar, Hamed Pirsiavash, Yann LeCun, and Micah Goldblum. "A Cookbook of Self-Supervised Learning". In: 2023.

[14] Borja Balle, Gilles Barthe, Marco Gaboardi, Justin Hsu, and Tetsuya Sato. "Hypothesis testing interpretations and renyi differential privacy". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020, pp. 2496–2506.

[15] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. "Beit: Bert pre-training of image transformers". In: *arXiv preprint arXiv:2106.08254* (2021).

[16] Manel Baradad, Richard Chen, Jonas Wulff, Tongzhou Wang, Rogerio Feris, Antonio Torralba, and Phillip Isola. "Procedural Image Programs for Representation Learning". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 6450–6462.

[17] Amir Beck and Marc Teboulle. "A fast iterative shrinkage-thresholding algorithm for linear inverse problems". In: *SIAM journal on imaging sciences* 2.1 (2009), pp. 183–202.

[18] Shai Ben-David, Pavel Hrubes, Shay Moran, Amir Shpilka, and Amir Yehudayoff. *A learning problem that is independent of the set theory ZFC axioms*. 2017. arXiv: `1711.05195 [cs.LG]`.

[19] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. "Machine learning with adversaries: Byzantine tolerant gradient descent". In: *Advances in Neural Information Processing Systems* 30 (2017).

[20] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. "On the opportunities and risks of foundation models". In: *arXiv preprint arXiv:2108.07258* (2021).

[21] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ B. Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen Creel, Jared Quincy Davis, Dorottya Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, and et al. "On the Opportunities and Risks of Foundation Models". In: *CoRR* abs/2108.07258 (2021). arXiv: 2108.07258. URL: https://arxiv.org/abs/2108.07258.

[22] Kallista Bonawitz, Peter Kairouz, Brendan McMahan, and Daniel Ramage. "Federated Learning and Privacy: Building privacy-preserving systems for machine learning and data science on decentralized data". In: *Queue* 19.5 (2021), pp. 87–114.

[23] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konecny, Stefano Mazzocchi, Brendan McMahan, et al. "Towards federated learning at scale: System design". In: *Proceedings of Machine Learning and Systems* 1 (2019), pp. 374–388.

[24] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. "Practical secure aggregation for privacy-preserving machine learning". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017, pp. 1175–1191.

[25] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[26] 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". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[27] Joan Bruna and Stéphane Mallat. "Invariant scattering convolution networks". en. In: *IEEE transactions on pattern analysis and machine intelligence* 35.8 (Aug. 2013), pp. 1872–1886. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2012.230. URL: http://dx.doi.org/10.1109/TPAMI.2012.230.

[28] Zhiqi Bu, Jialin Mao, and Shiyun Xu. "Scalable and efficient training of large convolutional neural networks with differential privacy". In: *arXiv preprint arXiv:2205.10683* (2022).

[29] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. "End-to-End Object Detection with Transformers". In: (May 2020). arXiv: 2005.12872 [cs.CV]. URL: http://arxiv.org/abs/2005.12872.

[30] Nicholas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramer, Borja Balle, Daphne Ippolito, and Eric Wallace. "Extracting training data from diffusion models". In: *arXiv preprint arXiv:2301.13188* (2023).

[31] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom B Brown, Dawn Song, Ulfar Erlingsson, et al. "Extracting Training Data from Large Language Models." In: *USENIX Security Symposium*. Vol. 6. 2021.

[32] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. "Emerging Properties in Self-Supervised Vision Transformers". In: *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, 2021, pp. 9630–9640. DOI: `10.1109/ICCV48922.2021.00951`. URL: `https://doi.org/10.1109/ICCV48922.2021.00951`.

[33] Kwan Ho Ryan Chan, Yaodong Yu, Chong You, Haozhi Qi, John Wright, and Yi Ma. "ReduNet: A White-box Deep Network from the Principle of Maximizing Rate Reduction". In: *Journal of Machine Learning Research* 23.114 (2022), pp. 1–103. URL: `http://jmlr.org/papers/v23/21-0631.html`.

[34] Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. "Deep adaptive image clustering". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 5879–5887.

[35] Zachary Charles, Zachary Garrett, Zhouyuan Huo, Sergei Shmulyian, and Virginia Smith. "On large-cohort training for federated learning". In: *Advances in Neural Information Processing Systems* 34 (2021).

[36] El Mahdi Chayti and Sai Praneeth Karimireddy. "Optimization with access to auxiliary information". In: *arXiv preprint arXiv:2206.00395* (2022).

[37] Hongrui Chen, Holden Lee, and Jianfeng Lu. "Improved Analysis of Score-based Generative Modeling: User-Friendly Bounds under Minimal Smoothness Assumptions". In: *arXiv preprint arXiv:2211.01916* (2022).

[38] Sitan Chen, Giannis Daras, and Alexandros G Dimakis. "Restoration-Degradation Beyond Linear Diffusions: A Non-Asymptotic Analysis For DDIM-Type Samplers". In: (Mar. 2023). arXiv: `2303.03384 [cs.LG]`. URL: `http://arxiv.org/abs/2303.03384`.

[39] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. "A Simple Framework for Contrastive Learning of Visual Representations". In: *arXiv preprint arXiv:2002.05709* (2020).

[40] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. "A Simple Framework for Contrastive Learning of Visual Representations". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé Iii and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 1597–1607. URL: `https://proceedings.mlr.press/v119/chen20j.html`.

[41] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. “Big self-supervised models are strong semi-supervised learners”. In: *Advances in neural information processing systems* 33 (2020), pp. 22243–22255.

[42] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, et al. “Symbolic discovery of optimization algorithms”. In: *arXiv preprint arXiv:2302.06675* (2023).

[43] Xinlei Chen and Kaiming He. “Exploring simple siamese representation learning”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 15750–15758.

[44] Yubei Chen, Dylan Paiton, and Bruno Olshausen. “The sparse manifold transform”. In: *Advances in neural information processing systems* 31 (2018).

[45] Adam Coates, Andrew Ng, and Honglak Lee. “An analysis of single-layer networks in unsupervised feature learning”. In: *International Conference on Artificial Intelligence and Statistics*. 2011, pp. 215–223.

[46] Taco Cohen and Max Welling. “Group equivariant convolutional networks”. In: *International Conference on Machine Learning*. 2016, pp. 2990–2999.

[47] Taco S Cohen, Mario Geiger, and Maurice Weiler. “A general theory of equivariant cnns on homogeneous spaces”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 9142–9153.

[48] Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. “Exploiting shared representations for personalized federated learning”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 2089–2099.

[49] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. USA: Wiley-Interscience, 2006. ISBN: 0471241954.

[50] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. “Autoaugment: Learning augmentation policies from data”. In: *arXiv preprint arXiv:1805.09501* (2018).

[51] Rachel Cummings, Damien Desfontaines, David Evans, Roxana Geambasu, Matthew Jagielski, Yangsibo Huang, Peter Kairouz, Gautam Kamath, Sewoong Oh, Olga Ohrimenko, et al. “Challenges towards the next frontier in privacy”. In: *arXiv preprint arXiv:2304.06929* (2023).

[52] Xili Dai, Shengbang Tong, Mingyang Li, Ziyang Wu, Michael Psenka, Kwan Ho Ryan Chan, Pengyuan Zhai, Yaodong Yu, Xiaojun Yuan, Heung-Yeung Shum, et al. “Ctrl: Closed-loop transcription to an ldr via minimaxing rate reduction”. In: *Entropy* 24.4 (2022), p. 456.

[53] Amit Daniely, Sivan Sabato, Shai Ben-David, and Shai Shalev-Shwartz. “Multiclass Learnability and the ERM Principle”. In: *J. Mach. Learn. Res.* 16.1 (Jan. 2015), pp. 2377–2404. ISSN: 1532-4435.

[54] Soham De, Leonard Berrada, Jamie Hayes, Samuel L Smith, and Borja Balle. "Unlocking high-accuracy differentially private image classification through scale". In: *arXiv preprint arXiv:2204.13650* (2022).

[55] Brent De Weerdt, Yonina C Eldar, and Nikos Deligiannis. "Designing Transformer Networks for Sparse Recovery of Sequential Data Using Deep Unfolding". In: *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. June 2023, pp. 1–5. DOI: 10.1109/ICASSP49357.2023.10094712. URL: http://dx.doi.org/10.1109/ICASSP49357.2023.10094712.

[56] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, et al. "Large scale distributed deep networks". In: *Advances in Neural Information Processing Systems* 25 (2012).

[57] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. "SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives". In: *Advances in Neural Information Processing Systems* 27 (2014).

[58] Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. "Scaling vision transformers to 22 billion parameters". In: *arXiv preprint arXiv:2302.05442* (2023).

[59] 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*. Ieee. 2009, pp. 248–255.

[60] Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. "Adaptive personalized federated learning". In: *arXiv preprint arXiv:2003.13461* (2020).

[61] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 4171–4186.

[62] David L Donoho and Carrie Grimes. "Image Manifolds which are Isometric to Euclidean Space". In: *Journal of mathematical imaging and vision* 23.1 (July 2005), pp. 5–24. ISSN: 0924-9907, 1573-7683. DOI: 10.1007/s10851-005-4965-4. URL: https://doi.org/10.1007/s10851-005-4965-4.

[63] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929* (2020).

[64] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. "Calibrating noise to sensitivity in private data analysis". In: *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*. Springer. 2006, pp. 265–284.

[65] Cynthia Dwork, Aaron Roth, et al. "The algorithmic foundations of differential privacy." In: *Found. Trends Theor. Comput. Sci.* 9.3-4 (2014), pp. 211–407.

[66] Bradley Efron. "Tweedie's Formula and Selection Bias". en. In: *Journal of the American Statistical Association* 106.496 (2011), pp. 1602–1614. ISSN: 0162-1459. DOI: 10.1198/jasa.2011.tm11181. URL: http://dx.doi.org/10.1198/jasa.2011.tm11181.

[67] Gamaleldin Elsayed, Dilip Krishnan, Hossein Mobahi, Kevin Regan, and Samy Bengio. "Large margin deep networks for classification". In: *Advances in neural information processing systems*. 2018, pp. 842–852.

[68] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. "Personalized federated learning: A meta-learning approach". In: *arXiv preprint arXiv:2002.07948* (2020).

[69] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. "Local Model Poisoning Attacks to Byzantine-Robust Federated Learning". In: *29th USENIX Security Symposium (USENIX Security 20)*. 2020, pp. 1605–1622.

[70] Maryam Fazel, Haitham Hindi, and Stephen P Boyd. "Log-det heuristic for matrix rank minimization with applications to Hankel and Euclidean distance matrices". In: *Proceedings of the 2003 American Control Conference, 2003.* Vol. 3. IEEE. 2003, pp. 2156–2162.

[71] Li Fei-Fei, Robert Fergus, and Pietro Perona. "One-shot learning of object categories". In: *IEEE transactions on pattern analysis and machine intelligence* 28.4 (2006), pp. 594–611.

[72] Pedro F Felzenszwalb and Daniel P Huttenlocher. "Pictorial Structures for Object Recognition". In: *International journal of computer vision* 61.1 (Jan. 2005), pp. 55–79. ISSN: 0920-5691, 1573-1405. DOI: 10.1023/B:VISI.0000042934.15159.49. URL: https://doi.org/10.1023/B:VISI.0000042934.15159.49.

[73] Pedro F. Felzenszwalb, David A. McAllester, and Deva Ramanan. "A discriminatively trained, multiscale, deformable part model". In: *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008), 24-26 June 2008, Anchorage, Alaska, USA*. IEEE Computer Society, 2008. DOI: 10.1109/CVPR.2008.4587597. URL: https://doi.org/10.1109/CVPR.2008.4587597.

[74] Stanislav Fort, Gintare Karolina Dziugaite, Mansheej Paul, Sepideh Kharaghani, Daniel M Roy, and Surya Ganguli. "Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the Neural Tangent Kernel". In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 5850–5861.

[75] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. "Mitigating sybils in federated learning poisoning". In: *arXiv preprint arXiv:1808.04866* (2018).

[76] Ross Girshick. “Fast r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[77] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[78] Micah Goldblum, Jonas Geiping, Avi Schwarzschild, Michael Moeller, and Tom Goldstein. “Truth or backpropaganda? An empirical investigation of deep learning theory”. In: *arXiv preprint arXiv:1910.00359* (2019).

[79] Yuan Gong, Andrew Rouditchenko, Alexander H Liu, David Harwath, Leonid Karlinsky, Hilde Kuehne, and James R Glass. “Contrastive audio-visual masked autoencoder”. In: *The Eleventh International Conference on Learning Representations*. 2022.

[80] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[81] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. “Generative adversarial nets”. In: *Advances in Neural Information Processing Systems*. 2014, pp. 2672–2680.

[82] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. “Accurate, large minibatch SGD: Training Imagenet in 1 hour”. In: *arXiv preprint arXiv:1706.02677* (2017).

[83] Karol Gregor and Yann LeCun. “Learning fast approximations of sparse coding”. In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. Omnipress. 2010, pp. 399–406.

[84] Rémi Gribonval, Rodolphe Jenatton, and Francis Bach. “Sparse and spurious: dictionary learning with noise and outliers”. In: (July 2014). arXiv: `1407.5155 [cs.LG]`. URL: `http://arxiv.org/abs/1407.5155`.

[85] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. “Bootstrap your own latent-a new approach to self-supervised learning”. In: *Advances in neural information processing systems* 33 (2020), pp. 21271–21284.

[86] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. “On calibration of modern neural networks”. In: *International conference on machine learning*. PMLR. 2017, pp. 1321–1330.

[87] László Györfi, Michael Kohler, Adam Krzyzak, and Harro Walk. *A Distribution-Free Theory of Nonparametric Regression*. en. Springer New York, Dec. 2010. ISBN: 9781441929983. URL: `https://play.google.com/store/books/details?id=_RoFkgAACAAJ`.

[88] Farzin Haddadpour, Mohammad Mahdi Kamani, Aryan Mokhtari, and Mehrdad Mahdavi. “Federated learning with compression: Unified analysis and sharp guarantees”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2021, pp. 2350–2358.

[89] Raia Hadsell, Sumit Chopra, and Yann LeCun. “Dimensionality reduction by learning an invariant mapping”. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2006*. 2006, pp. 1735–1742.

[90] Benjamin D Haeffele, Chong You, and René Vidal. “A Critique of Self-Expressive Deep Subspace Clustering”. In: *arXiv preprint arXiv:2010.03697* (2020).

[91] Hangfeng He and Weijie J Su. “A law of data separation in deep learning”. In: *arXiv preprint arXiv:2210.17020* (2022).

[92] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. “Masked Autoencoders Are Scalable Vision Learners”. In: (Nov. 2021). arXiv: `2111.06377 [cs.CV]`. URL: `http://arxiv.org/abs/2111.06377`.

[93] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. “Masked autoencoders are scalable vision learners”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 16000–16009.

[94] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. “Momentum contrast for unsupervised visual representation learning”. In: *arXiv preprint arXiv:1911.05722* (2019).

[95] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. “Momentum contrast for unsupervised visual representation learning”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 9729–9738.

[96] Kaiming He, Ross Girshick, and Piotr Dollár. “Rethinking imagenet pre-training”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 4918–4927.

[97] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. “Mask R-CNN”. In: (Mar. 2017). arXiv: `1703.06870 [cs.CV]`. URL: `http://arxiv.org/abs/1703.06870`.

[98] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. “Mask r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[99] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016, pp. 770–778. DOI: `10.1109/CVPR.2016.90`. URL: `http://dx.doi.org/10.1109/CVPR.2016.90`.

[100] 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*. 2016, pp. 770–778.

[101] Lie He, Sai Praneeth Karimireddy, and Martin Jaggi. “Byzantine-robust decentralized learning via self-centered clipping”. In: *arXiv preprint arXiv:2202.01545* (2022).

[102] Peter Henderson, Xuechen Li, Dan Jurafsky, Tatsunori Hashimoto, Mark A Lemley, and Percy Liang. "Foundation Models and Fair Use". In: *arXiv preprint arXiv:2303.15715* (2023).

[103] Geoffrey Hinton. *How to represent part-whole hierarchies in a neural network*. 2021. arXiv: `2102.12627 [cs.CV]`.

[104] Geoffrey Hinton. *The Forward-Forward Algorithm: Some Preliminary Investigations*. 2022. arXiv: `2212.13345 [cs.LG]`.

[105] Geoffrey E Hinton and Richard Zemel. "Autoencoders, Minimum Description Length and Helmholtz Free Energy". In: *Advances in Neural Information Processing Systems*. Ed. by J. Cowan, G. Tesauro, and J. Alspector. Vol. 6. Morgan-Kaufmann, 1993.

[106] Geoffrey E. Hinton, Alex Krizhevsky, and Sida D. Wang. "Transforming Auto-Encoders". In: *Artificial Neural Networks and Machine Learning - ICANN 2011 - 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I*. Ed. by Timo Honkela, Wlodzislaw Duch, Mark A. Girolami, and Samuel Kaski. Vol. 6791. Lecture Notes in Computer Science. Springer, 2011, pp. 44–51. DOI: `10.1007/978-3-642-21735-7\_6`. URL: `https://doi.org/10.1007/978-3-642-21735-7%5C_6`.

[107] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. "Learning deep representations by mutual information estimation and maximization". In: *arXiv preprint arXiv:1808.06670* (2018).

[108] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6840–6851.

[109] Benjamin Hoover, Yuchen Liang, Bao Pham, Rameswar Panda, Hendrik Strobelt, Duen Horng Chau, Mohammed J Zaki, and Dmitry Krotov. "Energy Transformer". In: (Feb. 2023). arXiv: `2302.07253 [cs.LG]`. URL: `http://arxiv.org/abs/2302.07253`.

[110] Richard Zou Horace He. *functorch: JAX-like composable function transforms for PyTorch*. `https://github.com/pytorch/functorch`. 2021.

[111] Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip Gibbons. "The non-iid data quagmire of decentralized machine learning". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 4387–4398.

[112] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. "Measuring the effects of non-identical data distribution for federated visual classification". In: *arXiv preprint arXiv:1909.06335* (2019).

[113] Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, and Masashi Sugiyama. "Learning Discrete Representations via Information Maximizing Self-Augmented Training". In: *International Conference on Machine Learning*. 2017, pp. 1558–1567.

[114] Alyssa Huang, Peihan Liu, Ryumei Nakada, Linjun Zhang, and Wanrong Zhang. "Safeguarding Data in Multimodal AI: A Differentially Private Approach to CLIP Training". In: *arXiv preprint arXiv:2306.08173* (2023).

[115] Baihe Huang, Xiaoxiao Li, Zhao Song, and Xin Yang. "Fl-ntk: A neural tangent kernel-based framework for federated learning analysis". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 4423–4434.

[116] Lawrence Hubert and Phipps Arabie. "Comparing partitions". In: *Journal of Classification* 2.1 (1985), pp. 193–218.

[117] Like Hui and Mikhail Belkin. "Evaluation of neural architectures trained with square loss vs cross-entropy in classification tasks". In: *arXiv preprint arXiv:2006.07322* (2020).

[118] Aapo Hyvärinen. "Estimation of Non-Normalized Statistical Models by Score Matching". In: *Journal of machine learning research: JMLR* 6.24 (2005), pp. 695–709. ISSN: 1532-4435, 1533-7928. URL: `https://www.jmlr.org/papers/v6/hyvarinen05a.html`.

[119] Forrest N Iandola, Matthew W Moskewicz, Khalid Ashraf, and Kurt Keutzer. "Firecaffe: near-linear acceleration of deep neural network training on compute clusters". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2592–2600.

[120] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. pmlr. 2015, pp. 448–456.

[121] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. "Averaging weights leads to wider optima and better generalization". In: *arXiv preprint arXiv:1803.05407* (2018).

[122] Arthur Jacot, Franck Gabriel, and Clément Hongler. "Neural tangent kernel: Convergence and generalization in neural networks". In: *Advances in Neural Information Processing Systems* 31 (2018).

[123] Samy Jelassi, Michael Sander, and Yuanzhi Li. "Vision transformers provably learn spatial structure". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 37822–37836.

[124] Pan Ji, Tong Zhang, Hongdong Li, Mathieu Salzmann, and Ian Reid. "Deep subspace clustering networks". In: *Advances in Neural Information Processing Systems*. 2017, pp. 24–33.

[125] Xu Ji, João F Henriques, and Andrea Vedaldi. "Invariant information clustering for unsupervised image classification and segmentation". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 9865–9874.

[126] Rie Johnson and Tong Zhang. "Accelerating stochastic gradient descent using predictive variance reduction". In: *Advances in Neural Information Processing Systems* 26 (2013).

[127] Ian T Jolliffe. *Principal Component Analysis*. 2nd. Springer-Verlag, 2002.

[128] Charles I Jones and Christopher Tonetti. "Nonrivalry and the Economics of Data". In: *American Economic Review* 110.9 (2020), pp. 2819–58.

[129] Zahra Kadkhodaie and Eero P Simoncelli. "Solving Linear Inverse Problems Using the Prior Implicit in a Denoiser". In: (July 2020). arXiv: `2007.13640 [cs.CV]`. URL: `http://arxiv.org/abs/2007.13640`.

[130] Peter Kairouz, H. Brendan McMahan, et al. "Advances and open problems in federated learning". In: *Foundations and Trends® in Machine Learning* 14.1–2 (2021), pp. 1–210.

[131] Zhao Kang, Chong Peng, Jie Cheng, and Qiang Cheng. "Logdet rank minimization with application to subspace clustering". In: *Computational Intelligence and Neuroscience* 2015 (2015).

[132] Sai Praneeth Karimireddy, Lie He, and Martin Jaggi. "Byzantine-Robust Learning on Heterogeneous Datasets via Bucketing". In: *International Conference on Learning Representations*. 2021.

[133] Sai Praneeth Karimireddy, Martin Jaggi, Satyen Kale, Mehryar Mohri, Sashank J Reddi, Sebastian U Stich, and Ananda Theertha Suresh. "Mime: Mimicking centralized stochastic algorithms in federated learning". In: *arXiv preprint arXiv:2008.03606* (2020).

[134] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. "Scaffold: Stochastic controlled averaging for federated learning". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 5132–5143.

[135] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. "Elucidating the design space of diffusion-based generative models". In: *arXiv preprint arXiv:2206.00364* (2022).

[136] Tero Karras, Samuli Laine, and Timo Aila. "A Style-Based Generator Architecture for Generative Adversarial Networks". In: (Dec. 2018). arXiv: `1812.04948 [cs.NE]`. URL: `http://arxiv.org/abs/1812.04948`.

[137] Hirokatsu Kataoka, Kazushige Okayasu, Asato Matsumoto, Eisuke Yamagata, Ryosuke Yamada, Nakamasa Inoue, Akio Nakamura, and Yutaka Satoh. "Pre-training without natural images". In: *Proceedings of the Asian Conference on Computer Vision*. 2020.

[138] Koray Kavukcuoglu, Marc'Aurelio Ranzato, Rob Fergus, and Yann LeCun. "Learning invariant features through topographic filter maps". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2009, pp. 1605–1612.

[139] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[140] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. "Segment Anything". In: (Apr. 2023). arXiv: `2304.02643 [cs.CV]`. URL: `http://arxiv.org/abs/2304.02643`.

[141] Frederic Koehler, Alexander Heckett, and Andrej Risteski. "Statistical Efficiency of Score Matching: The View from Isoperimetry". In: (Oct. 2022). arXiv: `2210.00726 [cs.LG]`. URL: `http://arxiv.org/abs/2210.00726`.

[142] Artemy Kolchinsky, Brendan D Tracey, and Steven Van Kuyk. "Caveats for information bottleneck in deterministic scenarios". In: *arXiv preprint arXiv:1808.07593* (2018).

[143] Mark A Kramer. "Nonlinear principal component analysis using autoassociative neural networks". In: *AIChE Journal* 37.2 (1991), pp. 233–243.

[144] Alex Krizhevsky. "Learning multiple layers of features from tiny images". In: (2009). URL: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.222.9220&rep=rep1&type=pdf`.

[145] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Communications of the ACM* 60.6 (2017), pp. 84–90.

[146] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*. Ed. by Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger. 2012, pp. 1106–1114. URL: `https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html`.

[147] Viraj Kulkarni, Milind Kulkarni, and Aniruddha Pant. "Survey of personalization techniques for federated learning". In: *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*. IEEE. 2020, pp. 794–797.

[148] Bogdan Kulynych, David Madras, Smitha Milli, Inioluwa Deborah Raji, Angela Zhou, and Richard Zemel. *Participatory Approaches to Machine Learning*. International Conference on Machine Learning Workshop. 2020.

[149] Alexey Kurakin, Shuang Song, Steve Chien, Roxana Geambasu, Andreas Terzis, and Abhradeep Thakurta. "Toward training at imagenet scale with differential privacy". In: *arXiv preprint arXiv:2201.12328* (2022).

[150] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition". In: *Proc. IEEE* 86.11 (1998), pp. 2278–2324. DOI: `10.1109/5.726791`. URL: `https://doi.org/10.1109/5.726791`.

[151] Yann LeCun, Fu Jie Huang, and Leon Bottou. "Learning methods for generic object recognition with invariance to pose and lighting". In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* Vol. 2. IEEE. 2004, pp. II–104.

[152] Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. “Wide neural networks of any depth evolve as linear models under gradient descent”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[153] José Lezama, Qiang Qiu, Pablo Musé, and Guillermo Sapiro. “OLE: Orthogonal low-rank embedding-a plug and play geometric loss for deep learning”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8109–8118.

[154] Hongkang Li, Meng Wang, Sijia Liu, and Pin-Yu Chen. “A Theoretical Understanding of shallow Vision Transformers: Learning, Generalization, and Sample Complexity”. In: *arXiv preprint arXiv:2302.06015* (2023).

[155] Ke Li, Shichong Peng, Tianhao Zhang, and Jitendra Malik. “Multimodal Image Synthesis with Conditional Implicit Maximum Likelihood Estimation”. In: *International Journal of Computer Vision* (2020).

[156] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. “Federated learning on non-iid data silos: An experimental study”. In: *arXiv preprint arXiv:2102.02079* (2021).

[157] Qinbin Li, Bingsheng He, and Dawn Song. “Model-contrastive federated learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 10713–10722.

[158] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. “Federated optimization in heterogeneous networks”. In: *Proceedings of Machine Learning and Systems* 2 (2020), pp. 429–450.

[159] Tian Li, Maziar Sanjabi, Ahmad Beirami, and Virginia Smith. “Fair resource allocation in federated learning”. In: *arXiv preprint arXiv:1905.10497* (2019).

[160] Xuechen Li, Florian Tramer, Percy Liang, and Tatsunori Hashimoto. “Large Language Models Can Be Strong Differentially Private Learners”. In: *International Conference on Learning Representations*. 2022. URL: `https://openreview.net/forum?id=bVuP3ltATMz`.

[161] Yanghao Li, Haoqi Fan, Ronghang Hu, Christoph Feichtenhofer, and Kaiming He. “Scaling Language-Image Pre-training via Masking”. In: *arXiv preprint arXiv:2212.00794* (2022).

[162] Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joey Gonzalez. “Train big, then compress: Rethinking model size for efficient training and inference of transformers”. In: *International Conference on machine learning*. PMLR. 2020, pp. 5958–5968.

[163] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. “Refinenet: Multi-path refinement networks for high-resolution semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1925–1934.

[164] Tao Lin, Sai Praneeth Karimireddy, Sebastian U Stich, and Martin Jaggi. "Quasi-global momentum: Accelerating decentralized deep learning on heterogeneous data". In: *arXiv preprint arXiv:2102.04761* (2021).

[165] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. "Ensemble distillation for robust model fusion in federated learning". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 2351–2363.

[166] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. "Microsoft COCO: Common Objects in Context". In: *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*. Ed. by David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars. Vol. 8693. Lecture Notes in Computer Science. Springer, 2014, pp. 740–755. DOI: `10.1007/978-3-319-10602-1\_48`. URL: `https://doi.org/10.1007/978-3-319-10602-1%5C_48`.

[167] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. "Explainable ai: A review of machine learning interpretability methods". In: *Entropy* 23.1 (2020), p. 18.

[168] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[169] Philip M Long. "Properties of the after kernel". In: *arXiv preprint arXiv:2105.10585* (2021).

[170] Ilya Loshchilov and Frank Hutter. "Decoupled weight decay regularization". In: *arXiv preprint arXiv:1711.05101* (2017).

[171] Lingjuan Lyu, Han Yu, and Qiang Yang. "Threats to federated learning: A survey". In: *arXiv preprint arXiv:2003.02133* (2020).

[172] Yi Ma, Harm Derksen, Wei Hong, and John Wright. "Segmentation of multivariate mixed data via lossy data coding and compression". In: *PAMI* (2007).

[173] Yi Ma, Doris Tsao, and Heung-Yeung Shum. "On the principles of parsimony and self-consistency for the emergence of intelligence". In: *Frontiers of Information Technology & Electronic Engineering* 23.9 (2022), pp. 1298–1323.

[174] Jan MacDonald, Stephan Wäldchen, Sascha Hauch, and Gitta Kutyniok. "A Rate-Distortion Framework for Explaining Neural Network Decisions". In: *CoRR* abs/1905.11092 (2019). arXiv: `1905.11092`. URL: `http://arxiv.org/abs/1905.11092`.

[175] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. "Fine-grained visual classification of aircraft". In: *arXiv preprint arXiv:1306.5151* (2013).

[176] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. "Communication-efficient learning of deep networks from decentralized data". In: *Artificial Intelligence and Statistics*. PMLR. 2017, pp. 1273–1282.

[177] Casey Meehan, Florian Bordes, Pascal Vincent, Kamalika Chaudhuri, and Chuan Guo. "Do SSL Models Have Déjà Vu? A Case of Unintended Memorization in Self-supervised Learning". In: *arXiv e-prints* (2023), arXiv–2304.

[178] Peyman Milanfar. "A Tour of Modern Image Filtering: New Insights and Methods, Both Practical and Theoretical". In: *IEEE Signal Processing Magazine* 30.1 (Jan. 2013), pp. 106–128. ISSN: 1558-0792. DOI: `10.1109/MSP.2011.2179329`. URL: `http://dx.doi.org/10.1109/MSP.2011.2179329`.

[179] Ilya Mironov. "Renyi differential privacy". In: *2017 IEEE 30th computer security foundations symposium (CSF)*. IEEE. 2017, pp. 263–275.

[180] Ilya Mironov, Kunal Talwar, and Li Zhang. "R\'enyi differential privacy of the sampled gaussian mechanism". In: *arXiv preprint arXiv:1908.10530* (2019).

[181] Konstantin Mishchenko, Grigory Malinovsky, Sebastian Stich, and Peter Richtárik. "ProxSkip: Yes! Local Gradient Steps Provably Lead to Communication Acceleration! Finally!" In: *arXiv preprint arXiv:2202.09357* (2022).

[182] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. "Agnostic federated learning". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 4615–4625.

[183] Viraaji Mothukuri, Reza M Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. "A survey on security and privacy of federated learning". In: *Future Generation Computer Systems* 115 (2021), pp. 619–640.

[184] Fangzhou Mu, Yingyu Liang, and Yin Li. "Gradients as features for deep representation learning". In: *arXiv preprint arXiv:2004.05529* (2020).

[185] Norman Mu, Alexander Kirillov, David Wagner, and Saining Xie. "Slip: Self-supervision meets language-image pre-training". In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVI*. Springer. 2022, pp. 529–544.

[186] Oliver Nina, Jamison Moody, and Clarissa Milligan. "A Decoder-Free Approach for Unsupervised Clustering and Manifold Learning with Random Triplet Mining". In: *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. IEEE. 2019, pp. 3987–3994.

[187] Bruno A Olshausen and David J Field. "Sparse coding with an overcomplete basis set: A strategy employed by V1?" In: *Vision research* 37.23 (1997), pp. 3311–3325.

[188] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. "In-context Learning and Induction Heads". In: *CoRR* abs/2209.11895 (2022). DOI: `10.48550/ARXIV.2209.11895`. arXiv: `2209.11895`. URL: `https://doi.org/10.48550/arXiv.2209.11895`.

[189] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. “Representation learning with contrastive predictive coding”. In: *arXiv preprint arXiv:1807.03748* (2018).

[190] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael G. Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. “DINOv2: Learning Robust Visual Features without Supervision”. In: *CoRR* abs/2304.07193 (2023). DOI: 10.48550/ARXIV.2304.07193. arXiv: 2304.07193. URL: https://doi.org/10.48550/arXiv.2304.07193.

[191] 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”. In: *Advances in neural information processing systems* 32 (2019).

[192] Kaan Ozkara, Navjot Singh, Deepesh Data, and Suhas Diggavi. “QuPeD: Quantized Personalization via Distillation with Applications to Federated Learning”. In: *Advances in Neural Information Processing Systems* 34 (2021).

[193] Druv Pai, Michael Psenka, Chih-Yuan Chiu, Manxi Wu, Edgar Dobriban, and Yi Ma. “Pursuit of a discriminative representation for multiple subspaces via sequential games”. In: *arXiv preprint arXiv:2206.09120* (2022).

[194] Vardan Papyan, Yaniv Romano, Jeremias Sulam, and Michael Elad. “Theoretical Foundations of Deep Learning via Sparse Representations: A Multilayer Sparse Model and Its Connection to Convolutional Neural Networks”. In: *IEEE Signal Processing Magazine* 35.4 (July 2018), pp. 72–89. ISSN: 1558-0792. DOI: 10.1109/MSP.2018.2820224. URL: http://dx.doi.org/10.1109/MSP.2018.2820224.

[195] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. “Automatic differentiation in PyTorch”. In: (2017).

[196] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. “PyTorch: An imperative style, high-performance deep learning library”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 8024–8035.

[197] Xi Peng, Jiashi Feng, Shijie Xiao, Jiwen Lu, Zhang Yi, and Shuicheng Yan. “Deep sparse subspace clustering”. In: *arXiv preprint arXiv:1709.08374* (2017).

[198] Alex Pentland, Alexander Lipton, and Thomas Hardjono. *Building the New Economy: Data as Capital*. MIT Press, 2021.

[199] Haozhi Qi, Chong You, Xiaolong Wang, Yi Ma, and Jitendra Malik. “Deep Isometric Learning for Visual Recognition”. In: *Proceedings of the International Conference on International Conference on Machine Learning*. 2020.

[200] J. R. Quinlan. “Induction of Decision Trees”. In: *Mach. Learn.* 1.1 (Mar. 1986), pp. 81–106. ISSN: 0885-6125. DOI: 10.1023/A:1022643204877. URL: https://doi.org/10.1023/A:1022643204877.

[201] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. “Learning transferable visual models from natural language supervision”. In: *International conference on machine learning*. PMLR. 2021, pp. 8748–8763.

[202] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. “Learning Transferable Visual Models From Natural Language Supervision”. In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 8748–8763. URL: https://proceedings.mlr.press/v139/radford21a.html.

[203] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. “Robust Speech Recognition via Large-Scale Weak Supervision”. In: *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*. Ed. by Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett. Vol. 202. Proceedings of Machine Learning Research. PMLR, 2023, pp. 28492–28518.

[204] Swaroop Ramaswamy, Om Thakkar, Rajiv Mathews, Galen Andrew, H Brendan McMahan, and Francoise Beaufays. “Training production language models without memorizing user data”. In: *arXiv preprint arXiv:2009.10031* (2020).

[205] Martin Raphan and Eero P Simoncelli. “Least squares estimation without priors or supervision”. en. In: *Neural computation* 23.2 (Feb. 2011), pp. 374–420. ISSN: 0899-7667, 1530-888X. DOI: 10.1162/NECO_a_00076. URL: http://dx.doi.org/10.1162/NECO_a_00076.

[206] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. “Do imagenet classifiers generalize to imagenet?” In: *International conference on machine learning*. PMLR. 2019, pp. 5389–5400.

[207] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konecny, Sanjiv Kumar, and Hugh Brendan McMahan. “Adaptive Federated Optimization”. In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=LkFG3lB13U5.

[208] Alfréd Rényi et al. “On measures of entropy and information”. In: *Proceedings of the fourth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 547-561. Berkeley, California, USA. 1961.

[209] Pierre H Richemond, Jean-Bastien Grill, Florent Altché, Corentin Tallec, Florian Strub, Andrew Brock, Samuel Smith, Soham De, Razvan Pascanu, Bilal Piot, et al. "Byol works even without batch statistics". In: *arXiv preprint arXiv:2010.10241* (2020).

[210] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. "Contractive auto-encoders: Explicit invariance during feature extraction". In: *In International Conference on Machine Learning*. 2011, pp. 833–840.

[211] Yaniv Romano, Michael Elad, and Peyman Milanfar. "The Little Engine That Could: Regularization by Denoising (RED)". In: *SIAM journal on imaging sciences* 10.4 (Jan. 2017), pp. 1804–1844. DOI: 10.1137/16M1102884. URL: https://doi.org/10.1137/16M1102884.

[212] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. "High-resolution image synthesis with latent diffusion models". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 10684–10695.

[213] Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. "Dynamic Routing Between Capsules". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett. 2017, pp. 3856–3866. URL: https://proceedings.neurips.cc/paper/2017/hash/2cad8fa47bbef282badbb8de5374b89
Abstract.html.

[214] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding". In: (May 2022). arXiv: 2205.11487 [cs.CV]. URL: http://arxiv.org/abs/2205.11487.

[215] Tom Sander, Pierre Stock, and Alexandre Sablayrolles. "TAN without a burn: Scaling Laws of DP-SGD". In: *arXiv preprint arXiv:2210.03403* (2022).

[216] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. "Laion-5b: An open large-scale dataset for training next generation image-text models". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 25278–25294.

[217] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. "Laion-400m: Open dataset of clip-filtered 400 million image-text pairs". In: *arXiv preprint arXiv:2111.02114* (2021).

[218] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[219] Naresh Sharma and Naqueeb Ahmad Warsi. “Fundamental bound on the reliability of quantum information transmission”. In: *Physical review letters* 110.8 (2013), p. 080501.

[220] Yanyao Shen and Sujay Sanghavi. “Learning with bad training data via iterative trimmed loss minimization”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 5739–5748.

[221] Yuxin Shi, Han Yu, and Cyril Leung. “A Survey of Fairness-Aware Federated Learning”. In: *arXiv preprint arXiv:2111.01872* (2021).

[222] Ravid Shwartz-Ziv and Yann LeCun. “To Compress or Not to Compress–Self-Supervised Learning and Information Theory: A Review”. In: *arXiv preprint arXiv:2304.09355* (2023).

[223] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *International Conference on Learning Representations*. 2015.

[224] Sidak Pal Singh and Martin Jaggi. “Model fusion via optimal transport”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 22045–22055.

[225] Jinhyun So, Basak Guler, and A Salman Avestimehr. “Byzantine-resilient secure federated learning”. In: *IEEE Journal on Selected Areas in Communications* 39.7 (2020), pp. 2168–2181.

[226] Jascha Sohl-Dickstein, Eric A Weiss, Niru Maheswaranathan, and Surya Ganguli. “Deep Unsupervised Learning using Nonequilibrium Thermodynamics”. In: (Mar. 2015). arXiv: `1503.03585 [cs.LG]`. URL: `http://arxiv.org/abs/1503.03585`.

[227] Gowthami Somepalli, Vasu Singla, Micah Goldblum, Jonas Geiping, and Tom Goldstein. “Diffusion art or digital forgery? investigating data replication in diffusion models”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 6048–6058.

[228] Jiaming Song, Chenlin Meng, and Stefano Ermon. “Denoising Diffusion Implicit Models”. In: (Oct. 2020). arXiv: `2010.02502 [cs.LG]`. URL: `http://arxiv.org/abs/2010.02502`.

[229] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. “Stochastic gradient descent with differentially private updates”. In: *2013 IEEE global conference on signal and information processing*. IEEE. 2013, pp. 245–248.

[230] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. “Consistency models”. In: *arXiv preprint arXiv:2303.01469* (2023).

[231] Yang Song and Stefano Ermon. “Generative Modeling by Estimating Gradients of the Data Distribution”. In: (July 2019). arXiv: `1907.05600 [cs.LG]`. URL: `http://arxiv.org/abs/1907.05600`.

[232] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. "Score-Based Generative Modeling through Stochastic Differential Equations". In: (Nov. 2020). arXiv: 2011.13456 [cs.LG]. URL: http://arxiv.org/abs/2011.13456.

[233] Daniel A Spielman, Huan Wang, and John Wright. "Exact Recovery of Sparsely-Used Dictionaries". In: (June 2012). arXiv: 1206.5882 [cs.LG]. URL: http://arxiv.org/abs/1206.5882.

[234] Charles M Stein. "Estimation of the Mean of a Multivariate Normal Distribution". en. In: *The Annals of Statistics* 9.6 (Nov. 1981), pp. 1135–1151. ISSN: 0090-5364, 2168-8966. DOI: 10.1214/aos/1176345632. URL: https://projecteuclid.org/journals/annals-of-statistics/volume-9/issue-6/Estimation-of-the-Mean-of-a-Multivariate-Normal-Distribution/10.1214/aos/1176345632.full.

[235] Sebastian U Stich and Sai Praneeth Karimireddy. "The error-feedback framework: Better rates for sgd with delayed gradients and compressed updates". In: *Journal of Machine Learning Research* 21 (2020), pp. 1–36.

[236] Alexander Strehl and Joydeep Ghosh. "Cluster ensembles—a knowledge reuse framework for combining multiple partitions". In: *Journal of Machine Learning Research* 3.Dec (2002), pp. 583–617.

[237] Xiaoxia Sun, Nasser M Nasrabadi, and Trac D Tran. "Supervised deep sparse coding networks". In: *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE. 2018, pp. 346–350.

[238] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H Brendan McMahan. "Can you really backdoor federated learning?" In: *arXiv preprint arXiv:1911.07963* (2019).

[239] Ananda Theertha Suresh, X Yu Felix, Sanjiv Kumar, and H Brendan McMahan. "Distributed mean estimation with limited communication". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 3329–3337.

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

[241] Yue Tan, Guodong Long, Lu Liu, Tianyi Zhou, Qinghua Lu, Jing Jiang, and Chengqi Zhang. "Fedproto: Federated prototype learning over heterogeneous devices". In: *arXiv preprint arXiv:2105.00243* (2021).

[242] Jean Ogier du Terrail, Samy-Safwan Ayed, Edwige Cyffers, Felix Grimberg, Chaoyang He, Regis Loeb, Paul Mangold, Tanguy Marchand, Othmane Marfoq, Erum Mushtaq, et al. "FLamby: Datasets and Benchmarks for Cross-Silo Federated Learning in Realistic Settings". In: (2022).

[243] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. "What makes for good views for contrastive learning?" In: *Advances in neural information processing systems* 33 (2020), pp. 6827–6839.

[244] Naftali Tishby and Noga Zaslavsky. "Deep learning and the information bottleneck principle". In: *2015 IEEE Information Theory Workshop (ITW)*. IEEE. 2015, pp. 1–5.

[245] Bahareh Tolooshams and Demba Ba. "Stable and Interpretable Unrolled Dictionary Learning". In: *arXiv preprint arXiv:2106.00058* (2021).

[246] Shengbang Tong, Xili Dai, Ziyang Wu, Mingyang Li, Brent Yi, and Yi Ma. "Incremental Learning of Structured Memory via Closed-Loop Transcription". In: *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL: `https://openreview.net/pdf?id=XrgjF5-M3xi`.

[247] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. "Llama: Open and efficient foundation language models". In: *arXiv preprint arXiv:2302.13971* (2023).

[248] Florian Tramer and Dan Boneh. "Differentially private learning needs better features (or much more data)". In: *arXiv preprint arXiv:2011.11660* (2020).

[249] Grant Van Horn, Elijah Cole, Sara Beery, Kimberly Wilber, Serge Belongie, and Oisin Mac Aodha. "Benchmarking representation learning for natural world image collections". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 12884–12893.

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

[251] Singanallur V Venkatakrishnan, Charles A Bouman, and Brendt Wohlberg. "Plug-and-Play priors for model based reconstruction". In: *2013 IEEE Global Conference on Signal and Information Processing*. Dec. 2013, pp. 945–948. DOI: `10.1109/GlobalSIP.2013.6737048`. URL: `http://dx.doi.org/10.1109/GlobalSIP.2013.6737048`.

[252] Rene Vidal. *Attention: Self-Expression Is All You Need*. Unpublished; available: `https://openreview.net/forum?id=MmujBClawFo`. 2022. URL: `https://openreview.net/forum?id=MmujBClawFo`.

[253] Rene Vidal, Yi Ma, and S. S. Sastry. *Generalized Principal Component Analysis*. 1st. Springer Publishing Company, Incorporated, 2016. ISBN: 0387878106.

[254] René Vidal, Yi Ma, and Shankar Sastry. *Generalized Principal Component Analysis*. Springer Verlag, 2016.

[255] Pascal Vincent. "A connection between score matching and denoising autoencoders". en. In: *Neural computation* 23.7 (July 2011), pp. 1661–1674. ISSN: 0899-7667, 1530-888X. DOI: `10.1162/NECO\_a\_00142`. URL: `http://dx.doi.org/10.1162/NECO_a_00142`.

[256] Michael B Wakin, David L Donoho, Hyeokho Choi, and Richard G Baraniuk. "The multiscale structure of non-differentiable image manifolds". In: *Proceedings of SPIE, the International Society for Optical Engineering*. 2005, 59141B–1.

[257] Michael B Wakin, David L Donoho, Hyeokho Choi, and Richard G Baraniuk. "The multiscale structure of non-differentiable image manifolds". In: *Wavelets XI*. Vol. 5914. SPIE. 2005, pp. 413–429.

[258] Haoqing Wang, Xun Guo, Zhi-Hong Deng, and Yan Lu. "Rethinking minimal sufficient representation in contrastive learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 16041–16050.

[259] Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. "Attack of the tails: Yes, you really can backdoor federated learning". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 16070–16084.

[260] Jianyu Wang, Zachary Charles, Zheng Xu, Gauri Joshi, H Brendan McMahan, Maruan Al-Shedivat, Galen Andrew, Salman Avestimehr, Katharine Daly, Deepesh Data, et al. "A field guide to federated optimization". In: *arXiv preprint arXiv:2107.06917* (2021).

[261] Jianyu Wang, Rudrajit Das, Gauri Joshi, Satyen Kale, Zheng Xu, and Tong Zhang. "On the Unreasonable Effectiveness of Federated Averaging with Heterogeneous Data". In: *arXiv preprint arXiv:2206.04723* (2022).

[262] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. "Tackling the objective inconsistency problem in heterogeneous federated optimization". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 7611–7623.

[263] Jianyu Wang, Vinayak Tantia, Nicolas Ballas, and Michael Rabbat. "SlowMo: Improving communication-efficient distributed SGD with slow momentum". In: *arXiv preprint arXiv:1910.00643* (2019).

[264] Peng Wang, Huikang Liu, Druv Pai, Yaodong Yu, Zhihui Zhu, Qing Qu, and Yi Ma. "A Global Geometric Analysis of Maximal Coding Rate Reduction". In: *Forty-first International Conference on Machine Learning*. 2024. URL: `https://openreview.net/forum?id=u9qmjV2khT`.

[265] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K Leung, Christian Makaya, Ting He, and Kevin Chan. "Adaptive federated learning in resource constrained edge computing systems". In: *IEEE Journal on Selected Areas in Communications* 37.6 (2019), pp. 1205–1221.

[266] Xudong Wang, Rohit Girdhar, Stella X. Yu, and Ishan Misra. "Cut and Learn for Unsupervised Object Detection and Instance Segmentation". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*. IEEE, 2023, pp. 3124–3134. DOI: 10.1109/CVPR52729.2023.00305. URL: https://doi.org/10.1109/CVPR52729.2023.00305.

[267] Alexander Wei, Wei Hu, and Jacob Steinhardt. "More than a toy: Random matrix models predict how real-world neural representations generalize". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 23549–23588.

[268] Blake E Woodworth, Kumar Kshitij Patel, and Nati Srebro. "Minibatch vs local sgd for heterogeneous distributed learning". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6281–6292.

[269] Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. "Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time". In: *arXiv preprint arXiv:2203.05482* (2022).

[270] John Wright and Yi Ma. *High-Dimensional Data Analysis with Low-Dimensional Models: Principles, Computation, and Applications*. Cambridge University Press, 2022.

[271] John Wright, Yangyu Tao, Zhouchen Lin, Yi Ma, and Heung-Yeung Shum. "Classification via minimum incremental coding length (MICL)". In: *Advances in Neural Information Processing Systems*. 2008, pp. 1633–1640.

[272] Jianlong Wu, Keyu Long, Fei Wang, Chen Qian, Cheng Li, Zhouchen Lin, and Hongbin Zha. "Deep comprehensive correlation mining for image clustering". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 8150–8159.

[273] Qiong Wu, Kaiwen He, and Xu Chen. "Personalized federated learning for intelligent IoT applications: A cloud-edge based framework". In: *IEEE Open Journal of the Computer Society* 1 (2020), pp. 35–44.

[274] Yuxin Wu and Kaiming He. "Group normalization". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 3–19.

[275] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. *Detectron2*. https://github.com/facebookresearch/detectron2. 2019.

[276] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. "Unsupervised feature learning via non-parametric instance discrimination". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3733–3742.

[277] Han Xiao, Kashif Rasul, and Roland Vollgraf. "Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms". In: *arXiv preprint arXiv:1708.07747* (2017).

[278] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. "Unified perceptual parsing for scene understanding". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 418–434.

[279] Junyuan Xie, Ross Girshick, and Ali Farhadi. "Unsupervised deep embedding for clustering analysis". In: *International Conference on Machine Learning*. 2016, pp. 478–487.

[280] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. "Aggregated residual transformations for deep neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1492–1500.

[281] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. "Simmim: A simple framework for masked image modeling". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 9653–9663.

[282] Jianwei Yang, Devi Parikh, and Dhruv Batra. "Joint unsupervised learning of deep representations and image clusters". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 5147–5156.

[283] Yongyi Yang, Zengfeng Huang, and David P Wipf. "Transformers from an Optimization Perspective". In: *Advances in Neural Information Processing Systems*. Ed. by S Koyejo, S Mohamed, A Agarwal, D Belgrave, K Cho, and A Oh. Vol. 35. Curran Associates, Inc., 2022, pp. 36958–36971. URL: `https://proceedings.neurips.cc/paper_files/paper/2022/file/efd1e27afcb94addd03b9e14c8d9f78f-Paper-Conference.pdf`.

[284] Chong You, Chun-Guang Li, Daniel P Robinson, and René Vidal. "Oracle based active set algorithm for scalable elastic net subspace clustering". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 3928–3937.

[285] Yang You, Igor Gitman, and Boris Ginsburg. "Large batch training of convolutional networks". In: *arXiv preprint arXiv:1708.03888* (2017).

[286] Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Ghosh, Akash Bharadwaj, Jessica Zhao, et al. "Opacus: User-friendly differential privacy library in PyTorch". In: *arXiv preprint arXiv:2109.12298* (2021).

[287] Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, et al. "Differentially private fine-tuning of language models". In: *arXiv preprint arXiv:2110.06500* (2021).

[288] Fuxun Yu, Weishan Zhang, Zhuwei Qin, Zirui Xu, Di Wang, Chenchen Liu, Zhi Tian, and Xiang Chen. "Fed2: Feature-Aligned Federated Learning". In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2021, pp. 2066–2074.

[289] Yaodong Yu, Sam Buchanan, Druv Pai, Tianzhe Chu, Ziyang Wu, Shengbang Tong, Benjamin Haeffele, and Yi Ma. "White-box transformers via sparse rate reduction". In: *Advances in Neural Information Processing Systems* 36 (2024).

[290] Yaodong Yu, Kwan Ho Ryan Chan, Chong You, Chaobing Song, and Yi Ma. “Learning diverse and discriminative representations via the principle of maximal coding rate reduction”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 9422–9434.

[291] Yaodong Yu, Tianzhe Chu, Shengbang Tong, Ziyang Wu, Druv Pai, Sam Buchanan, and Yi Ma. “Emergence of Segmentation with Minimalistic White-Box Transformers”. In: *Conference on Parsimony and Learning*. PMLR. 2024, pp. 72–93.

[292] Yaodong Yu, Maziar Sanjabi, Yi Ma, Kamalika Chaudhuri, and Chuan Guo. “ViP: A Differentially Private Foundation Model for Computer Vision”. In: *Forty-first International Conference on Machine Learning*. 2024. URL: `https://openreview.net/forum?id=6aKwVmHQI1`.

[293] Yaodong Yu, Alexander Wei, Sai Praneeth Karimireddy, Yi Ma, and Michael Jordan. “TCT: Convexifying Federated Learning using Bootstrapped Neural Tangent Kernels”. In: *Advances in Neural Information Processing Systems*. Ed. by Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho. 2022. URL: `https://openreview.net/forum?id=jzd2bE5MxW`.

[294] Kai Yue, Richeng Jin, Ryan Pilgrim, Chau-Wai Wong, Dror Baron, and Huaiyu Dai. “Neural Tangent Kernel Empowered Federated Learning”. In: *International Conference on Machine Learning*. PMLR. 2022, pp. 25783–25803.

[295] Luca Zancato, Alessandro Achille, Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto. “Predicting training time without training”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6136–6146.

[296] John Zarka, Louis Thiry, Tomás Angles, and Stéphane Mallat. “Deep network classification by scattering and homotopy dictionary learning”. In: *arXiv preprint arXiv:1910.03561* (2019).

[297] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. “Barlow twins: Self-supervised learning via redundancy reduction”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 12310–12320.

[298] Matthew D Zeiler and Rob Fergus. “Visualizing and understanding convolutional networks”. In: *European Conference on Computer Vision*. Springer. 2014, pp. 818–833.

[299] Yuexiang Zhai, Zitong Yang, Zhenyu Liao, John Wright, and Yi Ma. “Complete dictionary learning via l 4-norm maximization over the orthogonal group”. In: *The Journal of Machine Learning Research* 21.1 (2020), pp. 6622–6689.

[300] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. “Understanding deep learning requires rethinking generalization”. In: *International Conference on Learning Representations*. 2017.

[301] Junjian Zhang, Chun-Guang Li, Chong You, Xianbiao Qi, Honggang Zhang, Jun Guo, and Zhouchen Lin. “Self-supervised convolutional subspace clustering network”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5473–5482.

[302] Tong Zhang, Pan Ji, Mehrtash Harandi, Richard Hartley, and Ian Reid. "Scalable deep k-subspace clustering". In: *Asian Conference on Computer Vision*. Springer. 2018, pp. 466–481.

[303] Tong Zhang, Pan Ji, Mehrtash Harandi, Wenbing Huang, and Hongdong Li. "Neural collaborative subspace clustering". In: *arXiv preprint arXiv:1904.10596* (2019).

[304] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. "Learning deep features for scene recognition using places database". In: *Advances in neural information processing systems* 27 (2014).

[305] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. "Semantic understanding of scenes through the ade20k dataset". In: *International Journal of Computer Vision* 127 (2019), pp. 302–321.

[306] Pan Zhou, Yunqing Hou, and Jiashi Feng. "Deep adversarial subspace clustering". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1596–1604.

Appendix A

Appendix: Representation Learning via Maximal Coding Rate Reduction

This section is organized as follows. We present background and preliminary results for the $\log\det(\cdot)$ function and the coding rate function in Section A.1.1. Then, Section A.1.2 and A.1.3 provide technical lemmas for bounding the coding rate and coding rate reduction functions, respectively. Such lemmas are key results for proving our main theoretical results, which are stated informally in Theorem 2.2.1 and formally in Section A.1.4. Finally, proof of our main theoretical results is provided in Section A.1.5.

Notations Throughout this section, we use $\mathbb{S}^d_{++}$, $\mathbb{R}_+$ and $\mathbb{Z}_{++}$ to denote the set of symmetric positive definite matrices of size $d\times d$, nonnegative real numbers and positive integers, respectively.

A.1 Deferred Proofs

A.1.1 Preliminaries

Properties of the $\log\det(\cdot)$ function.

Lemma A.1.1. *The function* $\log\det(\cdot) : \mathbb{S}^d_{++} \to \mathbb{R}$ *is strictly concave. That is,*

$$\log\det((1-\alpha)\boldsymbol{Z}_1 + \alpha\boldsymbol{Z}_2)) \geq (1-\alpha)\log\det(\boldsymbol{Z}_1) + \alpha\log\det(\boldsymbol{Z}_2)$$

for any $\alpha \in (0,1)$ *and* $\{\boldsymbol{Z}_1, \boldsymbol{Z}_2\} \subseteq \mathbb{S}^d_{++}$, *with equality holds if and only if* $\boldsymbol{Z}_1 = \boldsymbol{Z}_2$.

Proof. Consider an arbitrary line given by $\boldsymbol{Z} = \boldsymbol{Z}_0 + t\Delta\boldsymbol{Z}$ where $\boldsymbol{Z}_0$ and $\Delta\boldsymbol{Z} \neq \boldsymbol{0}$ are symmetric matrices of size $d\times d$. Let $f(t) \doteq \log\det(\boldsymbol{Z}_0 + t\Delta\boldsymbol{Z})$ be a function defined on an interval of values of t for which $\boldsymbol{Z}_0 + t\Delta\boldsymbol{Z} \in \mathbb{S}^d_{++}$. Following the same argument as in [25], we may assume $\boldsymbol{Z}_0 \in \mathbb{S}^d_{++}$ and get

$$f(t) = \log\det\boldsymbol{Z}_0 + \sum_{i=1}^{d}\log(1+t\lambda_i),$$

where $\{\lambda_i\}_{i=1}^d$ are eigenvalues of $\boldsymbol{Z}_0^{-\frac{1}{2}}\Delta \boldsymbol{Z}\boldsymbol{Z}_0^{-\frac{1}{2}}$. The second order derivative of $f(t)$ is given by

$$f''(t) = -\sum_{i=1}^{d} \frac{\lambda_i^2}{(1+t\lambda_i)^2} < 0.$$

Therefore, $f(t)$ is strictly concave along the line $\boldsymbol{Z} = \boldsymbol{Z}_0 + t\Delta\boldsymbol{Z}$. By definition, we conclude that $\log\det(\cdot)$ is strictly concave. □

Properties of the coding rate function. The following properties, also known as the Sylvester's determinant theorem, for the coding rate function are known in the paper [172].

Lemma A.1.2 (Commutative property [172])**.** *For any $\boldsymbol{Z} \in \mathbb{R}^{d\times m}$ we have*

$$R(\boldsymbol{Z}, \epsilon) \doteq \frac{1}{2}\log\det\left(\boldsymbol{I} + \frac{d}{m\epsilon^2}\boldsymbol{Z}\boldsymbol{Z}^\top\right) = \frac{1}{2}\log\det\left(\boldsymbol{I} + \frac{d}{m\epsilon^2}\boldsymbol{Z}^\top\boldsymbol{Z}\right).$$

Lemma A.1.3 (Invariant property [172])**.** *For any $\boldsymbol{Z} \in \mathbb{R}^{d\times m}$ and any orthogonal matrices $\boldsymbol{U} \in \mathbb{R}^{d\times d}$ and $\boldsymbol{V} \in \mathbb{R}^{m\times m}$ we have*

$$R(\boldsymbol{Z}, \epsilon) = R(\boldsymbol{U}\boldsymbol{Z}\boldsymbol{V}^\top, \epsilon).$$

A.1.2 Lower and Upper Bounds for Coding Rate

The following result provides an upper and a lower bound on the coding rate of $\boldsymbol{Z}$ as a function of the coding rate for its components $\{\boldsymbol{Z}_j\}_{j=1}^k$. The lower bound is tight when all the components $\{\boldsymbol{Z}_j\}_{j=1}^k$ have the same covariance (assuming that they have zero mean). The upper bound is tight when the components $\{\boldsymbol{Z}_j\}_{j=1}^k$ are pair-wise orthogonal.

Lemma A.1.4. *For any $\{\boldsymbol{Z}_j \in \mathbb{R}^{d\times m_j}\}_{j=1}^k$ and any $\epsilon > 0$, let $\boldsymbol{Z} = [\boldsymbol{Z}_1, \cdots, \boldsymbol{Z}_k] \in \mathbb{R}^{d\times m}$ with $m = \sum_{j=1}^k m_j$. We have*

$$\begin{aligned}\sum_{j=1}^{k}\frac{m_j}{2}\log\det\left(\boldsymbol{I} + \frac{d}{m_j\epsilon^2}\boldsymbol{Z}_j\boldsymbol{Z}_j^\top\right) &\le \frac{m}{2}\log\det\left(\boldsymbol{I} + \frac{d}{m\epsilon^2}\boldsymbol{Z}\boldsymbol{Z}^\top\right) \\ &\le \sum_{j=1}^{k}\frac{m}{2}\log\det\left(\boldsymbol{I} + \frac{d}{m\epsilon^2}\boldsymbol{Z}_j\boldsymbol{Z}_j^\top\right),\end{aligned} \tag{A.1}$$

where the first equality holds if and only if

$$\frac{\boldsymbol{Z}_1\boldsymbol{Z}_1^\top}{m_1} = \frac{\boldsymbol{Z}_2\boldsymbol{Z}_2^\top}{m_2} = \cdots = \frac{\boldsymbol{Z}_k\boldsymbol{Z}_k^\top}{m_k},$$

and the second equality holds if and only if $\boldsymbol{Z}_{j_1}^\top\boldsymbol{Z}_{j_2} = \boldsymbol{0}$ for all $1 \le j_1 < j_2 \le k$.

Proof. By Lemma A.1.1, $\log\det(\cdot)$ is strictly concave. Therefore,

$$\log\det\left(\sum_{j=1}^{k}\alpha_j \boldsymbol{S}_j\right) \geq \sum_{j=1}^{k}\alpha_j \log\det(\boldsymbol{S}_j), \text{ for all } \{\alpha_j > 0\}_{j=1}^{k}, \sum_{j=1}^{k}\alpha_j = 1 \text{ and } \{\boldsymbol{S}_j \in \mathbb{S}_{++}^{d}\}_{j=1}^{k},$$

where equality holds if and only if $\boldsymbol{S}_1 = \boldsymbol{S}_2 = \cdots = \boldsymbol{S}_k$. Take $\alpha_j = \frac{m_j}{m}$ and $\boldsymbol{S}_j = \boldsymbol{I} + \frac{d}{m_j\epsilon^2}\boldsymbol{Z}_j\boldsymbol{Z}_j^\top$, we get

$$\frac{m}{2}\log\det\left(\boldsymbol{I} + \frac{d}{m\epsilon^2}\boldsymbol{Z}\boldsymbol{Z}^\top\right) \geq \sum_{j=1}^{k}\frac{m_j}{2}\log\det\left(\boldsymbol{I} + \frac{d}{m_j\epsilon^2}\boldsymbol{Z}_j\boldsymbol{Z}_j^\top\right),$$

with equality holds if and only if $\frac{\boldsymbol{Z}_1\boldsymbol{Z}_1^\top}{m_1} = \cdots = \frac{\boldsymbol{Z}_k\boldsymbol{Z}_k^\top}{m_k}$. This proves the lower bound in (A.1).

We now prove the upper bound. By the strict concavity of $\log\det(\cdot)$, we have

$$\log\det(\boldsymbol{Q}) \leq \log\det(\boldsymbol{S}) + \langle\nabla\log\det(\boldsymbol{S}), \boldsymbol{Q} - \boldsymbol{S}\rangle, \text{ for all } \{\boldsymbol{Q}, \boldsymbol{S}\} \subseteq \mathbb{S}_{++}^{m},$$

where equality holds if and only if $\boldsymbol{Q} = \boldsymbol{S}$. Plugging in $\nabla\log\det(\boldsymbol{S}) = \boldsymbol{S}^{-1}$ (see e.g., [25]) and $\boldsymbol{S}^{-1} = (\boldsymbol{S}^{-1})^\top$ gives

$$\log\det(\boldsymbol{Q}) \leq \log\det(\boldsymbol{S}) + \mathrm{tr}(\boldsymbol{S}^{-1}\boldsymbol{Q}) - m. \tag{A.2}$$

We now take

$$\boldsymbol{Q} = \boldsymbol{I} + \frac{d}{m\epsilon^2}\boldsymbol{Z}^\top\boldsymbol{Z} = \boldsymbol{I} + \frac{d}{m\epsilon^2}\begin{bmatrix}\boldsymbol{Z}_1^\top\boldsymbol{Z}_1 & \boldsymbol{Z}_1^\top\boldsymbol{Z}_2 & \cdots & \boldsymbol{Z}_1^\top\boldsymbol{Z}_k \\ \boldsymbol{Z}_2^\top\boldsymbol{Z}_1 & \boldsymbol{Z}_2^\top\boldsymbol{Z}_2 & \cdots & \boldsymbol{Z}_2^\top\boldsymbol{Z}_2 \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{Z}_k^\top\boldsymbol{Z}_1 & \boldsymbol{Z}_k^\top\boldsymbol{Z}_2 & \cdots & \boldsymbol{Z}_k^\top\boldsymbol{Z}_k\end{bmatrix}, \text{ and} \tag{A.3}$$

$$\boldsymbol{S} = \boldsymbol{I} + \frac{d}{m\epsilon^2}\begin{bmatrix}\boldsymbol{Z}_1^\top\boldsymbol{Z}_1 & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{Z}_2^\top\boldsymbol{Z}_2 & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{Z}_k^\top\boldsymbol{Z}_k\end{bmatrix}.$$

From the property of determinant for block diagonal matrix, we have

$$\log\det(\boldsymbol{S}) = \sum_{j=1}^{k}\log\det\left(\boldsymbol{I} + \frac{d}{m\epsilon^2}\boldsymbol{Z}_j^\top\boldsymbol{Z}_j\right). \tag{A.4}$$

Also, note that

$$\begin{aligned}&\mathrm{tr}(\boldsymbol{S}^{-1}\boldsymbol{Q}) \\ &= \mathrm{tr}\begin{bmatrix}(\boldsymbol{I} + \frac{d}{m\epsilon^2}\boldsymbol{Z}_1^\top\boldsymbol{Z}_1)^{-1}(\boldsymbol{I} + \frac{d}{m\epsilon^2}\boldsymbol{Z}_1^\top\boldsymbol{Z}_1) & \cdots & (\boldsymbol{I} + \frac{d}{m\epsilon^2}\boldsymbol{Z}_1^\top\boldsymbol{Z}_1)^{-1}(\boldsymbol{I} + \frac{d}{m\epsilon^2}\boldsymbol{Z}_1^\top\boldsymbol{Z}_k) \\ \vdots & \ddots & \vdots \\ (\boldsymbol{I} + \frac{d}{m\epsilon^2}\boldsymbol{Z}_k^\top\boldsymbol{Z}_k)^{-1}(\boldsymbol{I} + \frac{d}{m\epsilon^2}\boldsymbol{Z}_k^\top\boldsymbol{Z}_1) & \cdots & (\boldsymbol{I} + \frac{d}{m\epsilon^2}\boldsymbol{Z}_k^\top\boldsymbol{Z}_k)^{-1}(\boldsymbol{I} + \frac{d}{m\epsilon^2}\boldsymbol{Z}_k^\top\boldsymbol{Z}_k)\end{bmatrix} \\ &= \mathrm{tr}\begin{bmatrix}\boldsymbol{I} & \cdots & * \\ \vdots & \ddots & \vdots \\ * & \cdots & \boldsymbol{I}\end{bmatrix} = m,\end{aligned} \tag{A.5}$$

where "*" denotes nonzero quantities that are irrelevant for the purpose of computing the trace. Plugging (A.4) and (A.5) back in (A.2) gives

$$\frac{m}{2}\log\det\left(\boldsymbol{I}+\frac{d}{m\epsilon^2}\boldsymbol{Z}^\top\boldsymbol{Z}\right)\leq\sum_{j=1}^{k}\frac{m}{2}\log\det\left(\boldsymbol{I}+\frac{d}{m\epsilon^2}\boldsymbol{Z}_j^\top\boldsymbol{Z}_j\right),$$

where the equality holds if and only if $\boldsymbol{Q}=\boldsymbol{S}$, which by the formulation in (A.3), holds if and only if $\boldsymbol{Z}_{j_1}^\top\boldsymbol{Z}_{j_2}=\mathbf{0}$ for all $1\leq j_1<j_2\leq k$. Further using the result in Lemma A.1.2 gives

$$\frac{m}{2}\log\det\left(\boldsymbol{I}+\frac{d}{m\epsilon^2}\boldsymbol{Z}\boldsymbol{Z}^\top\right)\leq\sum_{j=1}^{k}\frac{m}{2}\log\det\left(\boldsymbol{I}+\frac{d}{m\epsilon^2}\boldsymbol{Z}_j\boldsymbol{Z}_j^\top\right),$$

which produces the upper bound in (A.1). □

A.1.3 An Upper Bound on Coding Rate Reduction

We may now provide an upper bound on the coding rate reduction $\Delta R(\boldsymbol{Z},\boldsymbol{\Pi},\epsilon)$ (defined in (2.8)) in terms of its individual components $\{\boldsymbol{Z}_j\}_{j=1}^k$.

Lemma A.1.5. *For any $\boldsymbol{Z}\in\mathbb{R}^{d\times m},\boldsymbol{\Pi}\in\Omega$ and $\epsilon>0$, let $\boldsymbol{Z}_j\in\mathbb{R}^{d\times m_j}$ be $\boldsymbol{Z}\boldsymbol{\Pi}_j$ with zero columns removed. We have*

$$\Delta R(\boldsymbol{Z},\boldsymbol{\Pi},\epsilon)\leq\sum_{j=1}^{k}\frac{1}{2m}\log\left(\frac{\det^m\left(\boldsymbol{I}+\frac{d}{m\epsilon^2}\boldsymbol{Z}_j\boldsymbol{Z}_j^\top\right)}{\det^{m_j}\left(\boldsymbol{I}+\frac{d}{m_j\epsilon^2}\boldsymbol{Z}_j\boldsymbol{Z}_j^\top\right)}\right),\tag{A.6}$$

with equality holds if and only if $\boldsymbol{Z}_{j_1}^\top\boldsymbol{Z}_{j_2}=\mathbf{0}$ for all $1\leq j_1<j_2\leq k$.

Proof. From (2.4), (2.5) and (2.6), we have

$$\begin{aligned}
&\Delta R(\boldsymbol{Z}, \boldsymbol{\Pi}, \epsilon)\\
&= R(\boldsymbol{Z}, \epsilon) - R^c(\boldsymbol{Z}, \epsilon \mid \boldsymbol{\Pi})\\
&= \frac{1}{2}\log\left(\det\left(\boldsymbol{I} + \frac{d}{m\epsilon^2}\boldsymbol{Z}\boldsymbol{Z}^\top\right)\right) - \sum_{j=1}^{k}\left\{\frac{\mathsf{tr}(\boldsymbol{\Pi}_j)}{2m}\log\left(\det\left(\boldsymbol{I} + d\frac{\boldsymbol{Z}\boldsymbol{\Pi}_j\boldsymbol{Z}^\top}{\mathsf{tr}(\boldsymbol{\Pi}_j)\epsilon^2}\right)\right)\right\}\\
&= \frac{1}{2}\log\left(\det\left(\boldsymbol{I} + \frac{d}{m\epsilon^2}\boldsymbol{Z}\boldsymbol{Z}^\top\right)\right) - \sum_{j=1}^{k}\left\{\frac{m_j}{2m}\log\left(\det\left(\boldsymbol{I} + d\frac{\boldsymbol{Z}_j\boldsymbol{Z}_j^\top}{m_j\epsilon^2}\right)\right)\right\}\\
&\leq \sum_{j=1}^{k}\frac{1}{2}\log\left(\det\left(\boldsymbol{I} + \frac{d}{m\epsilon^2}\boldsymbol{Z}_j\boldsymbol{Z}_j^\top\right)\right) - \sum_{j=1}^{k}\left\{\frac{m_j}{2m}\log\left(\det\left(\boldsymbol{I} + d\frac{\boldsymbol{Z}_j\boldsymbol{Z}_j^\top}{m_j\epsilon^2}\right)\right)\right\}\\
&= \sum_{j=1}^{k}\frac{1}{2m}\log\left(\det{}^m\left(\boldsymbol{I} + \frac{d}{m\epsilon^2}\boldsymbol{Z}_j\boldsymbol{Z}_j^\top\right)\right) - \sum_{j=1}^{k}\left\{\frac{1}{2m}\log\left(\det{}^{m_j}\left(\boldsymbol{I} + d\frac{\boldsymbol{Z}_j\boldsymbol{Z}_j^\top}{m_j\epsilon^2}\right)\right)\right\}\\
&= \sum_{j=1}^{k}\frac{1}{2m}\log\left(\frac{\det^m\left(\boldsymbol{I} + \frac{d}{m\epsilon^2}\boldsymbol{Z}_j\boldsymbol{Z}_j^\top\right)}{\det^{m_j}\left(\boldsymbol{I} + \frac{d}{m_j\epsilon^2}\boldsymbol{Z}_j\boldsymbol{Z}_j^\top\right)}\right),
\end{aligned}$$

where the inequality follows from the upper bound in Lemma A.1.4, and that the equality holds if and only if $\boldsymbol{Z}_{j_1}^\top\boldsymbol{Z}_{j_2} = \mathbf{0}$ for all $1 \leq j_1 < j_2 \leq k$. □

A.1.4 Main Results: Properties of Maximal Coding Rate Reduction

We now present our main theoretical results. The following theorem states that for any fixed encoding of the partition $\boldsymbol{\Pi}$, the coding rate reduction is maximized by data $\boldsymbol{Z}$ that is maximally discriminative between different classes and is diverse within each of the classes. This result holds provided that the sum of rank for different classes is small relative to the ambient dimension, and that ϵ is small.

Theorem A.1.6. *Let $\boldsymbol{\Pi} = \{\boldsymbol{\Pi}_j \in \mathbb{R}^{m\times m}\}_{j=1}^k$ with $\{\boldsymbol{\Pi}_j \geq \mathbf{0}\}_{j=1}^k$ and $\boldsymbol{\Pi}_1 + \cdots + \boldsymbol{\Pi}_k = \boldsymbol{I}$ be a given set of diagonal matrices whose diagonal entries encode the membership of the m samples in the k classes. Given any $\epsilon > 0$, $d > 0$ and $\{d \geq d_j > 0\}_{j=1}^k$, consider the optimization problem*

$$\begin{aligned}
\boldsymbol{Z}^* \in &\operatorname*{arg\,max}_{\boldsymbol{Z}\in\mathbb{R}^{d\times m}} \Delta R(\boldsymbol{Z}, \boldsymbol{\Pi}, \epsilon)\\
&\textit{s.t. } \|\boldsymbol{Z}\boldsymbol{\Pi}_j\|_F^2 = \mathsf{tr}(\boldsymbol{\Pi}_j),\ \mathsf{rank}(\boldsymbol{Z}\boldsymbol{\Pi}_j) \leq d_j,\ \forall j \in \{1, \ldots, k\}.
\end{aligned} \tag{A.7}$$

Under the conditions

- (Large ambient dimension) $d \geq \sum_{j=1}^k d_j$, *and*
- (High coding precision) $\epsilon^4 < \min_{j\in\{1,\ldots,k\}}\left\{\frac{1}{16}\frac{\mathsf{tr}(\boldsymbol{\Pi}_j)}{m}\frac{d^2}{d_j^2}\right\}$,

the optimal solution $\boldsymbol{Z}^*$ satisfies

- (Between-class discriminative) $(\boldsymbol{Z}_{j_1}^*)^\top \boldsymbol{Z}_{j_2}^* = \mathbf{0}$ *for all* $1 \le j_1 < j_2 \le k$*, i.e.,* $\boldsymbol{Z}_{j_1}^*$ *and* $\boldsymbol{Z}_{j_2}^*$ *lie in orthogonal subspaces, and*
- (Within-class diverse) *For each* $j \in \{1, \dots, k\}$*, the rank of* $\boldsymbol{Z}_j^*$ *is equal to* d_j *and either all singular values of* $\boldsymbol{Z}_j^*$ *are equal to* $\frac{\operatorname{tr}(\boldsymbol{\Pi}_j)}{d_j}$*, or the* $d_j - 1$ *largest singular values of* $\boldsymbol{Z}_j^*$ *are equal and have value larger than* $\frac{\operatorname{tr}(\boldsymbol{\Pi}_j)}{d_j}$*,*

where $\boldsymbol{Z}_j^* \in \mathbb{R}^{d \times \operatorname{tr}(\boldsymbol{\Pi}_j)}$ *denotes* $\boldsymbol{Z}^*\boldsymbol{\Pi}_j$ *with zero columns removed.*

A.1.5 Proof of Main Results

We start with presenting a lemma that will be used in the proof to Theorem A.1.6. Refer to [264] for detailed proof.

Lemma A.1.7. *Given any twice differentiable* $f : \mathbb{R}_+ \to \mathbb{R}$*, integer* $r \in \mathbb{Z}_{++}$ *and* $c = (m/K) \in \mathbb{R}_+$*, consider the optimization problem*

$$\begin{aligned} &\max_{\boldsymbol{x}} \sum_{p=1}^{r} f(x_p) \\ &\textit{s.t. } \boldsymbol{x} = [x_1, \dots, x_r] \in \mathbb{R}_+^r,\ x_1 \ge x_2 \ge \dots \ge x_r,\ \textit{and} \sum_{p=1}^{r} x_p = c. \end{aligned} \tag{A.8}$$

Let $\boldsymbol{x}^*$ *be an arbitrary global solution to* (A.8)*. If the function* $f(\cdot)$ *is defined as*

$$f(x; \alpha, m, K) = \log(1 + \alpha x) - \frac{1}{K}\log(1 + K\alpha x), \tag{A.9}$$

where $\alpha \ge 4r\sqrt{K}/m > 0$ *and* $m, K > 0$ *are constant. then we have either*

- $\boldsymbol{x}^* = [\frac{c}{r}, \dots, \frac{c}{r}]$*, or*
- $\boldsymbol{x}^* = [x_H, \dots, x_H, x_L]$ *for some* $x_H \in (\frac{c}{r}, \frac{c}{r-1})$ *and* $x_L > 0$*.*

Next we provide the missing proofs for Theorem A.1.6.

Proof of Theorem A.1.6. Without loss of generality, let $\boldsymbol{Z}^* = [\boldsymbol{Z}_1^*, \dots, \boldsymbol{Z}_k^*]$ be the optimal solution of problem (A.7).

To show that $\boldsymbol{Z}_j^*, j \in \{1, \dots, k\}$ are pairwise orthogonal, suppose for the purpose of arriving at a contradiction that $(\boldsymbol{Z}_{j_1}^*)^\top \boldsymbol{Z}_{j_2}^* \ne \mathbf{0}$ for some $1 \le j_1 < j_2 \le k$. By using Lemma A.1.5, the strict inequality in (A.6) holds for the optimal solution $\boldsymbol{Z}^*$. That is,

$$\Delta R(\boldsymbol{Z}^*, \boldsymbol{\Pi}, \epsilon) < \sum_{j=1}^{k} \frac{1}{2m} \log\left(\frac{\det^m\left(\boldsymbol{I} + \frac{d}{m\epsilon^2}\boldsymbol{Z}_j^*(\boldsymbol{Z}_j^*)^\top\right)}{\det^{m_j}\left(\boldsymbol{I} + \frac{d}{m_j\epsilon^2}\boldsymbol{Z}_j^*(\boldsymbol{Z}_j^*)^\top\right)} \right). \tag{A.10}$$

On the other hand, since $\sum_{j=1}^{k} d_j \le d$, there exists $\{\boldsymbol{U}_j' \in \mathbb{R}^{d\times d_j}\}_{j=1}^{k}$ such that the columns of the matrix $[\boldsymbol{U}_1', \dots, \boldsymbol{U}_k']$ are orthonormal. Denote $\boldsymbol{Z}_j^* = \boldsymbol{U}_j^* \boldsymbol{\Sigma}_j^* (\boldsymbol{V}_j^*)^\top$ the compact SVD of $\boldsymbol{Z}_j^*$, and let

$$\boldsymbol{Z}' = [\boldsymbol{Z}_1', \dots, \boldsymbol{Z}_k'], \text{ where } \boldsymbol{Z}_j' = \boldsymbol{U}_j' \boldsymbol{\Sigma}_j^* (\boldsymbol{V}_j^*)^\top.$$

It follows that

$$(\boldsymbol{Z}_{j_1}')^\top \boldsymbol{Z}_{j_2}' = \boldsymbol{V}_{j_1}^* \boldsymbol{\Sigma}_{j_1}^* (\boldsymbol{U}_{j_1}')^\top \boldsymbol{U}_{j_2}' \boldsymbol{\Sigma}_{j_2}^* (\boldsymbol{V}_{j_2}^*)^\top = \boldsymbol{V}_{j_1}^* \boldsymbol{\Sigma}_{j_1}^* \boldsymbol{0} \boldsymbol{\Sigma}_{j_2}^* (\boldsymbol{V}_{j_2}^*)^\top = \boldsymbol{0} \text{ for all } 1 \le j_1 < j_2 \le k.$$

That is, the matrices $\boldsymbol{Z}_1', \dots, \boldsymbol{Z}_k'$ are pairwise orthogonal. Applying Lemma A.1.5 for $\boldsymbol{Z}'$ gives

$$\begin{aligned}\Delta R(\boldsymbol{Z}', \boldsymbol{\Pi}, \epsilon) &= \sum_{j=1}^{k} \frac{1}{2m} \log \left(\frac{\det^m \left(\boldsymbol{I} + \frac{d}{m\epsilon^2} \boldsymbol{Z}_j' (\boldsymbol{Z}_j')^\top\right)}{\det^{m_j} \left(\boldsymbol{I} + \frac{d}{m_j\epsilon^2} \boldsymbol{Z}_j' (\boldsymbol{Z}_j')^\top\right)} \right) \\ &= \sum_{j=1}^{k} \frac{1}{2m} \log \left(\frac{\det^m \left(\boldsymbol{I} + \frac{d}{m\epsilon^2} \boldsymbol{Z}_j^* (\boldsymbol{Z}_j^*)^\top\right)}{\det^{m_j} \left(\boldsymbol{I} + \frac{d}{m_j\epsilon^2} \boldsymbol{Z}_j^* (\boldsymbol{Z}_j^*)^\top\right)} \right),\end{aligned} \tag{A.11}$$

where the second equality follows from Lemma A.1.3. Comparing (A.10) and (A.11) gives $\Delta R(\boldsymbol{Z}', \boldsymbol{\Pi}, \epsilon) > \Delta R(\boldsymbol{Z}^*, \boldsymbol{\Pi}, \epsilon)$, which contradicts the optimality of $\boldsymbol{Z}^*$. Therefore, we must have

$$(\boldsymbol{Z}_{j_1}^*)^\top \boldsymbol{Z}_{j_2}^* = \boldsymbol{0} \text{ for all } 1 \le j_1 < j_2 \le k.$$

Moreover, from Lemma A.1.3 we have

$$\Delta R(\boldsymbol{Z}^*, \boldsymbol{\Pi}, \epsilon) = \sum_{j=1}^{k} \frac{1}{2m} \log \left(\frac{\det^m \left(\boldsymbol{I} + \frac{d}{m\epsilon^2} \boldsymbol{Z}_j^* (\boldsymbol{Z}_j^*)^\top\right)}{\det^{m_j} \left(\boldsymbol{I} + \frac{d}{m_j\epsilon^2} \boldsymbol{Z}_j^* (\boldsymbol{Z}_j^*)^\top\right)} \right). \tag{A.12}$$

We now prove the result concerning the singular values of $\boldsymbol{Z}_j^*$. To start with, we claim that the following result holds:

$$\boldsymbol{Z}_j^* \in \arg\max_{\boldsymbol{Z}_j} \log \left(\frac{\det^m \left(\boldsymbol{I} + \frac{d}{m\epsilon^2} \boldsymbol{Z}_j \boldsymbol{Z}_j^\top\right)}{\det^{m_j} \left(\boldsymbol{I} + \frac{d}{m_j\epsilon^2} \boldsymbol{Z}_j \boldsymbol{Z}_j^\top\right)} \right) \text{ s.t. } \|\boldsymbol{Z}_j\|_F^2 = m_j, \, \mathsf{rank}(\boldsymbol{Z}_j) \le d_j. \tag{A.13}$$

To see why (A.13) holds, suppose that there exists $\widetilde{\boldsymbol{Z}}_j$ such that $\|\widetilde{\boldsymbol{Z}}_j\|_F^2 = m_j$, $\mathsf{rank}(\widetilde{\boldsymbol{Z}}_j) \le d_j$ and

$$\log \left(\frac{\det^m \left(\boldsymbol{I} + \frac{d}{m\epsilon^2} \widetilde{\boldsymbol{Z}}_j \widetilde{\boldsymbol{Z}}_j^\top\right)}{\det^{m_j} \left(\boldsymbol{I} + \frac{d}{m_j\epsilon^2} \widetilde{\boldsymbol{Z}}_j \widetilde{\boldsymbol{Z}}_j^\top\right)} \right) > \log \left(\frac{\det^m \left(\boldsymbol{I} + \frac{d}{m\epsilon^2} \boldsymbol{Z}_j^* (\boldsymbol{Z}_j^*)^\top\right)}{\det^{m_j} \left(\boldsymbol{I} + \frac{d}{m_j\epsilon^2} \boldsymbol{Z}_j^* (\boldsymbol{Z}_j^*)^\top\right)} \right). \tag{A.14}$$

Denote $\widetilde{\boldsymbol{Z}}_j = \widetilde{\boldsymbol{U}}_j \widetilde{\boldsymbol{\Sigma}}_j \widetilde{\boldsymbol{V}}_j^\top$ the compact SVD of $\widetilde{\boldsymbol{Z}}_j$ and let

$$\boldsymbol{Z}' = [\boldsymbol{Z}_1^*, \dots, \boldsymbol{Z}_{j-1}^*, \boldsymbol{Z}_j', \boldsymbol{Z}_{j+1}^*, \dots, \boldsymbol{Z}_k^*], \text{ where } \boldsymbol{Z}_j' := \boldsymbol{U}_j^* \widetilde{\boldsymbol{\Sigma}}_j \widetilde{\boldsymbol{V}}_j^\top.$$

Note that $\|\boldsymbol{Z}_j'\|_F^2 = m_j$, $\mathsf{rank}(\boldsymbol{Z}_j') \leq d_j$ and $(\boldsymbol{Z}_j')^\top \boldsymbol{Z}_{j'}^* = \boldsymbol{0}$ for all $j' \neq j$. It follows that $\boldsymbol{Z}'$ is a feasible solution to (A.7) and that the components of $\boldsymbol{Z}'$ are pairwise orthogonal. By using Lemma A.1.5, Lemma A.1.3 and (A.14) we have

$$\begin{aligned}
&\Delta R(\boldsymbol{Z}', \boldsymbol{\Pi}, \epsilon)\\
&= \frac{1}{2m}\log\left(\frac{\det^m\left(\boldsymbol{I} + \frac{d}{m\epsilon^2}\boldsymbol{Z}_j'(\boldsymbol{Z}_j')^\top\right)}{\det^{m_j}\left(\boldsymbol{I} + \frac{d}{m_j\epsilon^2}\boldsymbol{Z}_j'(\boldsymbol{Z}_j')^\top\right)}\right) + \sum_{j'\neq j}\frac{1}{2m}\log\left(\frac{\det^m\left(\boldsymbol{I} + \frac{d}{m\epsilon^2}\boldsymbol{Z}_{j'}^*(\boldsymbol{Z}_{j'}^*)^\top\right)}{\det^{m_{j'}}\left(\boldsymbol{I} + \frac{d}{m_{j'}\epsilon^2}\boldsymbol{Z}_{j'}^*(\boldsymbol{Z}_{j'}^*)^\top\right)}\right)\\
&= \frac{1}{2m}\log\left(\frac{\det^m\left(\boldsymbol{I} + \frac{d}{m\epsilon^2}\widetilde{\boldsymbol{Z}}_j(\widetilde{\boldsymbol{Z}}_j)^\top\right)}{\det^{m_j}\left(\boldsymbol{I} + \frac{d}{m_j\epsilon^2}\widetilde{\boldsymbol{Z}}_j(\widetilde{\boldsymbol{Z}}_j)^\top\right)}\right) + \sum_{j'\neq j}\frac{1}{2m}\log\left(\frac{\det^m\left(\boldsymbol{I} + \frac{d}{m\epsilon^2}\boldsymbol{Z}_{j'}^*(\boldsymbol{Z}_{j'}^*)^\top\right)}{\det^{m_{j'}}\left(\boldsymbol{I} + \frac{d}{m_{j'}\epsilon^2}\boldsymbol{Z}_{j'}^*(\boldsymbol{Z}_{j'}^*)^\top\right)}\right)\\
&> \frac{1}{2m}\log\left(\frac{\det^m\left(\boldsymbol{I} + \frac{d}{m\epsilon^2}\boldsymbol{Z}_j^*(\boldsymbol{Z}_j^*)^\top\right)}{\det^{m_j}\left(\boldsymbol{I} + \frac{d}{m_j\epsilon^2}\boldsymbol{Z}_j^*(\boldsymbol{Z}_j^*)^\top\right)}\right) + \sum_{j'\neq j}\frac{1}{2m}\log\left(\frac{\det^m\left(\boldsymbol{I} + \frac{d}{m\epsilon^2}\boldsymbol{Z}_{j'}^*(\boldsymbol{Z}_{j'}^*)^\top\right)}{\det^{m_{j'}}\left(\boldsymbol{I} + \frac{d}{m_{j'}\epsilon^2}\boldsymbol{Z}_{j'}^*(\boldsymbol{Z}_{j'}^*)^\top\right)}\right)\\
&= \sum_{j=1}^{k}\frac{1}{2m}\log\left(\frac{\det^m\left(\boldsymbol{I} + \frac{d}{m\epsilon^2}\boldsymbol{Z}_j^*(\boldsymbol{Z}_j^*)^\top\right)}{\det^{m_j}\left(\boldsymbol{I} + \frac{d}{m_j\epsilon^2}\boldsymbol{Z}_j^*(\boldsymbol{Z}_j^*)^\top\right)}\right).
\end{aligned}$$

Combining it with (A.12) shows $\Delta R(\boldsymbol{Z}', \boldsymbol{\Pi}, \epsilon) > \Delta R(\boldsymbol{Z}^*, \boldsymbol{\Pi}, \epsilon)$, contradicting the optimality of $\boldsymbol{Z}^*$. Therefore, the result in (A.13) holds.

Observe that the optimization problem in (A.13) depends on $\boldsymbol{Z}_j$ only through its singular values. That is, by letting $\boldsymbol{\sigma}_j := [\sigma_{1,j}, \ldots, \sigma_{\min(m_j,d),j}]$ be the singular values of $\boldsymbol{Z}_j$, we have

$$\log\left(\frac{\det^m\left(\boldsymbol{I} + \frac{d}{m\epsilon^2}\boldsymbol{Z}_j\boldsymbol{Z}_j^\top\right)}{\det^{m_j}\left(\boldsymbol{I} + \frac{d}{m_j\epsilon^2}\boldsymbol{Z}_j\boldsymbol{Z}_j^\top\right)}\right) = \sum_{p=1}^{\min\{m_j,d\}}\log\left(\frac{(1+\frac{d}{m\epsilon^2}\sigma_{p,j}^2)^m}{(1+\frac{d}{m_j\epsilon^2}\sigma_{p,j}^2)^{m_j}}\right),$$

also, we have

$$\|\boldsymbol{Z}_j\|_F^2 = \sum_{p=1}^{\min\{m_j,d\}}\sigma_{p,j}^2 \text{ and } \mathsf{rank}(\boldsymbol{Z}_j) = \|\boldsymbol{\sigma}_j\|_0.$$

Using these relations, (A.13) is equivalent to

$$\begin{aligned}
\max_{\boldsymbol{\sigma}_j \in \mathbb{R}_+^{\min\{m_j,d\}}} \quad & \sum_{p=1}^{\min\{m_j,d\}}\log\left(\frac{(1+\frac{d}{m\epsilon^2}\sigma_{p,j}^2)^m}{(1+\frac{d}{m_j\epsilon^2}\sigma_{p,j}^2)^{m_j}}\right)\\
\text{s.t.} \quad & \sum_{p=1}^{\min\{m_j,d\}}\sigma_{p,j}^2 = m_j, \text{ and } \mathsf{rank}(\boldsymbol{Z}_j) = \|\boldsymbol{\sigma}_j\|_0
\end{aligned} \tag{A.15}$$

Let $\boldsymbol{\sigma}_j^* = [\sigma_{1,j}^*, \ldots, \sigma_{\min\{m_j,d\},j}^*]$ be an optimal solution to (A.15). Without loss of generality we assume that the entries of $\boldsymbol{\sigma}_j^*$ are sorted in descending order. It follows that

$$\sigma_{p,j}^* = 0 \text{ for all } p > d_j,$$

and

$$[\sigma_{1,j}^*, \dots, \sigma_{d_j,j}^*] = \underset{\substack{[\sigma_{1,j},\dots,\sigma_{d_j,j}] \in \mathbb{R}_+^{d_j} \\ \sigma_{1,j} \geq \dots \geq \sigma_{d_j,j}}}{\arg\max} \sum_{p=1}^{d_j} \log\left(\frac{(1+\frac{d}{m\epsilon^2}\sigma_{p,j}^2)^m}{(1+\frac{d}{m_j\epsilon^2}\sigma_{p,j}^2)^{m_j}}\right) \quad \text{s.t.} \sum_{p=1}^{d_j} \sigma_{p,j}^2 = m_j. \tag{A.16}$$

Then we define

$$f(x; d, \epsilon, m_j, m) = \log\left(\frac{(1+\frac{d}{m\epsilon^2}x)^m}{(1+\frac{d}{m_j\epsilon^2}x)^{m_j}}\right),$$

and rewrite (A.16) as

$$\max_{\substack{[x_1,\dots,x_{d_j}] \in \mathbb{R}_+^{d_j} \\ x_1 \geq \dots \geq x_{d_j}}} \sum_{p=1}^{d_j} f(x_p; d, \epsilon, m_j, m) \quad \text{s.t.} \sum_{p=1}^{d_j} x_p = m_j. \tag{A.17}$$

We compute the first and second derivative for f with respect to x, which are given by

$$f'(x; d, \epsilon, m_j, m) = \frac{d^2 x(m - m_j)}{(dx + m\epsilon^2)(dx + m_j\epsilon^2)},$$
$$f''(x; d, \epsilon, m_j, m) = \frac{d^2(m - m_j)(m m_j \epsilon^4 - d^2 x^2)}{(dx + m\epsilon^2)^2(dx + m_j\epsilon^2)^2}.$$

Therefore, we now apply Lemma A.1.7 and conclude that the unique optimal solution to (A.17) is either

- $\boldsymbol{x}^* = [\frac{m_j}{d_j}, \dots, \frac{m_j}{d_j}]$, or
- $\boldsymbol{x}^* = [x_H, \dots, x_H, x_L]$ for some $x_H \in (\frac{m_j}{d_j}, \frac{m_j}{d_j - 1})$ and $x_L > 0$.

Equivalently, we have either

- $[\sigma_{1,j}^*, \dots, \sigma_{d_j,j}^*] = \left[\sqrt{\frac{m_j}{d_j}}, \dots, \sqrt{\frac{m_j}{d_j}}\right]$, or
- $[\sigma_{1,j}^*, \dots, \sigma_{d_j,j}^*] = [\sigma_H, \dots, \sigma_H, \sigma_L]$ for some $\sigma_H \in \left(\sqrt{\frac{m_j}{d_j}}, \sqrt{\frac{m_j}{d_j - 1}}\right)$ and $\sigma_L > 0$,

as claimed. □

A.2 Additional Simulations and Experiments

A.2.1 Simulations - Verifying Diversity Promoting Properties of MCR^2

As proved in Theorem A.1.6, the proposed MCR^2 objective promotes within-class diversity. In this section, we use simulated data to verify the diversity promoting property of MCR^2. As shown in Table A.1, we calculate our proposed MCR^2 objective on simulated data. We observe that orthogonal subspaces with *higher* dimension achieve higher MCR^2 value, which is consistent with our theoretical analysis in Theorem A.1.6.

A.2.2 Implementation Details

Training Setting. We mainly use ResNet-18 [100] in our experiments, where we use 4 residual blocks with layer widths $[64, 128, 256, 512]$. The implementation of network architectures used in this work are mainly based on this github repo.[1] For data augmentation in the supervised setting, we apply the `RandomCrop` and `RandomHorizontalFlip`. For the supervised setting, we train the models for 500 epochs and use stage-wise learning rate decay every 200 epochs (decay by a factor of 10). For the supervised setting, we train the models for 100 epochs and use stage-wise learning rate decay at 20-th epoch and 40-th epoch (decay by a factor of 10).

Evaluation Details. For the supervised setting, we set the number of principal components for nearest subspace classifier $r_j = 30$. We also study the effect of r_j in Section A.2.3.2. For the CIFAR100 dataset, we consider 20 superclasses and set the cluster number as 20, which is the same setting as in [34, 276].

Datasets. We apply the default datasets in PyTorch, including CIFAR10, CIFAR100, and STL10.

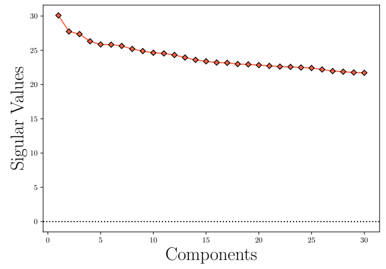
Augmentations $\mathcal{T}$ used for the self-supervised setting. We apply the same data augmentation for CIFAR10 dataset and CIFAR100 dataset and the pseudo-code is as follows.

```
import torchvision.transforms as transforms
TRANSFORM = transforms.Compose([
    transforms.RandomResizedCrop(32),
    transforms.RandomHorizontalFlip(),
    transforms.RandomApply([transforms.ColorJitter(0.4, 0.4, 0.4, 0.1)], p=0.8),
    transforms.RandomGrayscale(p=0.2),
    transforms.ToTensor()])
```

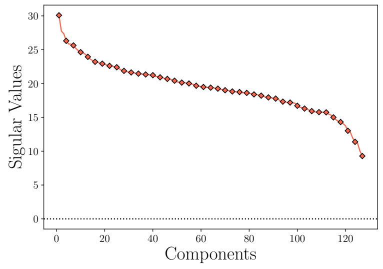
The augmentations we use for STL10 dataset and the pseudo-code is as follows.

```
import torchvision.transforms as transforms
TRANSFORM = transforms.Compose([
    transforms.RandomResizedCrop(96),
    transforms.RandomHorizontalFlip(),
    transforms.RandomApply([transforms.ColorJitter(0.8, 0.8, 0.8, 0.2)], p=0.8),
    transforms.RandomGrayscale(p=0.2),
    GaussianBlur(kernel_size=9),
    transforms.ToTensor()])
```

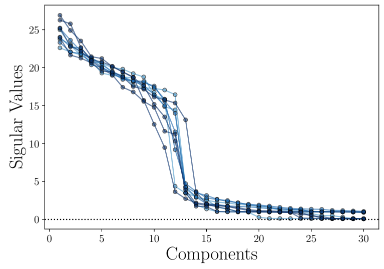
[1]`https://github.com/kuangliu/pytorch-cifar`



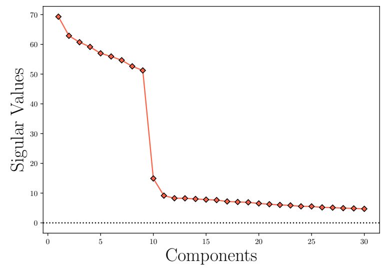
(a) PCA: MCR^2 training learned features for overall data.



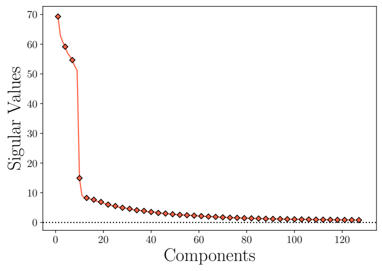
(b) PCA: MCR^2 training learned features for overall data.



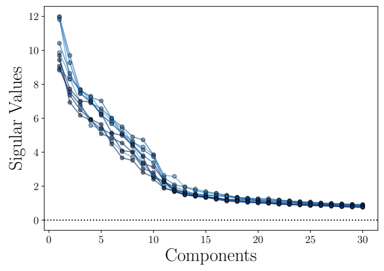
(c) PCA: MCR^2 training learned features for every class.



(d) PCA: cross-entropy training learned features for overall data.



(e) PCA: cross-entropy training learned features for overall data.



(f) PCA: cross-entropy training learned features for every class.

Figure A.1: Principal component analysis (PCA) of learned representations for the MCR^2 trained model (**first row**) and the cross-entropy trained model (**second row**). In (a) and (d), we visualize the first 30 components.

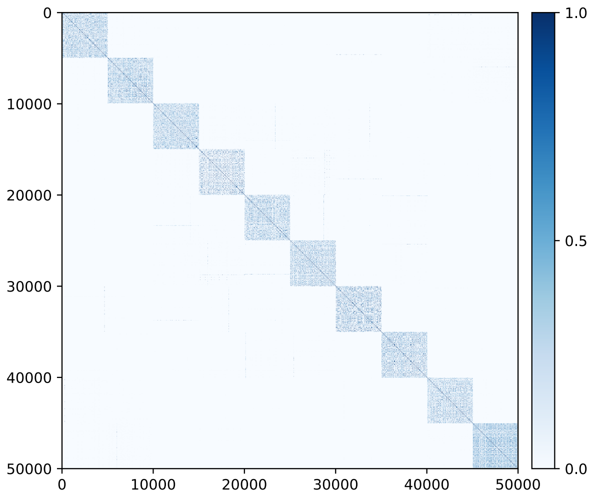
Cross-entropy training details. For CE models presented in Table 2.1, Figure A.1d-A.1f, and Figure A.2, we use the same network architecture, ResNet-18 [100], for cross-entropy training on CIFAR10, and set the output dimension as 10 for the last layer. We apply SGD, and set learning rate `lr=0.1`, momentum `momentum=0.9`, and weight decay `wd= 5e-4`. We set the total number of training epoch as 400, and use stage-wise learning rate decay every 150 epochs (decay by a factor of 10).

A.2.3 Additional Experimental Results

A.2.3.1 PCA Results of MCR^2 Training versus Cross-Entropy Training

For comparison, similar to Figure 2.3c, we calculate the principle components of representations learned by MCR^2 training and cross-entropy training. For cross-entropy training, we take the output of the second last layer as the learned representation. The results are summarized in Figure A.1. We also compare the cosine similarity between learned representations for both MCR^2 training and cross-entropy training, and the results are presented in Figure A.2.

As shown in Figure A.1, we observe that representations learned by MCR^2 are much more diverse, the dimension of learned features (each class) is around a dozen, and the dimension of the



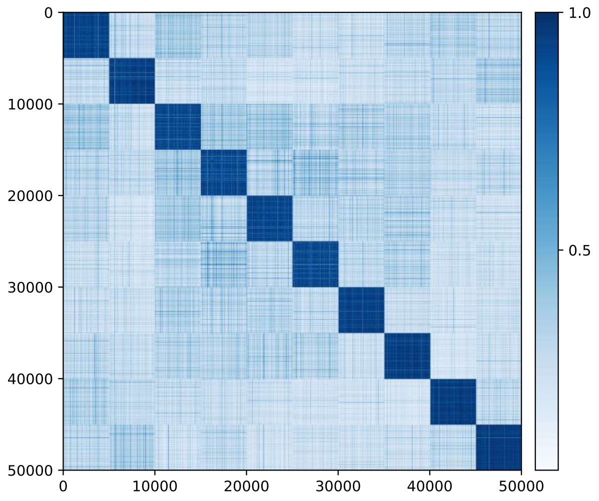
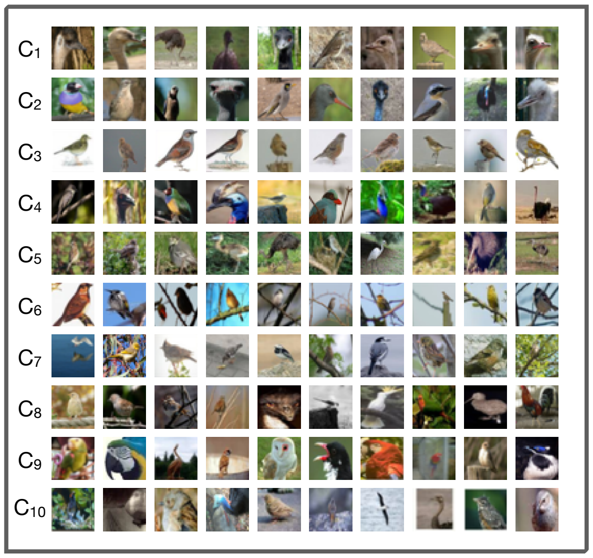
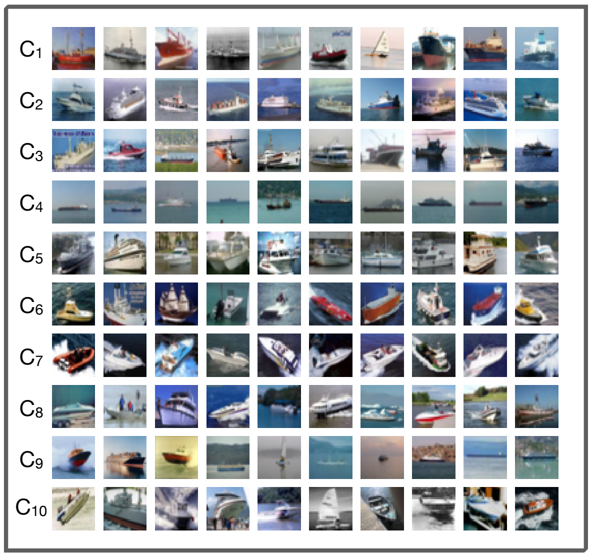


Figure A.2: Cosine similarity between learned features by using the MCR^2 objective (**left**) and CE loss (**right**).

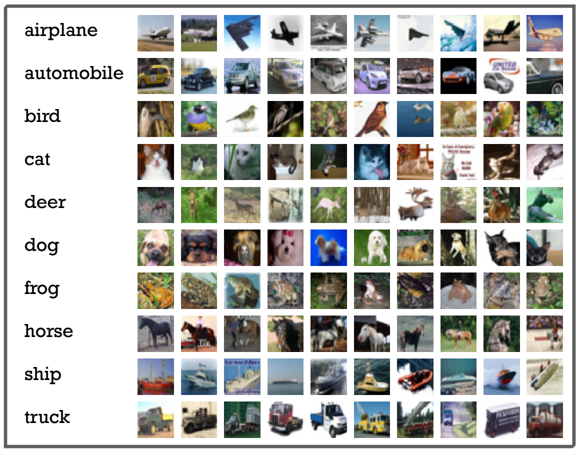


(a) Bird

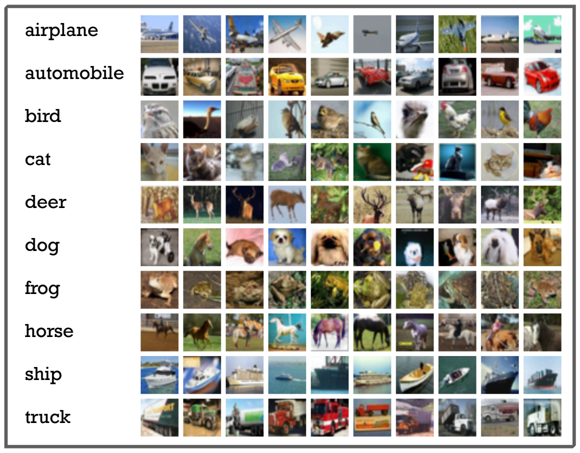


(b) Ship

Figure A.3: Visualization of principal components learned for class 2-'Bird' and class 8-'Ship'. For each class j, we first compute the top-10 singular vectors of the SVD of the learned features $\boldsymbol{Z}_j$. Then for the l-th singular vector of class j, $\boldsymbol{u}_j^l$, and for the feature of the i-th image of class j, $\boldsymbol{z}_j^i$, we calculate the absolute value of inner product, $|\langle \boldsymbol{z}_j^i, \boldsymbol{u}_j^l \rangle|$, then we select the top-10 images according to $|\langle \boldsymbol{z}_j^i, \boldsymbol{u}_j^l \rangle|$ for each singular vector. In the above two figures, each row corresponds to one singular vector (component C_l). The rows are sorted based on the magnitude of the associated singular values, from large to small.



(a) 10 representative images from each class based on top-10 principal components of the SVD of learned representations by MCR^2.



(b) Randomly selected 10 images from each class.

Figure A.4: Visualization of top-10 "principal" images for each class in the CIFAR10 dataset. **(a)** For each class-j, we first compute the top-10 singular vectors of the SVD of the learned features $\boldsymbol{Z}_j$. Then for the l-th singular vector of class j, $\boldsymbol{u}_j^l$, and for the feature of the i-th image of class j, $\boldsymbol{z}_j^i$, we calculate the absolute value of inner product, $|\langle \boldsymbol{z}_j^i, \boldsymbol{u}_j^l \rangle|$, then we select the largest one for each singular vector within class j. Each row corresponds to one class, and each image corresponds to one singular vector, ordered by the value of the associated singular value. **(b)** For each class, 10 images are randomly selected in the dataset. These images are the ones displayed in the CIFAR dataset website [144].

overall features is nearly 120, and the output dimension is 128. In contrast, the dimension of the overall features learned using entropy is slightly greater than 10, which is much smaller than that learned by MCR^2. From Figure A.2, for MCR^2 training, we find that the features of different class are almost orthogonal.

Visualize representative images selected from CIFAR10 dataset by using MCR^2. As mentioned in Section 2.1, obtaining the properties of desired representation in the proposed MCR^2 principle is equivalent to performing *nonlinear generalized principle components* on the given dataset. As shown in Figure A.1a-A.1c, MCR^2 can indeed learn such diverse and discriminative representations. In order to better interpret the representations learned by MCR^2, we select images according to their "principal" components (singular vectors using SVD) of the learned features. In Figure A.3, we visualize images selected from class-'Bird' and class-'Ship'. For each class, we first compute top-10 singular vectors of the SVD of the learned features and then for each of the top singular vectors, we display in each row the top-10 images whose corresponding features are closest to the singular vector. As shown in Figure A.3, we observe that images in the same row share many common characteristics such as shapes, textures, patterns, and styles, whereas images in different rows are significantly different from each other – suggesting our method captures all the different "modes" of the data even within the same class. Notice that top rows are associated with components with larger singular values, hence they are images that show up more frequently in the dataset.

In Figure A.4a, we visualize the 10 "principal" images selected from CIFAR10 for each of the 10 classes. That is, for each class, we display the 10 images whose corresponding features are most coherent with the top-10 singular vectors. We observe that the selected images are much more diverse and representative than those selected randomly from the dataset (displayed on the CIFAR official website), indicating such principal images can be used as a good "summary" of the dataset.

A.2.3.2 Experimental Results of MCR^2 in the Supervised Learning Setting.

Training details for mainline experiment. For the model presented in Figure 2.1 (**Right**) and Figure 2.3, we use ResNet-18 to parameterize $f(\cdot, \theta)$, and we set the output dimension $d = 128$, precision $\epsilon^2 = 0.5$, mini-batch size $m = 1,000$. We use SGD in Pytorch [196] as the optimizer, and set the learning rate `lr=0.01`, weight decay `wd=5e-4`, and `momentum=0.9`.

Experiments for studying the effect of hyperparameters and architectures. We present the experimental results of MCR^2 training in the supervised setting by using various training hyperparameters and different network architectures. The results are summarized in Table A.2. Besides the ResNet architecture, we also consider VGG architecture [223] and ResNext achitecture [280]. From Table A.2, we find that larger batch size m can lead to better performance. Also, models with higher output dimension d require larger training batch size m.

Effect of r_j on classification. Unless otherwise stated, we set the number of components $r_j = 30$ for nearest subspace classification. We study the effect of r_j when used for classification, and the results are summarized in Table A.3. We observe that the nearest subspace classification works for a wide range of r_j.

Effect of ϵ^2 on learning from corrupted labels. To further study the proposed MCR^2 on learning from corrupted labels, we use different precision parameters, $\epsilon^2 = 0.75, 1.0$, in addition to the one shown in Table 2.1. Except for the precision parameter ϵ^2, all the other parameters are the same as the mainline experiment (the first row in Table A.2). The first row ($\epsilon^2 = 0.5$) in Table A.4 is identical to the MCR^2 TRAINING in Table 2.2. Notice that with slightly different choices in ϵ^2, one might even see slightly improved performance over the ones reported in the main body.

A.2.4 Comparison with Related Work on Label Noise

We compare the proposed MCR^2 with OLE [153], Large Margin Deep Networks [67], and ITLM [220] in label noise robustness experiments on CIFAR10 dataset. In Table A.5, we compare MCR^2 with OLE [153] and Large Margin Deep Networks [67] on the corrupted label task using the same network, MCR^2 achieves significant better performance. We compare MCR^2 with ITLM [220] using the same network. MCR2 achieves better performance without any noise ratio dependent hyperparameters as required by [220].

A.2.4.1 Experimental Results of MCR^2 in the Self-supervised Learning Setting

Training details of MCR^2-CTRL. For three datasets (CIFAR10, CIFAR100, and STL10), we use ResNet-18 as in the supervised setting, and we set the output dimension $d = 128$, precision $\epsilon^2 = 0.5$, mini-batch size $k = 20$, number of augmentations $n = 50$, $\gamma_1 = \gamma_2 = 20$. We observe that MCR^2-CTRL can achieve better clustering performance by using smaller γ_2, i.e., $\gamma_2 = 15$, on CIFAR10 and CIFAR100 datasets. We use SGD in Pytorch [196] as the optimizer, and set the learning rate `lr=0.1`, weight decay `wd=5e-4`, and `momentum=0.9`.

Training dynamic comparison between MCR^2 and MCR^2-CTRL . In the self-supervised setting, we compare the training process for MCR^2 and MCR^2-CTRL in terms of $R, \widetilde{R}, R^c$, and ΔR. For MCR^2 training, the features first expand (for both R and R^c) then compress (for). For MCR^2-CTRL, both $\widetilde{R}$ and R^c first compress then $\widetilde{R}$ expands quickly and R^c remains small, as we have seen in Figure 2.5 in the main body.

Clustering results comparison. We compare the clustering performance between MCR^2 and MCR^2-CTRL in terms of NMI, ACC, and ARI. The clustering results are summarized in Table A.6. We find that MCR^2-CTRL can achieve better performance for clustering.

A.2.4.2 Clustering Metrics and More Results

We first introduce the definitions of normalized mutual information (NMI) [236], clustering accuracy (ACC), and adjusted rand index (ARI) [116].

Normalized mutual information (NMI). Suppose Y is the ground truth partition and C is the prediction partition. The NMI metric is defined as

$$\mathrm{NMI}(Y, C) = \frac{\sum_{i=1}^{k}\sum_{j=1}^{s}|Y_i \cap C_j| \log\left(\frac{m|Y_i \cap C_j|}{|Y_i||C_j|}\right)}{\sqrt{\left(\sum_{i=1}^{k}|Y_i|\log\left(\frac{|Y_i|}{m}\right)\right)\left(\sum_{j=1}^{s}|C_j|\log\left(\frac{|C_j|}{m}\right)\right)}},$$

where Y_i is the i-th cluster in Y and C_j is the j-th cluster in C, and m is the total number of samples.

Clustering accuracy (ACC). Given m samples, $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^m$. For the i-th sample $\boldsymbol{x}_i$, let $\boldsymbol{y}_i$ be its ground truth label, and let $\boldsymbol{c}_i$ be its cluster label. The ACC metric is defined as

$$\mathrm{ACC}(\boldsymbol{Y}, \boldsymbol{C}) = \max_{\sigma \in S} \frac{\sum_{i=1}^{m} \mathbf{1}\{\boldsymbol{y}_i = \sigma(\boldsymbol{c}_i)\}}{m},$$

where S is the set includes all the one-to-one mappings from cluster to label, and $\boldsymbol{Y} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m]$, $\boldsymbol{C} = [\boldsymbol{c}_1, \ldots, \boldsymbol{c}_m]$.

Adjusted rand index (ARI). Suppose there are m samples, and let Y and C be two clustering of these samples, where $Y = \{Y_1, \ldots, Y_r\}$ and $C = \{C_1, \ldots, C_s\}$. Let m_{ij} denote the number of the intersection between Y_i and C_j, i.e., $m_{ij} = |Y_i \cap C_j|$. The ARI metric is defined as

$$\mathrm{ARI} = \frac{\sum_{ij}\binom{m_{ij}}{2} - \left(\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right)/\binom{m}{2}}{\frac{1}{2}\left(\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right) - \left(\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}\right)/\binom{m}{2}},$$

where $a_i = \sum_j m_{ij}$ and $b_j = \sum_i m_{ij}$.

Comparison with [125, 113]. We compare MCR^2 with IIC [125] and IMSAT [113] in Table A.7. We find that MCR^2 outperforms IIC [125] and IMSAT [113] on both CIFAR10 and CIFAR100 by a large margin. For STL10, [113] applied pretrained ImageNet models and [125] used more data for training.

More experiments on the effect of hyperparameters of MCR^2-CTRL. We provide more experimental results of MCR^2-CTRL training in the self-supervised setting by varying training hyperparameters on the STL10 dataset. The results are summarized in Table A.8. Notice that the choice of hyperparameters only has small effect on the performance with the MCR^2-CTRL objective. We may hypothesize that, in order to further improve the performance, one has to seek other, potentially better, control of optimization dynamics or strategies. We leave those for future investigation.

Table A.1: **MCR^2 objective on simulated data.** We evaluate the proposed MCR^2 objective defined in (2.8), including R, R^c, and ΔR, on simulated data. The output dimension d is set as 512, 256, and 128. We set the batch size as $m = 1000$ and random assign the label of each sample from 0 to 9, i.e., 10 classes. We generate two types of data: 1) (RANDOM GAUSSIAN) For comparison with data without structures, for each class we generate random vectors sampled from Gaussian distribution (the dimension is set as the output dimension d) and normalize each vector to be on the unit sphere. 2) (SUBSPACE) For each class, we generate vectors sampled from its corresponding subspace with dimension d_j and normalize each vector to be on the unit sphere. We consider the subspaces from different classes are orthogonal/nonorthogonal to each other.

	R	R^c	ΔR	Orthogonal?	Output Dimension
Random Gaussian	552.70	193.29	360.41	✓	512
Subspace ($d_j = 50$)	545.63	108.46	**437.17**	✓	512
Subspace ($d_j = 40$)	487.07	92.71	394.36	✓	512
Subspace ($d_j = 30$)	413.08	74.84	338.24	✓	512
Subspace ($d_j = 20$)	318.52	54.48	264.04	✓	512
Subspace ($d_j = 10$)	195.46	30.97	164.49	✓	512
Subspace ($d_j = 1$)	31.18	4.27	26.91	✓	512
Random Gaussian	292.71	154.13	138.57	✓	256
Subspace ($d_j = 25$)	288.65	56.34	**232.31**	✓	256
Subspace ($d_j = 20$)	253.51	47.58	205.92	✓	256
Subspace ($d_j = 15$)	211.97	38.04	173.93	✓	256
Subspace ($d_j = 10$)	161.87	27.52	134.35	✓	256
Subspace ($d_j = 5$)	98.35	15.55	82.79	✓	256
Subspace ($d_j = 1$)	27.73	3.92	23.80	✓	256
Random Gaussian	150.05	110.85	39.19	✓	128
Subspace ($d_j = 12$)	144.36	27.72	**116.63**	✓	128
Subspace ($d_j = 10$)	129.12	24.06	105.05	✓	128
Subspace ($d_j = 8$)	112.01	20.18	91.83	✓	128
Subspace ($d_j = 6$)	92.55	16.04	76.51	✓	128
Subspace ($d_j = 4$)	69.57	11.51	58.06	✓	128
Subspace ($d_j = 2$)	41.68	6.45	35.23	✓	128
Subspace ($d_j = 1$)	24.28	3.57	20.70	✓	128
Subspace ($d_j = 50$)	145.60	75.31	70.29	✗	128
Subspace ($d_j = 40$)	142.69	65.68	77.01	✗	128
Subspace ($d_j = 30$)	135.42	54.27	81.15	✗	128
Subspace ($d_j = 20$)	120.98	40.71	80.27	✗	128
Subspace ($d_j = 15$)	111.10	32.89	78.21	✗	128
Subspace ($d_j = 12$)	101.94	27.73	74.21	✗	128

Table A.2: Experiments of MCR^2 in the supervised setting on the CIFAR10 dataset.

Arch	Dim d	Precision ϵ^2	BatchSize m	`lr`	ACC	Comment
ResNet-18	128	0.5	1,000	0.01	92.20%	Mainline, Fig 2.3
ResNext-29	128	0.5	1,000	0.01	92.55%	Different
VGG-11	128	0.5	1,000	0.01	90.76%	Architecture
ResNet-18	512	0.5	1,000	0.01	88.60%	Effect of
ResNet-18	256	0.5	1,000	0.01	92.10%	Output
ResNet-18	64	0.5	1,000	0.01	92.21%	Dimension
ResNet-18	128	1.0	1,000	0.01	93.06%	Effect of precision
ResNet-18	128	0.4	1,000	0.01	91.93%	
ResNet-18	128	0.2	1,000	0.01	90.06%	
ResNet-18	128	0.5	500	0.01	82.33%	Effect of Batch Size
ResNet-18	128	0.5	2,000	0.01	93.02%	
ResNet-18	128	0.5	4,000	0.01	92.59%	
ResNet-18	512	0.5	2,000	0.01	92.47%	
ResNet-18	512	0.5	4,000	0.01	92.17%	
ResNet-18	128	0.5	1,000	0.05	86.02%	Effect of `lr`
ResNet-18	128	0.5	1,000	0.005	92.39%	
ResNet-18	128	0.5	1,000	0.001	92.23%	

Table A.3: Effect of number of components r_j for nearest subspace classification in the supervised setting.

Number of components	$r_j = 10$	$r_j = 20$	$r_j = 30$	$r_j = 40$	$r_j = 50$
Mainline (Label Noise Ratio=0.0)	92.68%	92.53%	92.20%	92.32%	92.17%
Label Noise Ratio=0.1	91.71%	91.73%	91.16%	91.83%	91.78%
Label Noise Ratio=0.2	90.68%	90.61%	89.70%	90.62%	90.54%
Label Noise Ratio=0.3	88.24%	87.97%	88.18%	88.15%	88.10%
Label Noise Ratio=0.4	86.49%	86.67%	86.66%	86.71%	86.44%
Label Noise Ratio=0.5	83.90%	84.18%	84.30%	84.18%	83.76%

Table A.4: Effect of Precision ϵ^2 on classification results with features learned with labels corrupted at different levels by using MCR^2 training.

Precision	Ratio=0.1	Ratio=0.2	Ratio=0.3	Ratio=0.4	Ratio=0.5
$\epsilon^2 = 0.5$	91.16%	89.70%	88.18%	86.66%	84.30%
$\epsilon^2 = 0.75$	**92.37%**	90.82%	**89.91%**	**87.67%**	83.69%
$\epsilon^2 = 1.0$	91.93%	**91.11%**	89.60%	87.09%	**84.53%**

Table A.5: Comparison with related work on learning from noisy labels.

ResNet18	Ratio=0.1	Ratio=0.2	Ratio=0.3	Ratio=0.4	Ratio=0.5
OLE [153]	91.04%	86.01%	80.69%	71.79%	61.06%
LargeMargin [67]	90.10%	87.42%	83.77%	78.51%	72.48%
MCR^2	**91.16%**	**89.70%**	**88.18%**	**86.66%**	**84.30%**
WRN16	Ratio=0.1	Ratio=0.3	Ratio=0.5	Ratio=0.7	
ITLM [220]	90.33%	88.23%	82.51%	64.74%	
MCR^2	**91.55%**	**88.81%**	**84.25%**	**67.09%**	

Table A.6: Clustering comparison between MCR^2 and MCR^2-CTRL on CIFAR10 dataset.

	NMI	ACC	ARI
MCR^2	0.544	0.570	0.399
MCR^2-Ctrl	0.630	0.684	0.508

Table A.7: Compare with [125, 113] on clustering.

Dataset	Metric	IIC	IMSAT	MCR^2-Ctrl
	NMI	-	-	**0.630**
CIFAR10	ACC	0.617	0.456	**0.684**
	ARI	-	-	**0.508**
	NMI	-	-	**0.387**
CIFAR100	ACC	0.257	0.275	**0.375**
	ARI	-	-	**0.178**

Table A.8: Experiments of MCR^2-CTRL in the self-supervised setting on STL10 dataset.

Arch	Precision ϵ^2	Learning Rate `lr`	NMI	ACC	ARI
ResNet-18	0.5	0.1	0.446	0.491	0.290
ResNet-18	0.75	0.1	0.450	0.484	0.288
ResNet-18	0.25	0.1	0.447	0.489	0.293
ResNet-18	0.5	0.2	0.477	0.473	0.295
ResNet-18	0.5	0.05	0.444	0.496	0.293
ResNet-18	0.25	0.05	0.454	0.489	0.294

Appendix B

Appendix: Interpretable White-Box Transformers via Sparse Rate Reduction

B.1 Technical Details for CRATE

B.1.1 Companion to Chapter 3.2.2

We first wish to re-iterate the core contributions of our approach in Chapter 3.2.2 at a slightly more technical level. Connections between denoising and score matching are well-understood [135], and computing the optimal denoising function (i.e., the conditional expectation) against a mixture-of-Gaussians model is a rather simple computation giving existing tools such as Tweedie's formula [66]. These are not our main contributions. Instead, the main contributions of Chapter 3.2.2 are two-fold:

- First, we demonstrate a mechanism to learn representations via denoising within a idealized mixture of Gaussian data model for a single token (i.e., with sequence length $N = 1$).
- Second, we illustrate the similarities between a such-derived representation learning scheme and existing self-attention layers within the transformer (with sequence length 1), thus demonstrating an interpretation of the self-attention layer as a generalized mechanism to denoise against a mixture-of-Gaussian-marginal model for a set of tokens.

Now we provide more details alluded to in Chapter 3.2.2, which mostly form the technical aspects of the first listed contribution. To simplify the proofs, we use the following notation correspondences: $\boldsymbol{x} \mapsto \boldsymbol{z}^{\ell}$, $\boldsymbol{z} \mapsto \boldsymbol{z}^{\ell+1}$, and $\sigma \mapsto \sigma^{\ell}$. Refer to Appendix A.1 in [289] for detailed proof.

Proposition B.1.1. *Let $\boldsymbol{u}_1, \dots, \boldsymbol{u}_K \in \mathbb{R}^d$ be independent and have distribution $\boldsymbol{u}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_k)$ for $\boldsymbol{\Sigma}_k \succeq \mathbf{0}$, and let $\boldsymbol{z}$ take value $\boldsymbol{u}_k$ with probability $\pi_k > 0$. Let $\boldsymbol{w} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$ be independent of $\boldsymbol{z}$. Let $\boldsymbol{x} \doteq \boldsymbol{z} + \sigma\boldsymbol{w}$. Let $\boldsymbol{x} \mapsto q(\boldsymbol{x})$ be the density of $\boldsymbol{x}$. We define*

$$\boldsymbol{M}_k \doteq (\boldsymbol{\Sigma}_k + \sigma^2 \boldsymbol{I}_d)^{-1/2} \tag{B.1}$$

and assume that $\pi_i \det(\boldsymbol{M}_i) = \pi_j \det(\boldsymbol{M}_j)$ for all $1 \leq i \leq j \leq K$. Then we have

$$\nabla_{\boldsymbol{x}} \log q(\boldsymbol{x}) \tag{B.2}$$

$$= -\begin{bmatrix}\boldsymbol{M}_1, \cdots, \boldsymbol{M}_K\end{bmatrix} \left[\operatorname{diag}\left(\operatorname{softmax}\left(-\frac{1}{2}\begin{bmatrix}\|\boldsymbol{M}_1^*\boldsymbol{x}\|_2^2 \\ \vdots \\ \|\boldsymbol{M}_K^*\boldsymbol{x}\|_2^2\end{bmatrix}\right)\right) \otimes \boldsymbol{I}_d\right] \begin{bmatrix}\boldsymbol{M}_1^*\boldsymbol{x} \\ \vdots \\ \boldsymbol{M}_K^*\boldsymbol{x}\end{bmatrix}, \tag{B.3}$$

where $\otimes$ denotes the Kronecker product, i.e., the block matrix defined by

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} A_{11}\boldsymbol{B} & \cdots & A_{1n}\boldsymbol{B} \\ \vdots & \ddots & \vdots \\ A_{m1}\boldsymbol{B} & \cdots & A_{mn}\boldsymbol{B} \end{bmatrix} \tag{B.4}$$

Now we provide a final justification for the result cited in Chapter 3.2.2. Refer to Appendix A.1 in [289] for detailed proof.

Proposition B.1.2. *In the setting of Chapter B.1.1, diagonalize $\boldsymbol{\Sigma}_k = \boldsymbol{U}_k\boldsymbol{\Lambda}_k\boldsymbol{U}_k^*$ where $\boldsymbol{U}_k \in \mathbb{R}^{d\times p}$ is orthogonal and $\boldsymbol{\Lambda}_k \succ \boldsymbol{0} \in \mathbb{R}^{p\times p}$ is diagonal.*[1] *Then we have the approximation*

$$\mathbb{E}[\boldsymbol{z} \mid \boldsymbol{x}] \approx \begin{bmatrix}\boldsymbol{U}_1, \ldots, \boldsymbol{U}_K\end{bmatrix} \left[\operatorname{diag}\left(\operatorname{softmax}\left(\frac{1}{2\sigma^2}\begin{bmatrix}\|\boldsymbol{U}_1^*\boldsymbol{x}\|_2^2 \\ \vdots \\ \|\boldsymbol{U}_K^*\boldsymbol{x}\|_2^2\end{bmatrix}\right)\right) \otimes \boldsymbol{I}_p\right] \begin{bmatrix}\boldsymbol{U}_1^*\boldsymbol{x} \\ \vdots \\ \boldsymbol{U}_K^*\boldsymbol{x}\end{bmatrix}. \tag{B.5}$$

Remark 1. *Although Chapter B.1.2 is stated as an approximation rather than as a proposition, we believe it should be possible without too much extra work to convert it into a statement of asymptotic equivalence as $\sigma \to 0$ (in particular, holding for σ below the smallest (nonzero) eigenvalue of any $\boldsymbol{\Sigma}_k$. Most approximations taken in the derivation of Chapter B.1.2 can immediately be turned into asymptotic claims; the only slightly delicate point is treating the softmax, which can be accomplished using standard "high temperature" convergence behavior of the softmax function (in particular, as $\sigma \to 0$ in our expressions, the softmax concentrates on the "best head").*

B.1.2 Companion to Chapter 3.2.3

We again wish to re-iterate the core contribution of our approach in Chapter 3.2.3. The application of a compression perspective to representation learning has been discussed before, for example in the line of maximal coding rate reduction works [290]. In Chapter 3.2.3, we provide the following contributions and developments to this perspective:

- We propose a generalized coding rate function $R^c(\cdot; \boldsymbol{U}_{[K]})$ which measures the coding rate with respect to a set of subspaces $\boldsymbol{U}_{[K]}$ as opposed to a set of classes (as in [290, 33]), making the underlying formulation unsupervised.

[1]This assumption can be easily relaxed to $\boldsymbol{\Lambda}_k \succeq \boldsymbol{0}$ for all k, but requires some more notation to handle, and the form of the solution does not change. Thus we handle the case where all matrices are full rank for simplicity.

- We then show how if we adopt the framework of alternating minimization of the sparse rate reduction objective, then unrolling the first alternating step — gradient descent on this coding rate objective — nearly exactly recovers the common multi-head attention mechanism found in transformer networks (except that the query/key/value operators are all the same operation $\boldsymbol{U}_k^*$ now, which we interpret as projection onto a single subspace).

In the process of the second contribution, and in the following proofs, we make some simple approximations and technical assumptions. The validity of these assumptions may be explored, and the approximations refined, altogether providing a more complex (and possibly more performant) resulting self-attention like operator. For the sake of technical clarity and simplicity in this work, we make perhaps the *simplest possible choices*. As a result, we *do not* claim that our network is optimally designed, but rather that the principles we develop in this work (compression, denoising, sparsification, unrolled optimization) can provide the backbone for far superior and more interpretable network architectures in the future on sundry tasks. As it is, with our straightforward, simple, and interpretable design, we still obtain meaningful conceptual results and very solid empirical performance.

We now give the derivation of the approximation alluded to in Chapter 3.2.3.

Proposition B.1.3. *Let $\boldsymbol{Z} \in \mathbb{R}^{d\times N}$ have unit-norm columns, and $\boldsymbol{U}_{[K]} = (\boldsymbol{U}_1, \ldots, \boldsymbol{U}_K)$ such that each $\boldsymbol{U}_k \in \mathbb{R}^{d\times p}$ is an orthogonal matrix, the $(\boldsymbol{U}_k)_{k=1}^K$ are incoherent, and the columns of $\boldsymbol{Z}$ approximately lie on $\bigcup_{k=1}^K \mathrm{Span}(\boldsymbol{U}_k)$. Let $\gamma = \frac{p}{N\varepsilon^2}$. Let $\kappa > 0$. Then*

$$\boldsymbol{Z} - \kappa\nabla_{\boldsymbol{Z}} R^c(\boldsymbol{Z} \mid \boldsymbol{U}_{[K]}) \approx (1-\kappa\gamma)\boldsymbol{Z} + \kappa\gamma\, \texttt{MSSA}(\boldsymbol{Z}|\boldsymbol{U}_{[K]}), \tag{B.6}$$

where as in Chapter 3.2.3 we have

$$\texttt{SSA}(\boldsymbol{Z}|\boldsymbol{U}_k) = (\boldsymbol{U}_k^*\boldsymbol{Z})\mathrm{softmax}\left((\boldsymbol{U}_k^*\boldsymbol{Z})^*(\boldsymbol{U}_k^*\boldsymbol{Z})\right), \tag{B.7}$$

$$\texttt{MSSA}(\boldsymbol{Z}|\boldsymbol{U}_{[K]}) = \gamma\begin{bmatrix}\boldsymbol{U}_1, \ldots, \boldsymbol{U}_K\end{bmatrix}\begin{bmatrix}\texttt{SSA}(\boldsymbol{Z}|\boldsymbol{U}_1)\\ \vdots \\ \texttt{SSA}(\boldsymbol{Z}|\boldsymbol{U}_K)\end{bmatrix}, \tag{B.8}$$

where $\mathrm{softmax}(\cdot)$ *is the softmax operator (applied to each column of an input matrix), i.e.,*

$$\mathrm{softmax}(\boldsymbol{v}) = \frac{1}{\sum_{i=1}^n e^{v_i}}\begin{bmatrix}e^{v_1}\\ \vdots \\ e^{v_n}\end{bmatrix}, \tag{B.9}$$

$$\mathrm{softmax}\left(\begin{bmatrix}\boldsymbol{v}_1, \ldots, \boldsymbol{v}_K\end{bmatrix}\right) = \begin{bmatrix}\mathrm{softmax}(\boldsymbol{v}_1), \ldots, \mathrm{softmax}(\boldsymbol{v}_K)\end{bmatrix}. \tag{B.10}$$

Proof. According to Chapter 3.9, the gradient $\nabla_{\boldsymbol{Z}} R^c(\boldsymbol{Z}; \boldsymbol{U}_{[K]})$ is

$$\nabla_{\boldsymbol{Z}} R^c(\boldsymbol{Z}; \boldsymbol{U}_{[K]}) = \gamma\sum_{k=1}^K \boldsymbol{U}_k\boldsymbol{U}_k^*\boldsymbol{Z}\left(\boldsymbol{I} + \gamma(\boldsymbol{U}_k^*\boldsymbol{Z})^*(\boldsymbol{U}_k^*\boldsymbol{Z})\right)^{-1}. \tag{B.11}$$

Notice that according to [33], the gradient is precisely the residual of a ridge regression for each (projected) token $\boldsymbol{U}_k^* \boldsymbol{z}_i$ using other projected tokens $\boldsymbol{U}_k^* \boldsymbol{z}_j$ as the regressors, hence being the residual of an auto-regression. However, as we have seen in the work of ReduNet [33], computing the inverse

$$(\boldsymbol{I} + \gamma(\boldsymbol{U}_k^* \boldsymbol{Z})^*(\boldsymbol{U}_k^* \boldsymbol{Z}))^{-1}$$

can be expensive. Hence for computational efficiency, we may approximate it with the first order term of its von Neumann expansion:

$$\begin{aligned}
\nabla_{\boldsymbol{Z}} R^c(\boldsymbol{Z}; \boldsymbol{U}_{[K]}) &= \gamma \sum_{k=1}^{K} \boldsymbol{U}_k \boldsymbol{U}_k^* \boldsymbol{Z} \Big(\boldsymbol{I} + \gamma(\boldsymbol{U}_k^* \boldsymbol{Z})^*(\boldsymbol{U}_k^* \boldsymbol{Z})\Big)^{-1} &\text{(B.12)}\\
&\approx \gamma \sum_{k=1}^{K} \boldsymbol{U}_k \boldsymbol{U}_k^* \boldsymbol{Z} \Big(\boldsymbol{I} - \gamma(\boldsymbol{U}_k^* \boldsymbol{Z})^*(\boldsymbol{U}_k^* \boldsymbol{Z})\Big) &\text{(B.13)}\\
&= \gamma \sum_{k=1}^{K} \boldsymbol{U}_k \Big(\boldsymbol{U}_k^* \boldsymbol{Z} - \gamma \boldsymbol{U}_k^* \boldsymbol{Z} [(\boldsymbol{U}_k^* \boldsymbol{Z})^*(\boldsymbol{U}_k^* \boldsymbol{Z})]\Big) &\text{(B.14)}
\end{aligned}$$

Notice that the term $(\boldsymbol{U}_k^* \boldsymbol{Z})^*(\boldsymbol{U}_k^* \boldsymbol{Z})$ is the auto-correlation among the projected tokens. As the tokens $\boldsymbol{Z}$ may be from different subspaces, we would prefer to use only tokens that belong to the *same* subspace to regress and compress themselves. Hence we may convert the above correlation term into a subspace-membership indicator with a softmax operation, whence (B.14) becomes

$$\begin{aligned}
& \nabla_{\boldsymbol{Z}} R^c(\boldsymbol{Z}; \boldsymbol{U}_{[K]}) &\text{(B.15)}\\
\approx \quad & \gamma \sum_{k=1}^{K} \boldsymbol{U}_k \Big(\boldsymbol{U}_k^* \boldsymbol{Z} - \gamma \boldsymbol{U}_k^* \boldsymbol{Z} [(\boldsymbol{U}_k^* \boldsymbol{Z})^*(\boldsymbol{U}_k^* \boldsymbol{Z})]\Big) &\text{(B.16)}\\
\approx \quad & \gamma \sum_{k=1}^{K} \boldsymbol{U}_k \boldsymbol{U}_k^* \boldsymbol{Z} - \gamma^2 \sum_{k=1}^{K} \boldsymbol{U}_k \Big(\boldsymbol{U}_k^* \boldsymbol{Z} \operatorname{softmax}\left((\boldsymbol{U}_k^* \boldsymbol{Z})^*(\boldsymbol{U}_k^* \boldsymbol{Z})\right)\Big) &\text{(B.17)}
\end{aligned}$$

Then, we can rewrite the above approximation to the gradient of R^c as:

$$
\begin{aligned}
&\nabla_{\boldsymbol{Z}} R^c(\boldsymbol{Z}; \boldsymbol{U}_{[K]}) && \text{(B.18)}\\
\approx&\gamma \sum_{k=1}^{K} \boldsymbol{U}_k \boldsymbol{U}_k^* \boldsymbol{Z} - \gamma^2 \sum_{k=1}^{K} \boldsymbol{U}_k \left(\boldsymbol{U}_k^* \boldsymbol{Z} \operatorname{softmax}\left((\boldsymbol{U}_k^* \boldsymbol{Z})^* (\boldsymbol{U}_k^* \boldsymbol{Z})\right)\right) && \text{(B.19)}\\
=&\gamma \sum_{k=1}^{K} \boldsymbol{U}_k \boldsymbol{U}_k^* \boldsymbol{Z} - \gamma^2 \sum_{k=1}^{K} \boldsymbol{U}_k \, \texttt{SSA}(\boldsymbol{Z} \mid \boldsymbol{U}_k) && \text{(B.20)}\\
=&\underbrace{\left(\gamma \sum_{k=1}^{K} \boldsymbol{U}_k \boldsymbol{U}_k^*\right) \boldsymbol{Z}}_{\approx \gamma \boldsymbol{Z}} - \gamma^2 \left[\boldsymbol{U}_1, \cdots, \boldsymbol{U}_K\right] \begin{bmatrix} \texttt{SSA}(\boldsymbol{Z} \mid \boldsymbol{U}_1) \\ \vdots \\ \texttt{SSA}(\boldsymbol{Z} \mid \boldsymbol{U}_K) \end{bmatrix} && \text{(B.21)}\\
\approx&\gamma \boldsymbol{Z} - \gamma^2 \left[\boldsymbol{U}_1, \cdots, \boldsymbol{U}_K\right] \begin{bmatrix} \texttt{SSA}(\boldsymbol{Z} \mid \boldsymbol{U}_1) \\ \vdots \\ \texttt{SSA}(\boldsymbol{Z} \mid \boldsymbol{U}_K) \end{bmatrix}. && \text{(B.22)}
\end{aligned}
$$

Thus the gradient descent step with learning rate $\kappa > 0$ gives

$$
\boldsymbol{Z} - \kappa \nabla_{\boldsymbol{Z}} R^c(\boldsymbol{Z} \mid \boldsymbol{U}_{[K]}) \approx (1 - \kappa\gamma)\boldsymbol{Z} + \kappa\gamma^2 \left[\boldsymbol{U}_1, \ldots, \boldsymbol{U}_K\right] \begin{bmatrix} \texttt{SSA}(\boldsymbol{Z}|\boldsymbol{U}_1) \\ \vdots \\ \texttt{SSA}(\boldsymbol{Z}|\boldsymbol{U}_K) \end{bmatrix}. \tag{B.23}
$$

□

B.1.3 Implementation details

In this subsection, we provide more details for implementing CRATE on vision tasks.

B.1.3.1 Architecture of CRATE

Architectural modifications. Compared to the conceptual architecture proposed in Chapter 3.2.5 and Chapter 3.3, we make the following change for the sake of implementation simplicity:

- In the compression step, replace the term $\frac{p}{N\varepsilon^2}\left[\boldsymbol{U}_1, \ldots, \boldsymbol{U}_K\right]$ in the MSSA operator with another trainable parameter $\boldsymbol{W} \in \mathbb{R}^{d \times pK}$. Thus the MSSA block becomes

$$
\texttt{MSSA}(\boldsymbol{Z} \mid \boldsymbol{U}_{[K]}, \boldsymbol{W}) \doteq \boldsymbol{W} \begin{bmatrix} \texttt{SSA}(\boldsymbol{Z} \mid \boldsymbol{U}_1) \\ \vdots \\ \texttt{SSA}(\boldsymbol{Z} \mid \boldsymbol{U}_K) \end{bmatrix}. \tag{B.24}
$$

B.1.4 Additional Experiments

Table B.1: Top 1 accuracy of CRATE on various datasets with different model scales when pre-trained on ImageNet-21K and fine-tuned on the downstream datasets.

	CRATE-T	CRATE-S	CRATE-B	ViT-T	ViT-B
# parameters	5.74M	14.12M	38.83M	10.36M	102.61M
ImageNet-1K	62.7	74.2	79.5	71.8	85.8
CIFAR10	94.1	97.2	98.1	97.2	98.9
CIFAR100	76.7	84.1	87.9	84.4	90.1
Oxford Flowers-102	82.2	92.2	96.7	92.1	99.5
Oxford-IIIT-Pets	77.0	86.4	90.7	86.2	91.8

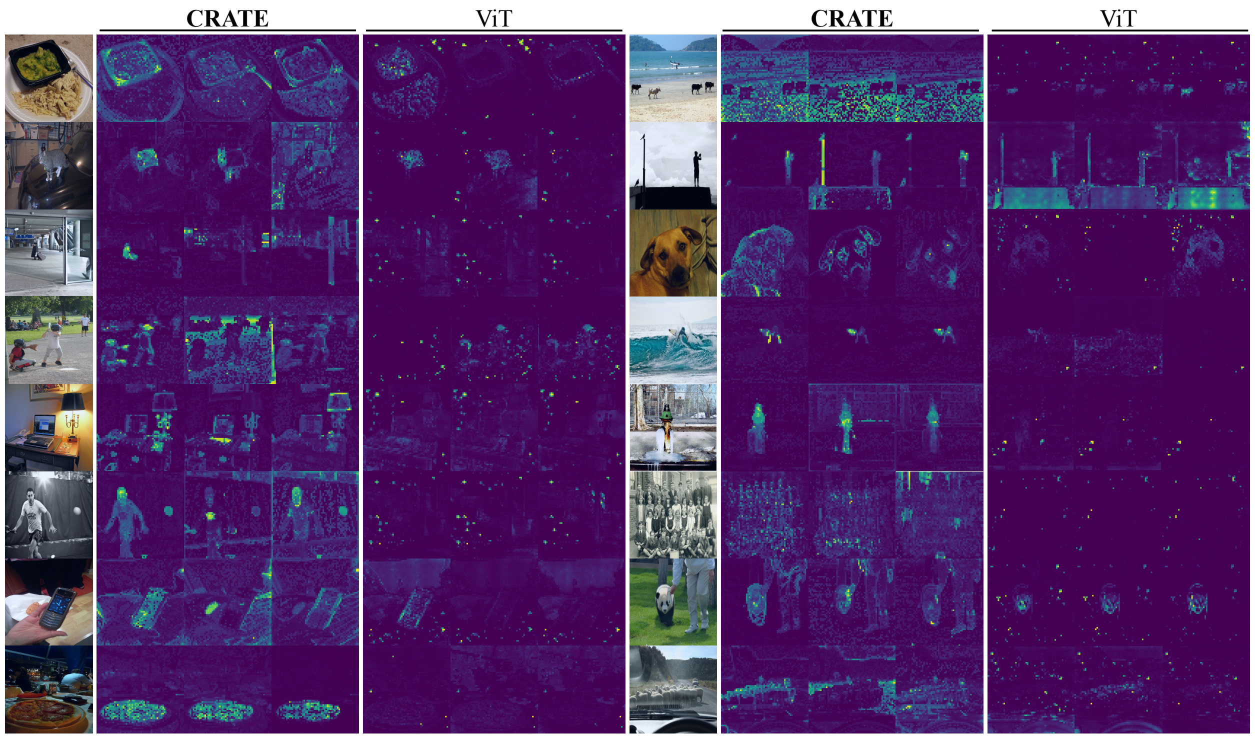


Figure B.1: **More attention maps of supervised CRATE and ViT** on images from COCO val2017. We select the second-to-last layer attention maps to visualize for CRATE and the last layer for ViT.

Appendix C

Appendix: Reliable Federated Learning via NTK Representations

C.1 Additional Details About TCT

C.1.1 An Efficient Implementation of SCAFFOLD

Algorithm 1 Efficient implementation of SCAFFOLD

```
Input: losses $\{L_k\}$, $k \in [K]$. Number of local steps $M$, server model $\theta^0$, learning rate $\eta$.
Initialization:    client corrections $\{h_k^{-1} = \mathbf{0}\}$, local models$\{\widehat{\theta}_i^0\} = \theta^0,\ k \in [K]$
for round $t = 0, 1, \ldots, T$ do
    for clients $k = 1, \ldots, K$ in parallel do
        Receive $\theta^t$ from server. Update correction
            $$h_k^t = h_k^{t-1} + \tfrac{1}{M\eta}(\theta^t - \widehat{\theta}_k^t). \quad (C.1)$$
        Initialize client local model $\widehat{\theta}_i^{t,0} = \theta^t$.
        for $m = 1, \ldots, M$ do
            Update with a stochastic gradient sampled from local client data
            $$\widehat{\theta}_k^{t,m+1} = \widehat{\theta}_k^{t,m} - \eta\left(\nabla L_k(\theta_i^{t,m}; \xi_k^{t,m}) - h_k^t\right). \quad (C.2)$$
        end for
        Set $\widehat{\theta}_k^{t+1} = \widehat{\theta}_k^{t,M+1}$. Communicate $\widehat{\theta}_k^{t+1}$ to server.
    end for
    Aggregate $\theta^{t+1} = \frac{1}{K}\sum_{k=1}^{K} \widehat{\theta}_k^{t+1}$ .[1]
end for
```

We describe a more communication efficient implementation of SCAFFOLD which is equivalent to Option II of SCAFFOLD from [134]. Our implementation only requires a single model to be communicated between the client and server each round, making its communication complexity exactly equivalent to that of FedAvg. To see the equivalence, we prove that our implementation satisfies the following condition for any time step $t \geq 0$:

$$\begin{aligned} c_k^{t+1} &:= \frac{1}{M} \sum_{m \in [M]} \nabla L_k(\theta_k^{t,m}; \xi_k^{t,m})\,, \text{ and} \\ c^{t+1} &:= \frac{1}{K} \sum_{k \in [K]} c_k^t\,, \text{ we maintain the invariant that} \\ h_k^{t+1} &= c_k^{t+1} - c^{t+1}\,. \end{aligned}$$

To see this, note that the local client model after updating in round t is

$$\begin{aligned} \widehat{\theta}_k^{t+1} &= \widehat{\theta}_k^{t,M+1} \\ &= \theta^t - \eta \sum_{k \in [K]} \nabla L_k(\theta_k^{t,m}; \xi_k^{t,m}) - h_k^t \\ &= \theta^t - M\eta(c_k^{t+1} - h_k^t)\,. \end{aligned}$$

By averaging this over the clients, we can see that the server model is

$$\theta^{t+1} = \theta^t - M\eta\Big(c^{t+1} - \frac{1}{K} \sum_{l \in [K]} h_l^t\Big).$$

By induction, suppose that $h_k^t = c_k^t - c^t$. This implies that summing over the clients, it becomes zero; i.e., $\sum_{l \in [K]} h_l^t = 0$. Plugging this and the previous computations, we have

$$\begin{aligned} h_k^{t+1} &= h_k^t + \tfrac{1}{M\eta}(\theta^{t+1} - \widehat{\theta}_k^{t+1}) \\ &= h_k^t + \tfrac{1}{M\eta}(-M\eta c^{t+1} + M\eta(c_k^{t+1} - h_k^t)) \\ &= c_k^{t+1} - c^{t+1}\,. \end{aligned}$$

For the base step at $t = 0$, note that $h_i^0 = \mathbf{0}$. This completes the proof by induction.

C.1.2 Additional Implementation Details

[1]Note that when different clients have different number of data points, the actual aggregation step is $\theta^{t+1} = \sum_{k=1}^{K} ({n_k}/{\sum_j n_j})\widehat{\theta}_k^{t+1}$. However, we present the simplified version with equal weights for all clients to ease the comparison with the pseudocode in [134].

Additional details about linear regression in TCT. In our experiments, we normalize the one-hot encoded label of each sample so that the normalized one-hot encoded label has mean 0. More specifically, we subtract $[1/C, \dots, 1/C]^\top \in \mathbb{R}^{C\times 1}$ from the one-hot encoding label vector, where C is the number of classes. Further, [117] show that performance for large number of classes can be improved by increasing the penalty for mis-classification and scaling the target from 1 to a larger value (e.g., 30). [4] show that using Leaky-ReLU, and using K-FAC preconditioning further improves the performance. However, we do not explore such optimizations in this work– these (and other optimization tricks for least-squares regression) can be easily incorporated into our framework.

Local learning rate for TCT. From our experiments, we find that small local learning rates ($\eta \leq 10^{-4}$) achieve good train/test accuracy performance for TCT with the normalization step. When the normalization step in TCT is applied, larger local learning rates diverge. Meanwhile, local learning rates from $[10^{-6}, 10^{-4}]$ achieve similar performance for TCT (as shown in Table C.6). On the other hand, without the normalization step, TCT with large learning rate ($\eta \in [0.01, 0.5]$) does not diverge. When running more communication rounds, TCT (without the normalization step) with large learning rate achieves similar performance as the default TCT (with the normalization step).

Additional details about Stage 2 of TCT. To solve the linear regression problem in TCT-**Stage 2**, we use the full batch gradient in Eq. (C.2) of Algorithm 1 in our implementation.

Additional details about Figure 4.5. We consider CIFAR10-[`#C=2`] in Figure 4.5a and 4.5b.

Details about the total amount of compute. We use NVIDIA 2080 Ti, A4000, and A100 GPUs, and our experiments required around 500 hours of GPU time.

Algorithm 2 TCT: complete pseudo-code

```
**Input:** input dim $D$, output dim $C$, loss $\ell(\cdot,\cdot) : \mathbb{R}^{C\times C} \to \mathbb{R}$, aggr. weights $\{w_1,\dots,w_K\}$,
model $f$ with parameters $\theta \in \mathbb{R}^P$: $f(x;\theta) = \phi \circ \omega\,(x) : \mathbb{R}^D \to \mathbb{R}^C$ (e.g., ResNet18), composed
of a feature extractor $\phi : \mathbb{R}^D \to \mathbb{R}^E$ and final linear layer $\omega : \mathbb{R}^E \to \mathbb{R}^C$.
**Hyper-parameters:** Local steps $M$ (default 500), Stage-1 lr $\eta_1$ (default 0.01), Stage-1 rounds
$T_1$ (default 100), Stage-2 lr $\eta_2$ (default $5 \cdot 10^{-5}$), Stage-2 rounds $T_2$ (default 100).

**Stage 1 (Bootstrapping):**
Initialize server model $\theta^0$.
**for** round $t = 0, 1, \dots, T_1$ **do**
  **for** clients $k = 1, \dots, K$ in parallel **do**
    Receive $\theta^t$ from server and initialize client local model $\widehat{\theta}_k^{t,0} = \theta^t$.
    **for** $m = 1, \dots, M$ **do**
      Update with a mini-batch gradient sampled from local client data $(x_k^{t,m}, y_k^{t,m})$
        $$\widehat{\theta}_k^{t,m+1} = \widehat{\theta}_k^{t,m} - \eta_i \left(\nabla_\theta \ell(f(x_k^{t,m}; \theta_k^{t,m}), y_k^{t,m}) - h_k^t\right).$$
    **end for**
    Communicate $\widehat{\theta}_k^{t+1}$ to server.
  **end for**
  Aggregate $\theta^{t+1} = \frac{1}{\sum_k w_k} \sum_{k=1}^K w_k \widehat{\theta}_k^{t+1}$.
**end for**

**Stage 2 (Convexification):**
Input: Bootstrapped parameters $\theta^B$ decomposed as $\theta^B = \phi^B \circ \omega^B$.
Randomly re-initialize using fixed seed linear layer $\omega^r$ and define $\theta^0 := \phi^B \circ \omega^r$.
[Comment:] *Define basis vector* $\mathbf{e_1} := (1, 0, \dots, 0)$. *For input* $x$, $(\mathbf{e}_1^\top f(x;\theta^0))$ *is the first logit.*
Optionally, compute a random sub-sampling mask $\mathscr{S}(\phi)$ over feature params using fixed seed .

[Comment:] *For a given input* $x$*, we will learn parameters* $(\varphi, b)$ *for prediction as*
  $$\hat{y} = \varphi^\top \phi_{\text{eNTK}}(x) + b, \quad \text{where} \quad \phi_{\text{eNTK}}(x) := \mathscr{S}\left(\nabla_\phi(\mathbf{e}_1^\top f(x;\phi^B \circ \omega^r))\right).$$

Compute normalized eNTK features $\tilde{\phi}_{\text{eNTK}}(x)$ (mean 0 and variance 1) across clients.
Also normalize targets to mean 0 using $\tilde{y} := y - \frac{1}{C}\mathbf{1}$.
Run SCAFFOLD (Algorithm 1) over params $\psi := (\varphi, b)$ with learning rate $\eta_2$, local steps $M$,
initial server params: $\psi^0 = \mathbf{0}$, and client losses $\{L_k\}$ defined over the local data as
  $$L_k(\psi) := \sum_{(x_k, y_k)} \left(\varphi^\top \tilde{\phi}_{\text{eNTK}}(x_k) + b - \tilde{y}_k\right)^2.$$
```

Algorithm 3 Compute eNTK Pseudocode, PyTorch-like

```
def compute_eNTK(model, X, num_params, subsample_size=100000, seed=123):
   """compute eNTK of input X with model"""
   # model: model for linearization
   # X: (n x d), n -- number of samples, d -- input dimension
   # subsample_size: parameter of subsampling operation
   # seed: random seed for subsampling operation
   # num_params: total number of parameters for model
   model.eval()
   params = list(model.parameters())
   torch.manual_seed(seed)
   torch.cuda.manual_seed(seed)
   random_index = torch.randperm(num_params)[:subsample_size]
   eNTKs = torch.zeros((X.size()[0], subsample_size))
   for i in range(X.size()[0]):
      # compute eNTK for the i-th input
      model.zero_grad()
      model.forward(X[i:i+1])[0].backward()
      eNTK = []
      for param in params:
         if param.requires_grad:
            eNTK.append(param.grad.flatten())
      eNTK = torch.cat(eNTK)
      # subsampling
      eNTKs[i, :] = eNTK[random_index]
   return eNTKs
```

C.2 Additional Experimental Results

C.2.1 Additional Baselines

In comparison with FedAdam and FedDyn. We compare TCT to FedAdam [207], FedDyn [3], and FedNova [262] in Table C.1. We consider four settings in Table C.1, including CIFAR10 ($\#\texttt{C} = 2$), CIFAR10 ($\alpha = 0.1$), CIFAR100 ($\alpha = 0.001$), and CIFAR100 ($\alpha = 0.01$). For FedDyn, we perform similar hyperparameter selection as FedAvg; i.e., select local learning rate from $\{0.1, 0.01, 0.001\}$. For FedAdam, following recommendation by [207], we set the global learning rate as $\eta_{\text{global}} = 0.1$ and select local learning rate from $\{10^{-1}, 10^{-1.5}, 10^{-2}, 10^{-2.5}, 10^{-3}\}$. Similar to results in Table 4.2, we find that TCT significantly outperforms the existing methods in high data heterogeneity settings.

Table C.1: The top-1 test accuracy ($\%$) of our algorithm (TCT) vs. other federated learning algorithms (FedAdam [207], FedDyn [3], and FedNova [262]) evaluated on CIFAR10 and CIFAR100. We vary the degree of data heterogeneity by controlling the α parameter of the symmetric Dirichlet distribution $\text{Dir}_K(\alpha)$ and the $\#\texttt{C}$ parameter for assigning how many labels each client owns. Higher accuracy is better. The highest top-1 accuracy in each setting is highlighted in **bold**.

Methods	**Datasets**			
	CIFAR10 ($\#\texttt{C} = 2$)	CIFAR10 ($\alpha = 0.1$)	CIFAR100 ($\alpha = 0.001$)	CIFAR100 ($\alpha = 0.01$)
FedAdam [207]	33.52%	62.57%	30.85%	37.16%
FedDyn [3]	51.67%	81.03%	50.86%	53.79%
FedNova [262]	53.27%	84.26%	56.06%	58.47%
TCT	**83.02%**	**89.21%**	**69.07%**	**69.66%**

C.2.2 Results of Other Architectures

In Section 4.4, we use batch normalization [120] as the default normalization layer on CIFAR10 and CIFAR100 datasets, and we denote the ResNet-18 with batch normalization layers by ResNet-18-BN. In Table C.2, we consider group normalization [274] on CIFAR10 and CIFAR100 and let ResNet-18-GN denote the ResNet-18 with group normalization. We set `num_groups=2` in group normalization layers. As shown in Table C.2, TCT achieves better performance than FedAvg with ResNet-18-GN on both CIFAR10 and CIFAR100 datasets. Our experiments indicate that in extremely heterogeneous settings, group norm is insufficient to fix FedAvg.

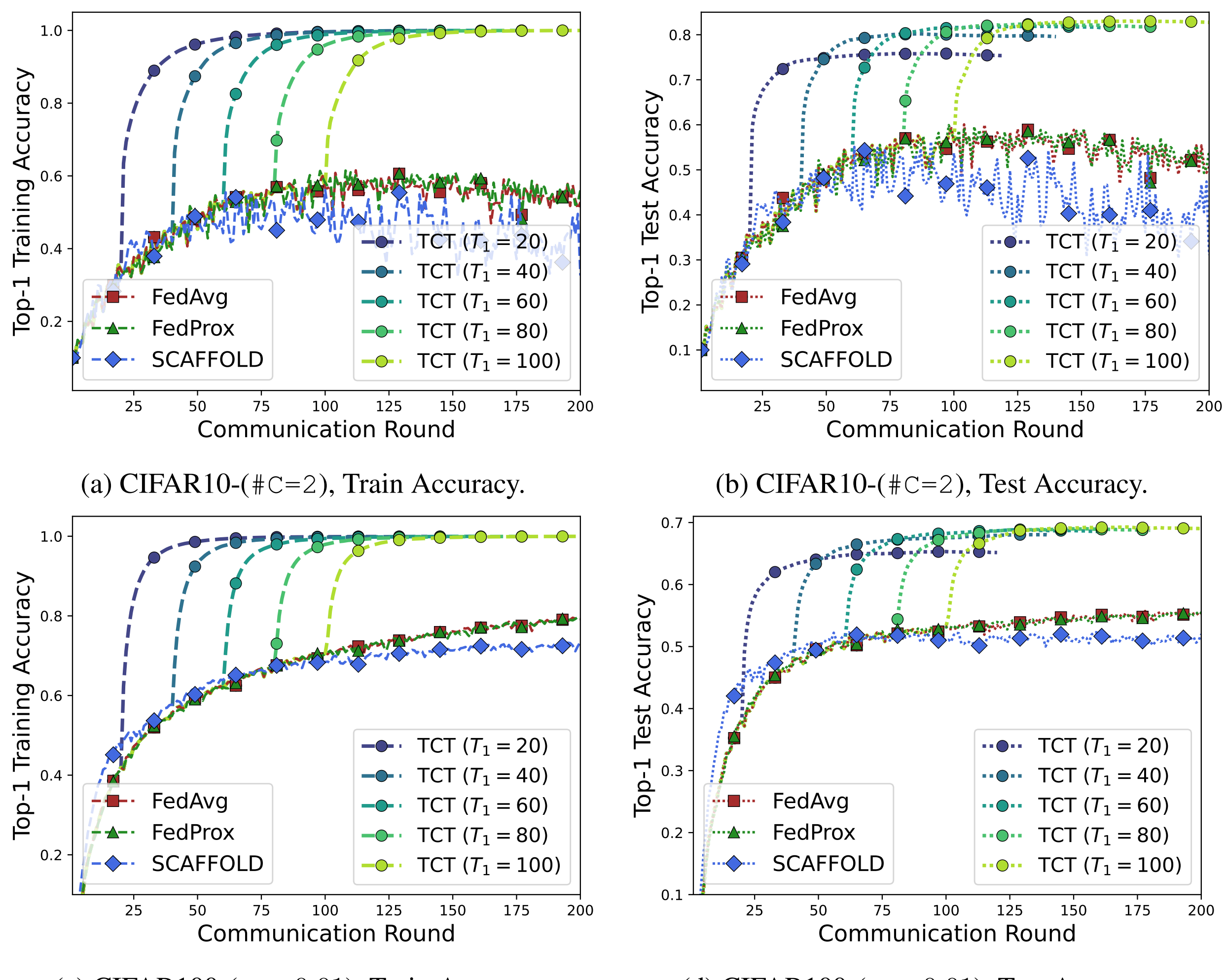
C.2.3 Additional Experimental Results of the Effect of Stage 1 Communication Round for TCT

In Figure C.1, we provide additional results of the effect of T_1 for TCT on CIFAR10 and CIFAR100 datasets. We find that TCT outperforms existing algorithm across all T_1 communication rounds, where $T_1 \geq 20$. Extending the number of rounds for the baseline algorithms to 200 rounds does

Table C.2: The top-1 test accuracy ($\%$) of our algorithm (TCT) vs. FedAvg(-GN) evaluated on CIFAR10 and CIFAR100. We vary the degree of data heterogeneity by controlling the α parameter of the symmetric Dirichlet distribution $\text{Dir}_K(\alpha)$ and the $\texttt{\#C}$ parameter for assigning how many labels each client owns. Higher accuracy is better. The highest top-1 accuracy in each setting is highlighted in **bold**.

Datasets	**Architectures**	**Methods**	**Non-i.i.d. degree**			
			$\texttt{\#C} = 1$	$\texttt{\#C} = 2$	$\alpha = 0.1$	$\alpha = 0.5$
CIFAR-10	ResNet-18-GN	FedAvg	21.23%	56.80%	84.72%	89.03%
	ResNet-18-BN	FedAvg	11.27%	56.86%	82.60%	90.43%
	ResNet-18-BN	*TCT*	**49.92%**	**83.02%**	**89.21%**	**91.10%**
			$\alpha = 0.001$	$\alpha = 0.01$	$\alpha = 0.1$	$\alpha = 0.5$
CIFAR-100	ResNet-18-GN	FedAvg	47.60%	48.60%	53.29%	55.39%
	ResNet-18-BN	FedAvg	53.89%	54.22%	63.49%	67.65%
	ResNet-18-BN	*TCT*	**68.42%**	**69.07%**	**69.66%**	**69.68%**

not improve their performance. In contrast, running 60 rounds of bootstrapping using FedAvg followed by 40 rounds of TCT gives near-optimal performance across all settings.



(a) CIFAR10-(#C=2), Train Accuracy.

(b) CIFAR10-(#C=2), Test Accuracy.

(c) CIFAR100-($\alpha = 0.01$), Train Accuracy.

(d) CIFAR100-($\alpha = 0.01$), Test Accuracy.

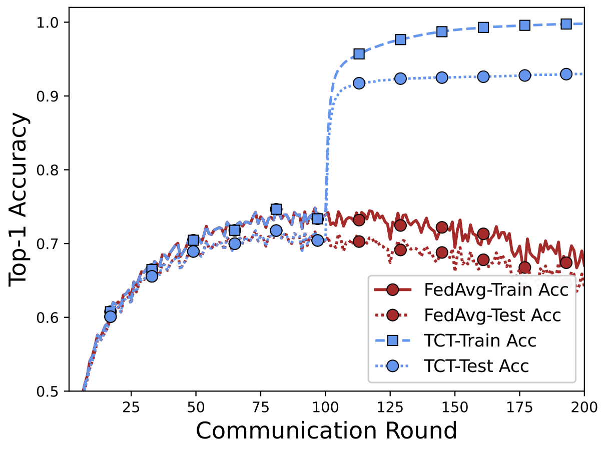
Figure C.1: We evaluate TCT on using checkpoints saved at different communication rounds T_1 in **Stage 1**. We compare TCT to existing algorithms, including FedAvg, FedProx, and SCAFFOLD. For all three existing algorithms, we visualize the results of local learning rate $\eta = 0.1$. The train/test accuracy results in the first T_1 communication rounds of TCT are the same as FedAvg. For example, "TCT ($T_1 = 20$)" corresponds to training the model with FedAvg for $T_1 = 20$ rounds in **Stage 1** and then running 100 rounds of SCAFFOLD for solving the linear regression problem in **Stage 2**. Plots **(a)** and **(c)** display training accuracy and **(b)** and **(d)** display test accuracy.

C.2.4 Additional Experimental Results of Pre-trained Models

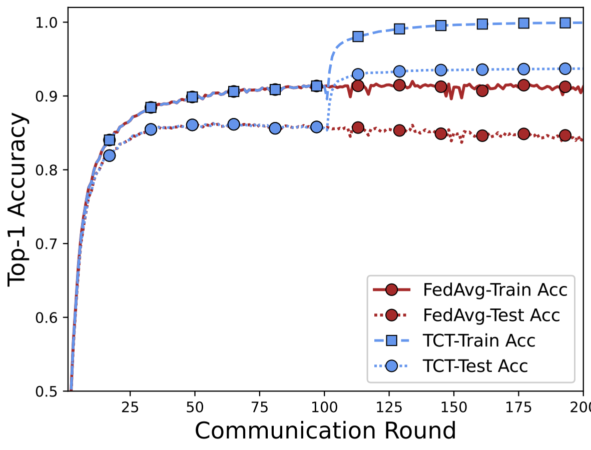
In Table C.3 and Figure C.2, we provide additional results of the effect of pre-training for FedAvg and TCT on CIFAR10 and CIFAR100 datasets. For both methods, we use the ResNet-18 pre-trained on ImageNet-1k [100] as the initialization. We use *FedAvg (last layer)* to denote applying FedAvg on learning the last linear layer of the model, i.e., layers except for the last linear layer are freezed during training. Compared to results in Table 4.2, we find that using pre-trained model as initialization largely improves the performance of both FedAvg and TCT. However, FedAvg still suffers from data heterogeneity. In contrast, TCT achieves similar performance as the centralized setting on both datasets across different degrees of data heterogeneity.

Table C.3: The top-1 test accuracy ($\%$) of our algorithm (TCT) vs. FedAvg evaluated on CIFAR10 and CIFAR100 with pre-trained model initialization. We vary the degree of data heterogeneity by controlling the α parameter of the symmetric Dirichlet distribution $\mathrm{Dir}_K(\alpha)$ and the #C parameter for assigning how many labels each client owns. Higher accuracy is better. The highest top-1 accuracy in each setting is highlighted in **bold**.

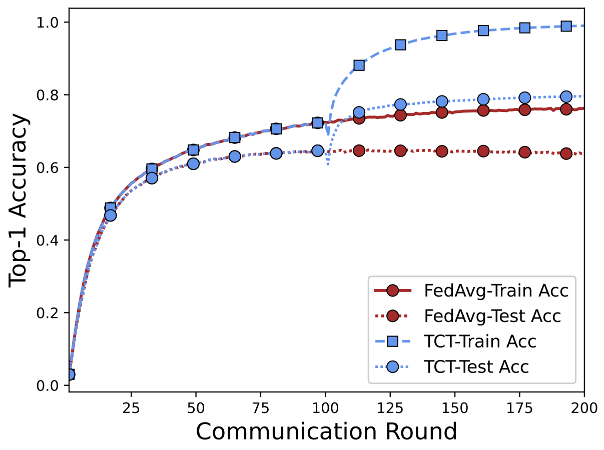
Methods	**Datasets**			
	CIFAR10 (#C $= 2$)	CIFAR10 ($\alpha = 0.1$)	CIFAR100 ($\alpha = 0.001$)	CIFAR100 ($\alpha = 0.01$)
Centralized	95.13%	95.13%	80.65%	80.65%
FedAvg (last layer)	63.60%	75.16%	50.40%	51.97%
FedAvg	64.73%	84.25%	62.23%	63.81%
TCT	**92.97%**	**93.70%**	**79.25%**	**79.55%**



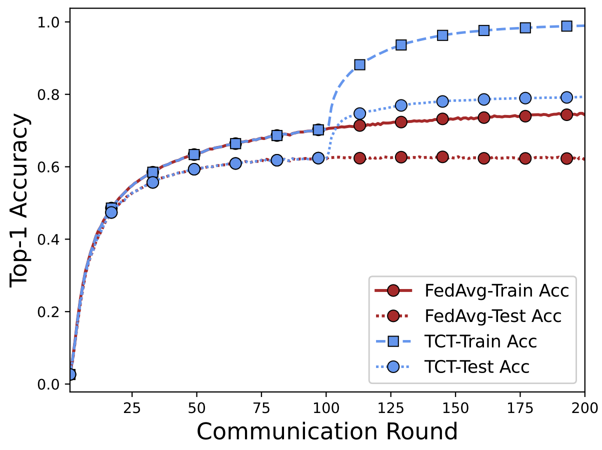
(a) CIFAR10-(#C=2).



(b) CIFAR10-($\alpha = 0.1$).



(c) CIFAR100-($\alpha = 0.01$).



(d) CIFAR100-($\alpha = 0.001$).

Figure C.2: We evaluate FedAvg and TCT on CIFAR10 and CIFAR100 datasets with pre-trained ResNet-18. Plots **(a)** and **(b)** display training/test accuracy on the CIFAR10 dataset and **(c)** and **(d)** display training/test accuracy on the CIFAR100 dataset.

C.2.5 Additional Experimental Results of One-round Communication

In Table C.4, we provide additional results of TCT on CIFAR10 and CIFAR100 datasets with one communication round in TCT-Stage 2. Specifically, we set the number of local steps $M = 500$, local learning rate $\eta = 0.00005$, and the total number of communication round $T = 1$ in TCT-Stage 2. The results are summarized in Table C.4. With only one communication round in TCT-Stage 2, TCT still achieves better performance than FedAvg in three out of four settings in Table C.4. On the other hand, we recommend setting the communication round in TCT-Stage 2 larger than 10 for our method TCT in order to achieve satisfying performance.

Table C.4: The top-1 test accuracy ($\%$) of our algorithm (TCT) on CIFAR10 and CIFAR100 with one communication round in TCT-Stage 2.

Methods	**Datasets**			
	CIFAR10 ($\#\texttt{C} = 2$)	CIFAR10 ($\alpha = 0.01$)	CIFAR100 ($\alpha = 0.001$)	CIFAR100 ($\alpha = 0.01$)
FedAvg	56.86%	82.60%	53.89%	54.22%
TCT	83.02%	89.21%	68.42%	69.07%
TCT-OneRound	64.94%	82.62%	55.50%	57.51%

C.2.6 Additional Experimental Results of Large Number of Clients

We study the performance of our proposed algorithm as well as existing algorithms in the large number of clients setting, where we consider the number of clients $K = 50$ on CIFAR100 with $\alpha = 0.001$. The results are summarized in Table C.5. We find our proposed method (TCT: 45.32%) significantly outperforms existing methods (best test accuracy: 16.70%).

Table C.5: The top-1 accuracy ($\%$) of our algorithm (TCT) vs. state-of-the-art federated learning algorithms evaluated on CIFAR100 with a large number of clients. We set the degree of data heterogeneity parameter $\alpha = 0.001$ and set the total number of clients $K = 50$. Higher accuracy is better. The highest top-1 accuracy is highlighted in **bold**.

Dataset	**Architecture**	**# Clients**	**FedAvg**	**FedProx**	**SCAFFOLD**	**TCT**
CIFAR100 ($\alpha = 0.001$)	ResNet-18	50	16.70%	16.24%	13.41%	**45.32%**

C.2.7 Additional Ablations

Effect of local learning rate for TCT and FedAdam. As mentioned in [207], FedAdam is more robust to the choice of local learning rate compared to FedAvg. We conduct additional ablations on the effect of local learning rate for TCT as well as FedAdam [207] on the CIFAR10 dataset. For each algorithm, we first select a base local learning rate η_{base} and then vary the local learning rate $\eta \in \{\eta_{\text{base}} \cdot 10^{0}, \eta_{\text{base}} \cdot 10^{-0.5}, \eta_{\text{base}} \cdot 10^{-1.0}, \eta_{\text{base}} \cdot 10^{-1.5}\}$. The results are summarized in

Table C.6. Compared to FedAdam, we find that TCT is much less sensitive to the choice of local learning rate.

Table C.6: The top-1 test accuracy ($\%$) of our algorithm (TCT) and FedAdam [207] evaluated on the CIFAR10 dataset. We vary the local learning rate for both algorithms. Higher accuracy is better.

Datasets	**Methods**	**Local learning rate**			
	($\eta_{\text{base}} = 10^{-4}$)	$\eta = \eta_{\text{base}} \cdot 10^{0}$	$\eta = \eta_{\text{base}} \cdot 10^{-0.5}$	$\eta = \eta_{\text{base}} \cdot 10^{-1.0}$	$\eta = \eta_{\text{base}} \cdot 10^{-1.5}$
CIFAR10-(#C=2)	TCT	82.12%	83.60%	83.51%	82.37%
CIFAR10-($\alpha = 0.1$)	TCT	88.68%	89.27%	89.23%	89.15%
	($\eta_{\text{base}} = 10^{-1.5}$)	$\eta = \eta_{\text{base}} \cdot 10^{0}$	$\eta = \eta_{\text{base}} \cdot 10^{-0.5}$	$\eta = \eta_{\text{base}} \cdot 10^{-1.0}$	$\eta = \eta_{\text{base}} \cdot 10^{-1.5}$
CIFAR10-(#C=2)	FedAdam [207]	31.29%	33.52%	26.20%	14.96%
CIFAR10-($\alpha = 0.1$)	FedAdam [207]	10.31%	37.26%	62.57%	49.18%

Effect of local learning rate and number of local steps for TCT. We conduct additional ablations on the effect of both the local learning rate η and the number of local steps M for TCT on CIFAR10 and CIFAR100 datasets. The results in Table C.7 and Table C.8 indicate that TCT is robust to the choice of the local learning rate η and the number of local steps M. We find that as the number of steps increases, the learning rate should predictably decrease. The performance is relatively stable along the diagonal, indicating that it is the product $M \cdot \eta$ which affects accuracy.

Table C.7: The top-1 test accuracy ($\%$) of our algorithm (TCT) evaluated on the CIFAR10 dataset. We consider CIFAR10-(#C=2) and we set $\eta_{\text{base}} = 10^{-4}$. We vary both the local learning rate and the number of local steps for TCT. Higher accuracy is better.

Number of local steps	**Local learning rate**			
	$\eta = \eta_{\text{base}} \cdot 10^{0}$	$\eta = \eta_{\text{base}} \cdot 10^{-0.5}$	$\eta = \eta_{\text{base}} \cdot 10^{-1.0}$	$\eta = \eta_{\text{base}} \cdot 10^{-1.5}$
$M = 50$	83.51%	82.37%	80.67%	78.14%
$M = 100$	83.59%	83.35%	81.73%	79.71%
$M = 500$	82.12%	83.60%	83.51%	82.37%
$M = 1000$	80.78%	82.92%	83.59%	83.35%

Effect of training loss for TCT-Stage 2. We compare the performance of quadratic loss (defined in Eq. (4.1)) and cross-entropy loss (i.e., applying the cross-entropy loss for learning the linear model in TCT-Stage 2, denoted by *TCT-CE*) for TCT in Table C.9. As shown in Table C.9, we find quadratic loss indeed achieves better performance than cross-entropy loss for TCT.

Effect of subsampling for TCT-Stage 2. We study the performance of full eNTK representation in TCT-Stage 2 (i.e., without random subsampling) to investigate the role of the subsampling

Table C.8: The top-1 test accuracy ($\%$) of our algorithm (TCT) evaluated on the CIFAR100 dataset. We consider CIFAR100-($\alpha = 0.01$) and we set $\eta_{\text{base}} = 10^{-4}$. We vary both the local learning rate and the number of local steps for TCT. Higher accuracy is better.

Number of local steps	**Local learning rate**			
	$\eta = \eta_{\text{base}} \cdot 10^{0}$	$\eta = \eta_{\text{base}} \cdot 10^{-0.5}$	$\eta = \eta_{\text{base}} \cdot 10^{-1.0}$	$\eta = \eta_{\text{base}} \cdot 10^{-1.5}$
$M = 50$	69.12%	67.31%	64.61%	61.34%
$M = 100$	69.54%	68.48%	66.43%	63.42%
$M = 500$	69.03%	69.60%	69.12%	67.31%
$M = 1000$	68.38%	69.42%	69.54%	68.48%

approximation. We provide the results in Table C.9. As shown in Table C.9, applying full eNTK representations slightly improves (improvements are smaller than 2% across all settings) the performance of TCT on CIFAR10/100. On the other hand, using subsampled eNTK reduces the communication cost more than 100x compared to the full eNTK and existing federated learning algorithms (#parameter of the whole model: 11,169,345, #parameters of the subsample eNTK: 100,000).

Applying last layer representations in TCT-Stage 2. We study the performance of only applying last layer representations in TCT-Stage 2, and the results are summarized in Table C.10. From Table C.10, we find that applying eNTK representations with high dimension (i.e., $p = 100,000$) outperforms using the representations before the last layer only, especially in the settings with high degrees of data heterogeneity. These results provide further evidence on applying eNTK features instead of the representations before the last layer features.

Table C.9: The top-1 accuracy (%) of our algorithm (TCT) vs. TCT-CE, TCT-full-eNTK on FMNIST, CIFAR10, and CIFAR100. We vary the degree of data heterogeneity by controlling the α parameter of the symmetric Dirichlet distribution $\mathrm{Dir}_K(\alpha)$ and the `#C` parameter for assigning how many labels each client owns. Higher accuracy is better. TCT-CE represents the variant of TCT where we apply cross-entropy loss in Stage 2 of TCT. TCT-full-eNTK represents the variant of TCT where we use the full eNTK representation (without subsampling) in Stage 1 of TCT.

Datasets	**Architectures**	**Methods**	**Non-i.i.d. degree**			
			$\#\mathrm{C}=1$	$\#\mathrm{C}=2$	$\alpha=0.1$	$\alpha=0.5$
FMNIST	SimpleCNN	*TCT*	86.32%	90.33%	90.78%	91.13%
		TCT-CE	86.50%	89.23%	89.66%	90.15%
		TCT-full-eNTK	86.32%	90.36%	90.90%	91.18%
			$\#\mathrm{C}=1$	$\#\mathrm{C}=2$	$\alpha=0.1$	$\alpha=0.5$
CIFAR-10	ResNet-18	*TCT*	49.92%	83.02%	89.21%	91.10%
		TCT-CE	45.13%	81.06%	88.03%	91.12%
		TCT-full-eNTK	50.38%	84.92%	89.72%	91.69%
			$\alpha=0.001$	$\alpha=0.01$	$\alpha=0.1$	$\alpha=0.5$
CIFAR-100	ResNet-18	*TCT*	68.42%	69.07%	69.66%	69.68%
		TCT-CE	63.46%	64.08%	65.22%	66.07%
		TCT-full-eNTK	69.81%	70.05%	70.12%	70.91%

Table C.10: The top-1 accuracy (%) of our algorithm (TCT) vs. TCT-last-layer on FMNIST, CIFAR10, and CIFAR100. We vary the degree of data heterogeneity by controlling the α parameter of the symmetric Dirichlet distribution $\mathrm{Dir}_K(\alpha)$ and the `#C` parameter for assigning how many labels each client owns. Higher accuracy is better. The highest top-1 accuracy in each setting is highlighted in **bold**. TCT-last-layer represents the variant of TCT where we apply the representations of last layer and cross-entropy loss in Stage 2 of TCT.

Datasets	**Architectures**	**Methods**	**Non-i.i.d. degree**			
			$\#\mathrm{C}=1$	$\#\mathrm{C}=2$	$\alpha=0.1$	$\alpha=0.5$
FMNIST	SimpleCNN	*TCT*	**86.32%**	**90.33%**	**90.78%**	**91.13%**
		TCT-last-layer	60.43%	83.96%	86.01%	89.33%
			$\#\mathrm{C}=1$	$\#\mathrm{C}=2$	$\alpha=0.1$	$\alpha=0.5$
CIFAR-10	ResNet-18	*TCT*	**49.92%**	**83.02%**	**89.21%**	**91.10%**
		TCT-last-layer	35.51%	74.55%	86.57%	90.76%
			$\alpha=0.001$	$\alpha=0.01$	$\alpha=0.1$	$\alpha=0.5$
CIFAR-100	ResNet-18	*TCT*	**68.42%**	**69.07%**	**69.66%**	**69.68%**
		TCT-last-layer	59.80%	60.04%	64.98%	66.22%

Appendix D

Appendix: Differentially Private Representation Learning

D.1 Implementation and Evaluation Details

In this section, we provide implementation details for training and evaluating *(Syn)-ViP*, *ViP*, as well as other existing methods.

D.1.1 Details for MAE model

In Table D.1, we provide details for backbones of MAE model with different model sizes. Both MAE-Large and MAE-Base encoders are constructed following the identical setup described in [93].

Table D.1: Details of MAE backbone variants used in *ViP*.

ViP model	MAE Backbone	Encoder depth	Encoder width	Decoder depth	Decoder width	# parameters
ViP-Nano	MAE-Nano	12	192	4	512	18.6M
ViP-Tiny	MAE-Tiny	12	384	4	512	34.8M
ViP-Small	MAE-Small	12	576	4	512	61.6M
ViP-Base	MAE-Base	12	768	4	512	99.0M
ViP-Large	MAE-Large	24	1024	4	512	233.3M

D.1.2 Details for ViP Pre-training

For *(Syn)-ViP* pre-training, we follow the training setup outlined in [93]: we apply the training parameters specified in Table 8 of [93] and pre-train pre-train *(Syn)-ViP* on the S21k dataset developed in [16], which comprises of 1,300,000 training samples, for a total of 1,000 epochs. Our *(Syn)-ViP* pre-training applies the self-supervised MAE training methodology and does not use the label information available in the S21k dataset.

We now present details for differentially private *ViP* pre-training. As mentioned in Section 5.3, we first initialize the model weights with *(Syn)-ViP* pre-trained on S21k dataset. Then we apply DP-AdamW[1]. See the table below for training hyperparameters.

Model	lr (η)	warmup iterations	wd (λ)	(β_1, β_2)	epsilon (ϵ)	lr decay
ViP-Base	$3.84 \cdot 10^{-4}$	1,000	0.005	(0.9, 0.95)	10^{-8}	cosine

For masking in the MAE training, we follow the random masking strategy and masking ratio of 75% in [93] for both *(Syn)-ViP* pre-training and *ViP* pre-training. The process of executing each iteration of DP-AdamW for training the *ViP-Base* model takes approximately 25 seconds when utilizing 48 A100 (40GB) GPUs. Each epoch of the *(Syn)-ViP-Base* model's training process takes roughly 90 seconds to complete with 48 A100 (40GB) GPUs.

D.1.3 Details for Downstream Classification Task

Linear probing. We follow the training setup in [93]: we apply BatchNorm [120] before the last linear layer, and use the LARS [285] optimizer. We choose the base learning rate `blr` $\in \{0.1, 0.05, 0.01\}$, batch size $B = 16,384$, weight decay $\lambda = 0.0$. We set warmup epoch as 10, and total training epoch as 90. We use the `RandomResizedCrop` and `RandomHorizontalFlip` augmentations.

Few-shot fine-tuning. For vision transformer based architectures, we apply the AdamW optimizer with learning rate of lr $\in \{3 \cdot 10^{-3}, 3 \cdot 10^{-4}, 3 \cdot 10^{-5}\}$ and set weight decay as 0.05. For convolutional neural networks (AlexNet, ResNet used in SimCLR), we apply the SGD optimizer because it consistently outperforms AdamW. We select learning rate lr $\in \{1 \cdot 10^{-2}, 1 \cdot 10^{-3}, 1 \cdot 10^{-4}\}$, while setting the momentum as 0.9 and the weight decay as 0.0. For all models we apply the cosine learning rate decay, and use 10 warm-up epochs and fine-tine with 200 total epochs. We apply AutoAugment [50] for data augmentation.

D.1.4 Details for Downstream Segmentation and Detection Tasks

COCO object detection and segmentation. We fine-tune the pre-trained *(Syn)-ViP* and *ViP* on COCO with the `Detectron2` package [275]. We apply the pre-trained *(Syn)-ViP-Base* and *ViP-Base* as the ViT initializations for the detection and segmentation tasks, and apply the default hyperparameter config in `Detectron2` for ViTDet-Base.

ADE20K semantic segmentation. We follow the setup described in [93] on evaluating pre-trained MAE models for semantic segmentation. We apply the UPerNet [278] and perform fine-tuning for 100 epochs with a batch size of 16.

[1]A variant of the standard DP-SGD — we first compute the noisy clipped stochastic gradient described in (5.3), then apply one step update of AdamW [170] using the estimated gradient.

D.1.5 Details for Differentially Private Fine-tuning on ImageNet

We use the pre-trained encoders of *(Syn)-ViP* and *ViP* and apply DP-AdamW for DP end-to-end fine-tuning. The details for parameters in DP-AdamW can found in the following table.

Model	sampling ratio q	noise σ	iterations T	lr	wd
ViP-Base / (Syn)-ViP-Base	$262,144/n$	5.6	1,500	$1.02 \cdot 10^{-3}$	0.005

We use 50 iterations for learning rate warm-up, and then keep the learning rate constant afterwards. For selecting parameters not presented in the aforementioned table, we adopt the default configuration of AdamW in `PyTorch` [195]. The fine-tuned model satisfies $(8, 8 \cdot 10^{-7})$-DP on the ImageNet-1K dataset in addition to the LAION233M dataset.

D.1.6 Details for Figure 5.1

For the linear probing results, we present the performance of the ViP-Large model, with the summarized results shown in the last row of Table D.1. Regarding the detection and segmentation results, we utilize the ViP-Base model as the ViT backbone, and the corresponding outcomes can be found in Table D.2.

D.2 Additional Experimental Results

In this section, we provide additional experimental results on evaluating *(Syn)-ViP*, *ViP*, as well as other existing methods.

D.2.1 Segmentation and Detection Evaluations of (Syn)-ViP/ViP

We summarize the results for object detection and segmentation in Table D.2. Training details can be found in Appendix D.1.4.

D.2.2 Additional Experiments on ViP Pre-training

In Figure D.1, we plot the training loss v.s. number of training steps for *ViP* training *without (Syn)-ViP initialization*. Compared to the results in Figure 5.4a, when pre-training from scracth with DP-AdamW, larger models do not converge faster than smaller ones. These results further demonstrate the effectiveness of synthetic pre-training for unlocking DP-SGD training of larger vision models.

D.2.3 DP Fine-tuning ViP on ImageNet-1K

Thus far, our main emphasis has been on evaluating DP pre-trained ViP through *non-private* linear probing or fine-tuning on downstream tasks. For certain use cases, the downstream task training

Table D.2: Evaluation of our DP models (*(Syn)-ViP*, *ViP*) as well as existing non-private baselines on COCO object detection/segmentation and ADE20K semantic segmentation.

Model	DP?	COCO AP^{box}	COCO AP^{mask}	ADE20K mIoU
SimCLR [39]	✗	37.9	33.3	-
Mask R-CNN [98]	✗	40.0	37.1	-
RefineNet [163]	✗	-	-	40.7
MAE [93]	✗	50.3	44.9	48.1
(Syn)-ViP	✓	45.0	40.1	38.8
ViP	✓	45.2	40.4	40.1

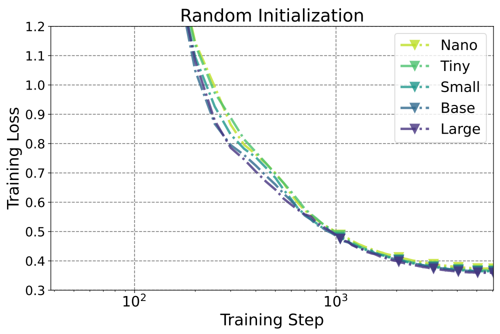


Figure D.1: Training loss of different model sizes. (with random initialization).

set may be privacy-sensitive as well and DP fine-tuning is required. We simulate such a scenario by fine-tuning the privately pre-trained ViP model[2] on ImageNet-1K with DP-SGD. As a result, the fine-tuned model satisfies $(8, 8 \cdot 10^{-7})$-DP on the ImageNet-1K dataset in addition to the LAION233M dataset. We compare against prior works on training DP ImageNet models without pre-training [149, 54, 215]; results are summarized in Table D.3.

By utilizing our pre-trained *ViP* as an initialization, we observe an improvement in top-1 accuracy of more than 10% compared to the previous SOTA [215], demonstrating the efficacy of our DP pre-training recipe.

D.2.4 Additional Experiments on the Classification Task

Comparison with non-private MAE. To gain a better understanding of the gap between non-private training and private training, we use the same synthetic pre-trained model as initialization and perform DP-AdamW training on LAION233M with $\sigma = 0.0$[3]. We keep most of the training parameters the same except for setting the sampling ratio to $q = 4096/n$ and the number of itera-

[2]ViP-Base pre-trained on LAION233 shown in the last row of Table 5.1.

[3]In this case, the $\epsilon = +\infty$ for the (ϵ, δ)-DP.

Table D.3: DP fine-tuning evaluation on ImageNet-1K. We compare (Syn)-ViP and ViP with existing DP training methods (DP-ResNet-18, DP-NFNet, and TAN) on ImageNet-1K.

Model	(ϵ, δ)-DP	Top-1 Accuracy
DP-ResNet-18 [149]	$(13.2, 10^{-6})$	6.2%
DP-NFNet [54]	$(8, 8 \cdot 10^{-7})$	32.4%
TAN [215]	$(8, 8 \cdot 10^{-7})$	39.2%
(Syn)-ViP	$(8, 8 \cdot 10^{-7})$	48.9% $\pm$ 0.2
ViP	$(8, 8 \cdot 10^{-7})$	**50.3%** $\pm$ 0.3

tions $T = 60,000$[4]. We then evaluate the linear probing (few-shot fine-tuning) performance of the trained model and provide the results in Table D.4 (Table D.5).

For linear probing, our *ViP* model closes more than half the gap between the *(Syn)-ViP* model and the non-private MAE model. With a more refined training recipe, it is plausible that the gap can be reduced even further, allowing DP-trained foundation vision models to rival non-privately trained ones on certain downstream tasks. In the context of few-shot fine-tuning, a comparison between private learning and the non-private MAE model reveals considerable potential for improvement in the private learning approach.

Comparison with ViP trained on de-duplicated LAION-2B. Recent work has demonstrated that there exist duplicated samples in the LAION dataset, which poses copyright and privacy challenges for foundation models trained on LAION. Therefore, we also pre-train our proposed ViP model on a de-duplicated subset of LAION-2B [216], denoted by Dedup-LAION-245M, which consists of a similar number of training samples (245 million) as the one we mainly consider in this work. We summarize the linear probing performance of the ViP pre-trained on Dedup-LAION-245M in Table D.4. We find the ViP model pre-trained on the de-duplicated LAION achieves similar performance as the one trained on LAION-400M [217].

Table D.4: Linear probing evaluation on downstream classification. We compare *ViP* and *(Syn)-ViP* with (non-private) MAE [93].

Model	Pre-train dataset	DP?	SSL?	ImageNet-1K‡	Places-365	Places-205	iNat-2021
(non-private) MAE	LAION-233M	✗	✓	60.5%	48.3%	51.8%	38.5%
(Syn)-ViP	LAION-233M	✓	✓	49.8%	43.2%	45.8%	32.4%
ViP	LAION-233M	✓	✓	55.7%	46.1%	48.5%	38.1%
ViP	Dedup-LAION-245M	✓	✓	55.5%	46.3%	48.1%	38.0%

[4]While the trained model may not necessarily achieve optimal performance, our main purpose is to present a non-private model that follows a similar training setup, with the exception of setting the noise to zero. This allows us to compare its performance to the private model.

Table D.5: Fine-tuning evaluation on few-shot downstream classification. We compare *ViP* and *(Syn)-ViP* with (non-private) MAE [93].

Model	Aircraft			Caltech-101			CIFAR-100		
	10-shot	20-shot	30-shot	5-shot	10-shot	30-shot	5-shot	10-shot	30-shot
(non-private) MAE	36.78%	56.82%	66.20%	72.93%	84.50%	92.78%	34.38%	47.98%	62.88%
(Syn)-ViP	21.79%	46.85%	58.45%	60.51%	76.21%	88.48%	27.62%	38.96%	55.84%
ViP	**31.62%**	**53.05%**	**64.26%**	**68.05%**	**79.03%**	**88.90%**	**30.73%**	**40.95%**	**57.52%**

Linear probing evaluation of ViP with different model sizes. We study the scaling behavior of ViP and (Syn)-ViP through linear probing. As shown in Table D.7, we compare the performance of ViP and (Syn)-ViP with different model sizes. The performance of ViP consistently improves across all datasets as the model size increases. In contrast, increasing the model size from MAE-Base to MAE-Large results in less than 1% improvement in top-1 accuracy for (Syn)-ViP. These findings further underscore the effectiveness of our proposed ViP training recipe for scaling up model size in private pre-training.

Table D.6: Linear probing evaluation of *ViP-LAION* with different privacy budget on ImageNet-1k classification. We vary the privacy budget epsilon (ϵ) from 2.0 to $+\infty$, where our default privacy budget is $\epsilon = 8.0$ and we use $\epsilon = +\infty$ to denote the non-private MAE model.

Model	Downstream dataset	$\epsilon = 2.0$	$\epsilon = 4.0$	$\epsilon = 8.0$	$\epsilon = +\infty$
ViP-LAION	ImageNet-1k	51.4%	53.8%	55.7%	60.5%

Table D.7: Additional linear probing evaluation on downstream classification (ViP with different model sizes).

Model	# parameters	Backbone	ImageNet-1K	Places-365	Places-205	iNat-2021
(Syn)-ViP-S	61.6M	MAE-Small	46.0%	40.9%	43.2%	28.3%
(Syn)-ViP-B	99.0M	MAE-Base	49.8%	43.2%	45.8%	32.4%
(Syn)-ViP-L	233.3M	MAE-Large	50.2%	43.3%	46.5%	32.7%
ViP-S	61.6M	MAE-Small	49.6%	42.4%	44.7%	30.0%
ViP-B	99.0M	MAE-Base	55.7%	46.1%	48.5%	38.1%
ViP-L	233.3M	MAE-Large	**58.0%**	**48.5%**	**50.8%**	**40.6%**

D.2.5 ViP Ablation Experiments

We study the effect of dataset size and batch size in *ViP* pre-training, and evaluate different models with linear probing and fine-tuning on ImageNet-1K. We consider the *ViP-Base* setting and the results are summarized in Table D.8.

Table D.8: Ablation studies on the effect of dataset size and batch size. The first row shows the result of (Syn)-ViP, which is the common starting point for all models in the subsequent rows. Difference in performance compared to (Syn)-ViP is shown in parentheses. See text for details. (‡ represents linear probing evaluation and ◇ represents 10-shot fine-tuning evaluation.)

Model	Batch Size	# Train data	Noise σ	ImageNet-1K ‡	Places-365 ‡	iNat-2021‡	Aircraft◇	CIFAR-100◇
(Syn)-ViP	-	-	-	49.8%	43.2%	32.4%	21.8%	39.0%
ViP	98,304	2M	2.50	52.6% (+2.8%)	44.8% (+1.6%)	37.0% (+4.6%)	29.1% (+7.3%)	39.9% (+0.9%)
ViP	98,304	23M	0.66	53.7% (+3.9%)	45.2% (+2.0%)	37.6% (+5.2%)	31.5% (+9.7%)	40.5% (+1.5%)
ViP	98,304	233M	0.48	55.7% (+5.9%)	46.1% (+2.9%)	38.1% (+5.7%)	31.6% (+9.8%)	41.0% (+2.0%)
ViP	8,192	233M	0.41	43.9% (- 5.9%)	41.0% (- 2.2%)	27.6% (- 4.8%)	15.0% (- 6.8%)	39.2% (+0.2%)
ViP	32,768	233M	0.45	53.0% (+3.2%)	45.1% (+1.9%)	36.2% (+3.8%)	30.0% (+8.2%)	40.3% (+1.3%)
ViP	98,304	233M	0.48	55.7% (+5.9%)	46.1% (+2.9%)	38.1% (+5.7%)	31.6% (+9.8%)	41.0% (+2.0%)

We study the effect of MAE-decoder depth and MAE-masking ratio in *ViP* pre-training, and evaluate different models with linear probing on ImageNet-1K. We consider the *ViP-Base* setting and the results are summarized in Table D.9.

Table D.9: Ablation studies on the effect of decoder depth and masking ratio in MAE.

Model	decoder depth	masking ratio	ImageNet-1K
ViP (default)	4	0.75	55.7%
ViP	*1*	0.75	43.4%
ViP	*2*	0.75	51.7%
ViP	*8*	0.75	50.1%
ViP	4	*0.25*	53.5%
ViP	4	*0.5*	54.7%