Thesis Proposal: Learning Generalizable and Transferable Representations with Big Models

Runtian Zhai

August 2024

1 Introduction

This thesis aims to study the theory and algorithms of representation learning, which is one of the main driving forces behind the recent boom in artificial intelligence (AI) and machine learning (ML). In particular, this thesis studies why and how big models can learn (a) generalizable, and (b) transferable representations of real-world data.

Since around 2018, machine learning (ML) has been shifting from end-to-end training to a new paradigm powered by representation learning and *foundation models* [4], very big models trained on very large generic data sets such as Wikipedia. Given the current momentum of this new paradigm, there is no doubt that it will shape the development in AI in the near future. Thus, this thesis studies a timely matter.

Generalization refers to the ability of an ML model trained on finite samples to still be able to perform well on new test data drawn from the same distribution. It is well known that generalization is extremely hard for high-dimensional data, a phenomenon called the *curse of dimensionality* [23]. Representation learning is a way of doing dimensionality reduction. By projecting high-dimensional raw data to a much lower-dimensional latent space, we hope to reduce the sample complexity. Indeed, foundation models can facilitate few-shot learning, where very few samples for a downstream task suffice.

Transferability refers to the ability of an ML model trained on data distribution P to still be able to perform well on another data distribution Q. Transferability has always been at the center of deep learning. In the early years following the deep learning boom in 2012, neural networks pretrained on ImageNet [19] were the go-to base models for computer vision, because people found that these models still performed well on data sets other than ImageNet. Transferability is why a foundation model pretrained on a generic data set can be applied to specific downstream tasks.

This proposal shall summarize the work I have done so far, and outline the directions I intend to explore towards my thesis defense. This proposal will not contain any proofs. On a high level, the work I have done can be summarized as:

- Generalization: I propose a new framework for studying representation learning, which can explain why bigger models can learn better representations than smaller ones in practice. The framework views representation learning as extracting basis functions of a functional space $\mathcal{F}$, which is defined with our prior knowledge about the downstream task, rather than a big model.
- Transferability: I focus on the subpopulation shift problem, where the test distribution is absolutely continuous to the train distribution. I reveal several problems

of existing methods, including the surprising negative result that many methods based on sample reweighting cannot lead to transferable models. Then, I propose new algorithms to solve these problems.

I plan to apply representation learning methods to practical applications, especially on tabular data. The specific directions I intend to explore are the following:

- Multiple sources of prior knowledge: I have already shown that one source of prior knowledge can induce a functional space $\mathcal{F}$, and representation learning extracts basis functions of this space. What if we have multiple sources of prior knowledge? How to combine multiple functional spaces?
- Extracting eigenfunctions of a general kernel: Prior work and my work have shown that contrastive and non-contrastive learning are natural ways of extracting the eigenfunctions of the contrastive kernel. But for a more general function, how can we extract its eigenfunctions?
- New metrics for evaluating learned representations: Currently the most widely used way of evaluating representations is with specific downstream tasks. However, one downstream task can only reflect one aspect of the learned representation, and it is widely observed that some representations are better than others on some tasks, but not on others. Is there a more general metric to compare representations?

2 Preliminaries

Representation learning refers to the process of learning low-dimensional representations of data that concisely encode relevant information useful for building classifiers or other predictors [3]. Let $\mathcal{X}$ be the space of raw data. The goal of representation learning is to learn an encoder $\hat{\Phi} = [\hat{\phi}_1, \cdots, \hat{\phi}_d] : \mathcal{X} \to \mathcal{Z}$, where $\mathcal{Z} = \mathbb{R}^d$ is called the latent space, and $\hat{\Phi}(x)$ is called the embedding of sample x. The task used to learn the encoder is called the *pretraining task*, or the *upstream task*. Once pretrained on a generic unlabeled data set, the encoder can be applied to a variety of specific *downstream tasks*, and each downstream task can have its own data distribution.

The method of pretraining a foundation model with massive unlabeled data is called *self-supervised learning* (*SSL*). In order to construct a training objective without any labels, we need to design an *auxiliary task*. The task is "auxiliary" because it is very unlikely to be similar to any downstream task, but is only used for self-supervision. For instance, when training a BERT [6], we randomly mask the sentences in the training set and ask the model to predict the masked tokens. Clearly, this auxiliary task is not similar to most downstream tasks, such as question answering, translation, summarizing, etc.

The downstream task is a standard ML task, where we need to learn a predictor $\hat{f}$ that approximates an unknown ground-truth labeling function $f^* : \mathcal{X} \to \mathcal{Y}$, with label space $\mathcal{Y}$. The predictor is evaluated with the expected risk $\mathcal{R}_Q(\hat{f}, f^*) = \mathbb{E}_{X\sim Q}[\ell(\hat{f}(X), f^*(X))]$, where Q is the data distribution of this downstream task, and $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is a loss function. In the first part of this thesis, we will assume Q to be equal to the pretraining data distribution $P_{\mathcal{X}}$. In the second part where we study transferability, $Q \neq P_{\mathcal{X}}$.

There are a number of ways to learn the predictor $\hat{f} : \mathcal{X} \to \mathcal{Y}$ based on the encoder $\hat{\Phi} : \mathcal{X} \to \mathcal{Z}$. This thesis will mainly focus on *linear probing*, which fits a linear model on top of $\hat{\Phi}$; that is, $\hat{f}(x) = W\hat{\Phi}(x)$ for some W.

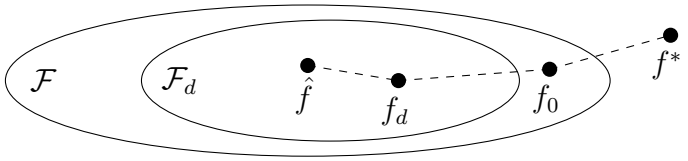

Figure 1: Illustration of the theoretical framework of understanding representation learning. A pretrained d-dimensional representation spans a d-dimensional subspace $\mathcal{F}_d$ of the functional space $\mathcal{F}$. f^* is the ground truth function, f_0 is the projection of f^* onto $\mathcal{F}$, and f_d is the projection of f_0 onto f_d. $\hat{f}$ is the downstream predictor.

3 Theoretical Framework for Understanding Generalization of Big Models in Representation Learning

The first part of the thesis will propose a new framework for studying representation learning. The framework is based on two main arguments:

(a) Representation learning is extracting basis functions of a functional space $\mathcal{F}$.

(b) The functional space $\mathcal{F}$ should be defined with prior knowledge instead of a model.

Denote the encoder by $\Phi = [\phi_1, \cdots, \phi_d]$, where $\phi_i : \mathcal{X} \to \mathbb{R}$. Suppose by prior knowledge, we know that all ϕ_i belong to a specific functional space $\mathcal{F}$, which is essentially the hypothesis space. If $\phi_1, \cdots, \phi_d \in \mathcal{F}$, then they will span a d-dimensional linear subspace of $\mathcal{F}$, denoted by $\mathcal{F}_d$. Therefore, upstream pretraining can be viewed as learning this subspace $\mathcal{F}_d$, as illustrated in Figure 1. At downstream, using a linear probe is equivalent to finding $\hat{f}$ within $\mathcal{F}_d$.

With this machinery, how can we bound the prediction error of $\hat{f}$? Let us assume for now that the error is measured by the squared L^2 distance, that is $\mathcal{R}(\hat{f}, f^*) = \left\| \hat{f} - f^* \right\|_2^2$. Let f_0 be the projection of f^* onto $\mathcal{F}$, and let f_d be the projection of f^* (or f_0) onto $\mathcal{F}_d$. Apparently, f_d is the best predictor of f^* in $\mathcal{F}_d$. However, with only finite samples, we might not find f_d, so there will be a gap between $\hat{f}$ and f_d. Figure 1 illustrates the above analysis, which produces the following decomposition:

$$\mathcal{R}(\hat{f}, f^*) = \left\| \hat{f} - f^* \right\|_2^2 = \|f_0 - f^*\|_2^2 + \|f_d - f_0\|_2^2 + \left\| \hat{f} - f_d \right\|_2^2. \tag{1}$$

The sum of the first two terms, equal to $\|f_d - f^*\|_2^2$, measures the risk of the best predictor in $\mathcal{F}_d$, and is called the *approximation error*. The last term measures how close we can get to this best predictor with finite samples, and is called the *estimation error*.

Note that f_0 depends on f^* and $\mathcal{F}$ (which comes from our prior knowledge), neither of which we can control, so $\|f_0 - f^*\|_2^2$ is an overhead term we have no power over. We are going to bound the other two terms in our generalization results. From this decomposition, we can also see that the representation dimension d controls the following trade-off: A larger d makes the second term smaller because $\mathcal{F}_d$ can cover more parts of $\mathcal{F}$, but also makes the third term larger as it increases the complexity of $\mathcal{F}_d$.

Finally, the functional space $\mathcal{F}$ is defined with our prior knowledge on the downstream task instead of a big model. The complexity of a big model is astronomical, so defining $\mathcal{F}$ with a big model could lead to vacuous and even misleading results. The main type of prior knowledge of similarity we are going to consider is similarity. This could be the similarity between two samples, a sample and its corrupted version, or two corrupted samples.

3.1 Induced RKHS and Soft Invariance

We study a large class of popular self-supervised learning methods based on data augmentation. Two popular self-supervised learning algorithms based on data augmentation are multi-view learning and reconstruction tasks. In multi-view learning, for each sample X, we randomly augment it twice to A and A^+, and call them two *views* of sample X. We enforce the encoder to give similar embeddings to A and A^+. In reconstruction, we augment $X \to A$, and trains an encoder and a decoder by asking them to reconstruct X from A. Let $\mathcal{X}$ be the input space, and $\mathcal{A}$ be the augmented space that contains augmentations of the original samples. Let $P_{\mathcal{X}}$ be the data distribution.

We want to understand why big models can learn generalizable representations, while classical learning theory says that big models can overfit easily. The key is what function class we choose to use. Classical learning theory chooses the function class to be the one induced by the model, and its complexity naturally depends on the size of the model. So for big models, these generalization bounds become vacuous. On the other hand, we define the function class with the data augmentation used to pretrain the encoder, and this function class is independent of the model size and architecture. More specifically, we view the big model as an algorithmic model instead of a data model [5]. The optimal representation to be learned is determined by the augmentation, and the role of the big model is to approximate that representation. The bigger the model is, the easier it can approximate the ideal representation.

The data augmentation incorporates the following prior knowledge: Two views augmented from the same sample should be similar. More formally, we assume g satisfies

$$\frac{1}{2}\mathbb{E}_{X\sim P_{\mathcal{X}}}\mathbb{E}_{A,A^+\sim p(\cdot|X)}\left[(g(A)-g(A^+))^2\right] \le \epsilon\|g\|^2_{P_{\mathcal{A}}}, \quad \text{for some } \epsilon \in (0,1). \tag{2}$$

This is a *soft invariance* condition, which says that on average, $g(A) \approx g(A^+)$. Let the ground truth function be f^*. Note that f^* is a function on $\mathcal{X}$, while g is a function on $\mathcal{A}$. Thus, we assume that there exists g^* that satisfies Eqn. (2), and

$$f^*(x) = \int g^*(a)p(a|x)da, \tag{3}$$

where $p(a|x)$ is given by the data augmentation. The function class of interest is the set of all f^* that satisfies the above condition. We show that this function class can be characterized by an induced RKHS. Define the following kernel:

$$K_X(x_1,x_2) = \int \frac{p(a|x_1)p(a|x_2)}{P_{\mathcal{A}}(a)}da, \tag{4}$$

where $P_{\mathcal{A}}(a) = \int p(a|x)dP_{\mathcal{X}}(x)$. Assume that this is a Mercer kernel. Let $\mathcal{H}_\Gamma$ be the RKHS associated with this kernel. Then, we can show that the above condition of f^* is equivalent to the following *isometry property*:

$$(1-\epsilon)\|f^*\|^2_{\mathcal{H}_\Gamma} \le \|f^*\|^2_{P_{\mathcal{X}}} \le \|f^*\|^2_{\mathcal{H}_\Gamma}. \tag{5}$$

Let the eigenvalues of $\mathcal{H}_\Gamma$ be $\lambda_1 \ge \lambda_2 \ge \cdots \ge 0$, with eigenfunctions $\phi_1, \phi_2, \cdots$ that form an orthonormal basis (ONB) of the L^2 function space *w.r.t.* $P_{\mathcal{X}}$. We can show that $\lambda_1 \le 1$. Thus, this isometry property implies that f^* can be roughly approximated by the top few eigenfunctions of $\mathcal{H}_\Gamma$. In what follows, we provide two sets of generalization results. The first one holds for an arbitrary encoder, and the second one considers the near-optimal d-dimensional encoder.

3.2 Generalization Bound for an Arbitrary Encoder

We focus on the least-squares regression problem, where the loss is the mean squared error (MSE). Let $\|\cdot\|_{P_\mathcal{X}}$ be the L^2 norm. The problem is formally stated as follows:

Problem. *Given unlabeled samples $\tilde{x}_1, \cdots, \tilde{x}_m$ and labeled samples $x_1, \cdots, x_n$ i.i.d. sampled from $P_\mathcal{X}$, and labels $y_k = f^*(x_k) + \nu_k$ for $k \in [n]$ and random noise ν_k, find a predictor $\hat{f} \in L^2(P_\mathcal{X})$ with a low prediction error* $\mathrm{err}(\hat{f}, f^*) := \|\hat{f} - f^*\|_{P_\mathcal{X}}^2 = \mathbb{E}_{P_\mathcal{X}}[(\hat{f}(X) - f^*(X))^2]$.

We assume that the scale of f^* is bounded as $\|f^*\|_{P_\mathcal{X}} \le B$. Then, by Eqn. (5), we have $\|f^*\|_{\mathcal{H}_\Gamma} \le \frac{B}{\sqrt{1-\epsilon}}$. Thus, we consider the following linear probe predictor:

Definition 1. *The final predictor is the **nonparametric least-squares estimate** defined as*

$$\hat{f} := \operatorname*{arg\,min}_{f: f = \boldsymbol{w}^\top \hat{\Phi}, \|f\|_{\mathcal{H}_\Gamma} \le \frac{B}{\sqrt{1-\epsilon}}} \left\{ \frac{1}{n} \sum_{k=1}^{n} (y_k - f(x_k))^2 \right\}.$$

We next introduce two key ingredients crucial to our analyses. The first ingredient is what we term the *augmentation complexity*, which takes the role of the model complexity.

Definition 2. *Define the **augmentation complexity** as $\kappa := \|K_X\|_\infty^{1/2}$, that is for $P_\mathcal{X}$-almost all x, it holds that*

$$K_X(x, x) = \sum_i \lambda_i \phi_i(x)^2 = \int \frac{p(a|x)^2}{P_\mathcal{A}(a)} da \le \kappa^2.$$

Our second ingredient is the *trace gap* that captures the quality of the encoder. It is based on the notion of the *ratio trace*. We assume that $\hat{\phi}_1, \cdots, \hat{\phi}_d$ are linearly independent.

Definition 3. *Define covariance matrices $\boldsymbol{F}, \boldsymbol{G}$ as $\boldsymbol{F}(i,j) = \langle \hat{\phi}_i, \hat{\phi}_j \rangle_{P_\mathcal{X}} = \langle \Gamma^* \hat{\psi}_i, \Gamma^* \hat{\psi}_j \rangle_{P_\mathcal{X}}$ and $\boldsymbol{G}(i,j) = \langle \hat{\psi}_i, \hat{\psi}_j \rangle_{P_\mathcal{A}}$. Then, the **ratio trace** is defined as* $\mathrm{Tr}(\boldsymbol{G}^{-1}\boldsymbol{F})$.

Ratio trace is a classical quantity in linear discriminant analysis (LDA) [24]. The max ratio trace of a d-dimensional encoder is $\lambda_1 + \cdots + \lambda_d$. Denote $S_\lambda(d) = \lambda_1 + \cdots + \lambda_d$.

Definition 4. *Define the **trace gap** of encoder $\hat{\Phi}$, with linearly independent $\hat{\phi}_1, \cdots, \hat{\phi}_d$, as*

$$\tau^2 := \inf_{d' \le d} \inf_{h_1, \cdots, h_{d'}} \left\{ S_\lambda(d'+1) - \mathrm{Tr}(\boldsymbol{G}_h^{-1} \boldsymbol{F}_h) \;\middle|\; h_i = \boldsymbol{w}_i^\top \hat{\Phi} \right\},$$

where $\tau \ge 0$, $\boldsymbol{G}_h = (\langle h_i, h_j \rangle_{P_\mathcal{A}})_{i,j \in [d']}$, and $\boldsymbol{F}_h = (\langle \Gamma^* h_i, \Gamma^* h_j \rangle_{P_\mathcal{X}})_{i,j \in [d']}$.

We now state our first main result, assuming the noise to be *i.i.d.* Gaussian.

Theorem 5. *Let $\nu_1, \cdots, \nu_n$ be i.i.d. $\mathcal{N}(0, \sigma^2)$ variates. Let $\hat{f}$ be given by Definition 1. If $\hat{\Psi}$ has d dimensions and $\tau < 1$ (d can be ∞), then there are universal constants c_0, c_1, c_2 such that with probability at least $1 - c_1 \exp\left(-\frac{c_2 \sqrt{2n S_\lambda(d+1)}}{\kappa}\right) - \exp\left(-\sqrt{\frac{2n\kappa^2 B^2}{1-\epsilon}}\right)$, there is*

$$\|\hat{f} - f^*\|_{P_\mathcal{X}}^2 \le \frac{9\tau^2(\tau + \epsilon) B^2}{(1-\tau^2)(1-\epsilon)} + \frac{c_0 \kappa (B^2 + \sigma B)}{1 - \epsilon} \sqrt{\frac{S_\lambda(d+1)}{n}} \qquad \text{for all } f^* \in \mathcal{F}^B(\Gamma; \epsilon).$$

This bound consists of two terms. The first term bounds the approximation error entailed by the limited capacity of the d-dimensional encoder, and the second term bounds the estimation error entailed by finite labeled samples, as we discussed in Eqn. (1). The number of unlabeled samples m does not appear in this bound, since this bound is for an arbitrary encoder that can be trained on any amount of data. In a typical pretraining setup, the trace gap τ^2 in the first term depends on m. Moreover, the first term might not vanish as $m, n \to \infty$, since τ^2 is lower bounded by λ_{d+1}; for instance, if d is too small, then $\mathcal{F}_d$ cannot cover the entirety of $\mathcal{F}$ and the approximation error can never be zero.

3.3 Generalization Results for the Near-Optimal Encoder

Let $\mathcal{F}$ be the set of all f^* that satisfies Eqn. (5), and $\|f^*\|_{P_\mathcal{X}} \le B$. We first show that the optimal top-d encoder consists of the top-d eigenfunctions of $\mathcal{H}_\Gamma$. Here "optimal" means that the encoder minimizes the worst-case approximation error over $\mathcal{F}$, defined as

$$\text{err}(\hat{\Phi}; \mathcal{F}) := \sup_{f\in\mathcal{F}} \min_{\boldsymbol{w}\in\mathbb{R}^d} \text{err}(\boldsymbol{w}^\top \hat{\Phi}, f) = \sup_{f\in\mathcal{F}} \min_{\boldsymbol{w}\in\mathbb{R}^d} \|\boldsymbol{w}^\top \hat{\Phi} - f\|_{P_\mathcal{X}}^2.$$

Proposition 6. *Suppose* $\frac{\lambda_{d+1}}{1-\lambda_{d+1}} \frac{\epsilon}{1-\epsilon} \le \frac{1}{2}$. *For any* $\hat{\Phi} = [\hat{\phi}_1, \cdots, \hat{\phi}_d]$ *where* $\hat{\phi}_i \in L^2(P_\mathcal{X})$, *there is*

$$\text{err}(\hat{\Phi}; \mathcal{F}) \ge \frac{\lambda_{d+1}}{1-\lambda_{d+1}} \frac{\epsilon}{1-\epsilon} B^2.$$

To attain equality, it is sufficient for $\hat{\mathcal{H}}_{\hat{\Phi}}$, *the linear space spanned by* $\hat{\Phi}$, *to contain all of* $\phi_1, \cdots, \phi_d$; *and this is also necessary if* $\lambda_{d+1} < \lambda_d$.

However, we cannot obtain this optimal encoder with only finite samples, because the eigenfunctions of $\mathcal{H}_\Gamma$ depend on the real data distribution $P_\mathcal{X}$ which we have no access to. Given samples $\tilde{x}_1, \cdots, \tilde{x}_m$, what we can access is the empirical distribution $\hat{P}_\mathcal{X}$, defined as the uniform distribution over these m samples. We can also define $\hat{P}_\mathcal{A}$ as $\hat{P}_\mathcal{A}(a) = \frac{1}{m}\sum_{k=1}^m p(a|\tilde{x}_k)$. With finite samples, Γ^* will still be the same (because $p(a|x)$ does not change for any a, x), but Γ will be the following empirical version:

$$(\bar{\Gamma} f)(a) = \frac{1}{m}\sum_{k=1}^m \frac{f(\tilde{x}_m) p(a|\tilde{x}_m)}{\hat{P}_\mathcal{A}(a)}.$$

Let the eigenvalues and eigenfunctions of $\Gamma^*\bar{\Gamma}$ be $\{(\bar{\lambda}_i, \bar{\phi}_i)\}$. We define the near-optimal d-dimensional encoder as $[\bar{\phi}_1, \cdots, \bar{\phi}_d]$. Then, we can prove the following result:

Theorem 7. *Let* $\boldsymbol{F}, \boldsymbol{G}$ *be the covariance matrices in Definition 3. Let* $\gamma_{\boldsymbol{G}} := \lambda_{\max}(\boldsymbol{G})/\lambda_{\min}(\boldsymbol{G})$ *be the condition number of* $\boldsymbol{G}$, *where* $\lambda_{\max}(\boldsymbol{G})$, $\lambda_{\min}(\boldsymbol{G})$ *are the largest and smallest eigenvalues of* $\boldsymbol{G}$. *For any* $\delta > 0$, *it holds for the near-optimal encoder with probability at least* $1-\delta$ *that*

$$\tau^2 \le S_\lambda(d+1) - \text{Tr}(\boldsymbol{G}^{-1}\boldsymbol{F}) \le \lambda_{d+1} + \left(2 + \sqrt{2\log\frac{2}{\delta}}\right) \frac{(\lambda_d^{-1} + \bar{\lambda}_d^{-1}\gamma_{\boldsymbol{G}}^{1/2} + 2)\kappa^2}{\sqrt{m}} d.$$

3.4 Variational Objectives Extracting Top-d Eigenfunctions

We have shown that the optimal encoder consists of the top-d eigenfunctions. The remaining question is how to extract them. It turns out that we do not need to explicitly compute K_X, which would not be scalable. Instead, there are a couple of pretraining objectives that are (uniquely) optimized by the top-d eigenfunctions. Thus, assuming perfect optimization, optimizing these objectives naturally gives us the top-d eigenfunctions. Here are two examples:

- Spectral contrastive loss [9]: $-2\mathbb{E}\left[\langle \hat{\Psi}(A), \hat{\Psi}(A^+)\rangle\right] + \mathbb{E}\left[\langle \hat{\Psi}(A), \hat{\Psi}(A^-)\rangle^2\right]$.
- Barlow twins [28]: $\min\ \mathbb{E}\left[\left\|\hat{\Psi}(A)\right\|_2^2\right]$ s.t. $\text{Cov}[\hat{\Psi}] = \boldsymbol{I}$.

The spectral contrastive loss can be applied to multi-modal models like CLIP [18]. Let X be the image and A be the text. Let $\hat{\Phi}$ be the image encoder and $\hat{\Psi}$ be the text encoder. Define the spectral CLIP loss as $\mathcal{L}(\hat{\Phi}, \hat{\Psi}) = -2\mathbb{E}[\langle \hat{\Phi}(X), \hat{\Psi}(A^+)\rangle] + \mathbb{E}[\langle \hat{\Phi}(X), \hat{\Psi}(A^-)\rangle^2]$. We can show that the $\hat{\Phi}$ that minimizes this loss is also the top-d eigenfunctions.

3.5 Spectrally Transformed Kernels

We can extend the top-d eigenfunctions to a more general formulation called the spectrally transformed kernels. Let $k(x, x')$ be a kernel that encodes inter-sample similarity. Let its eigen-decomposition be $k(x, x') = \sum_i \lambda_i \phi_i(x)\phi_i(x')$. Its spectrally transformed kernel (STK) is given by

$$k_s(x, x') = \sum_{i:\lambda_i>0} s(\lambda_i)\phi_i(x)\phi_i(x'), \tag{6}$$

for a monotonically non-decreasing transformation function $s : [0, +\infty) \to [0, +\infty)$.

When $s(\lambda) = \lambda^p$ for $p \geq 1$, we denote $k_s = k^p$, and it is easy to show that $k^{p+1}(x, x') = \int k^p(x, z)k(z, x')dP_{\mathcal{X}}(z)$. $\{k^p\}_{p\geq 1}$ are called diffusion kernels. Moreover, when $s(\lambda_i) = \lambda_i \mathbf{1}_{i\leq d}$ is the top-d truncation function, k_s only contains the top-d eigenfunctions, and recovers the previous section. Doing regression with an STK, which we call STKR, is a very general and principled way of learning with labeled and unlabeled data.

Suppose we have full access to k and s, and $s(\lambda) = \sum_i \pi_i s^i$ is a polynomial. In this case, however, we still cannot compute k_s with finite samples, because we cannot access ϕ_i. To implement STKR, we need to construct a new kernel $\hat{k}_s$ to approximate k_s, and this is where the unlabeled data becomes useful. More specifically, for each positive integer p, we will construct $\hat{k}^p$ to approximate k^p, and then $\hat{k}_s = \sum_p \pi_p \hat{k}^p$.

Consider how we can approximate k^2. Recall that $k^2(x, x') = \int k(x, z)k(x', z)dP_{\mathcal{X}}(z)$. A natural method is Monte-Carlo approximation, where we replace the integral with the average over the finite samples. Thus, $\hat{k}^2(x, x') = \frac{1}{n+m}\sum_{i=1}^{n+m} k(x, x_i)k(x', x_i)$. We can then construct $\hat{k}^p$ similarly. Define $\boldsymbol{v}_k(x) \in \mathbb{R}^{n+m}$ as $\boldsymbol{v}_k(x)[i] = k(x, x_i)$ for $i \in [n + m]$. Then, define $\hat{k}^1 = k$; and for $p \geq 2$, define $\hat{k}^p(x, x') = \frac{\boldsymbol{v}_k(x)^\top \boldsymbol{G}_k^{p-2} \boldsymbol{v}_k(x')}{(n+m)^{p-1}}$. With full access to k, one can compute $\hat{k}^p$ for any p. Note that the unlabeled data is leveraged when we compute $\hat{k}^p$, because $\boldsymbol{G}_k$ is the Gram matrix over both labeled and unlabeled samples.

When s is not a polynomial but the top-d truncation function, we can approximate k_s using kernel PCA. And then, we can prove statistical bounds for STKR with polynomial or top-d truncation transformation similar to the previous section. For application, we can apply STKR to node classification tasks, which we omit here.

4 Learning Transferable Models under Subpopulation Shift: Challenges and Solutions

It is well known that standard empirical risk minimization (ERM) can empirically achieve high test performance with big models on a variety of tasks. However, its performance could downgrade when the train and test data distributions are different, which is a common scenario in practice. If a model trained on distribution P can still work well on another distribution Q, then we say that this model has *out-of-distribution* (*OOD*) *generalization*. Particularly, here we focus on the *subpopulation shift* problem where $Q \ll P$, which means that $P(A) = 0$ always implies $Q(A) = 0$. A foundation model is pretrained on a very generic data set such as Wikipedia, so it is natural to assume that the upstream data set contains all downstream data.

OOD generalization is a very important topic, and various approaches to learning models robust against subpopulation shift have been proposed. Most of these approaches

are based on reweighting, which minimizes a weighted average of the model's losses on the training samples. Despite their popularity, these methods have two major challenges:

- Sensitivity to outliers: The idea of reweighting is to give more weights to "harder" samples, that is samples on which the model has higher losses. However, real data sets have outliers, and intuitively the model has higher losses on these outliers. Thus, reweighting gives more weights to outliers, which downgrades the performance.
- Overfitting: [20] empirically showed that reweighting requires a much larger regularization than ERM. Moreover, [8] conducted a large-scale empirical study and found that no reweighting method is significantly better than ERM. Why is reweighting more prone to overfitting, and is it better than ERM?

4.1 Generalized Reweighting (GRW)

Let the input space be $\mathcal{X} \subseteq \mathbb{R}^d$ and the output space be $\mathcal{Y} \subseteq \mathbb{R}$. For simplicity we assume $\mathcal{Y}$ to be one-dimensional, though our results can be easily extended to the multi-class scenario. We assume that every $\boldsymbol{x} \in \mathcal{X}$ satisfies $\|\boldsymbol{x}\|_2 \leq 1$. We have a training set $\{\boldsymbol{z}_i = (\boldsymbol{x}_i, y_i)\}_{i=1}^n$ *i.i.d.* sampled from an underlying distribution P over $\mathcal{X} \times \mathcal{Y}$. Denote $\boldsymbol{X} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n) \in \mathbb{R}^{d \times n}$, and $\boldsymbol{Y} = (y_1, \cdots, y_n) \in \mathbb{R}^n$. For any function $g : \mathcal{X} \mapsto \mathbb{R}^m$, we overload notation and denote $g(\boldsymbol{X}) = (g(\boldsymbol{x}_1), \cdots, g(\boldsymbol{x}_n)) \in \mathbb{R}^{m \times n}$.

ERM trains a model by minimizing its *expected risk* $\mathcal{R}(f; P) = \mathbb{E}_{\boldsymbol{z} \sim P}[\ell(f(\boldsymbol{x}), y)]$ via minimizing the *empirical risk* $\hat{\mathcal{R}}(f) = \frac{1}{n}\sum_{i=1}^n \ell(f(\boldsymbol{x}_i), y_i)$, where $\ell : \mathcal{Y} \times \mathcal{Y} \to [0, 1]$ is the loss function. On the other hand, at iteration t, generalized reweighting (GRW) minimizes the weighted empirical risk given by

$$\hat{\mathcal{R}}_{\boldsymbol{q}^{(t)}}(f) = \sum_{i=1}^n q_i^{(t)} \ell(f(\boldsymbol{x}_i), y_i), \tag{7}$$

where $\boldsymbol{q}^{(t)} = (q_1^{(t)}, \cdots, q_n^{(t)})$ is the sample weight vector, such that $q_1^{(t)} + \cdots + q_n^{(t)} = 1$. It is called "generalized" because in a standard reweighting algorithm, the sample weight $q_i^{(t)}$ does not change with time t; we call this static GRW. If $q_i^{(t)}$ can change with t, then we call it dynamic GRW. Note that ERM is a special case of static GRW.

A classical static GRW method is importance weighting (IW) [22]. Suppose the upstream data set contains K downstream domains of interest. Denote their training subsets by $\mathcal{D}_1, \cdots, \mathcal{D}_K$, and let n_k be the size of $\mathcal{D}_k$. The problem of ERM is that it will have a low performance on very small domains. To solve this, IW sets $q_i^{(t)} \equiv q_i = (Kn_k)^{-1}$ for all $\boldsymbol{z}_i \in \mathcal{D}_k$, so that the weighted empirical risk has equal weights on all domains.

The most popular family of dynamic GRW methods is *distributionally robust optimization (DRO)*. In DRO, we use the train distribution P to specify an uncertain set $\mathcal{U}(P)$, which is a set of distributions we believe the test distribution Q should belong to. One way to define the uncertainty set is to use a divergence function $D(Q \parallel P)$. For subpopulation shift, we can define $\mathcal{U}(P) = \{Q \mid Q \ll P, D(Q \parallel P) \leq \rho\}$ for some $\rho > 0$. Then, DRO minimizes the *worst-case expected risk* defined as

$$\mathcal{R}_{D,\rho}(f; P) = \sup_Q \{\mathbb{E}_Q[\ell(f(\boldsymbol{x}), y)] \mid Q \ll P, D(Q \parallel P) \leq \rho\}.$$

Moreover, in the domain-aware setting where we know the domains $\mathcal{D}_1, \cdots, \mathcal{D}_K$, then we can define the uncertainty set to be $\{P_1, \cdots, P_K\}$, where $P_k(\boldsymbol{z}) = P(\boldsymbol{z} | \boldsymbol{z} \in \mathcal{D}_k)$.

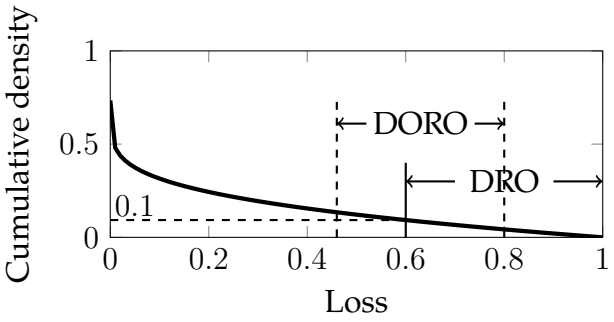

Figure 2: Comparison between DRO and DORO.

Then, the worse-case expected risk essentially becomes the *worst-group risk*:

$$\mathcal{R}_{\max}(f; P) = \max_{k=1,\cdots,K} \mathcal{R}(f; P_k) = \max_{k=1,\cdots,K} \mathbb{E}_{\boldsymbol{z}\sim P}[\ell(f(\boldsymbol{x}), y) | z \in \mathcal{D}_k]. \tag{8}$$

Group DRO [20] is a popular method of minimizing the worst-group risk. The method uses bi-level optimization, where the lower level updates the model to minimize the expected risk over Q, and the upper level updates Q to maximize the expected risk. Denote the empirical risk over group k by $\hat{\mathcal{R}}_k(f)$, and the model at time t by $f^{(t)}$. For all $k \in [K]$, group DRO iteratively sets $q_i^{(t)} = g_k^{(t)}/n_k$ for all $\boldsymbol{z}_i \in \mathcal{D}_k$, where $g_k^{(t)}$ is the group weight that is updated as

$$g_k^{(t)} \propto g_k^{(t-1)} \exp\left[\nu \hat{\mathcal{R}}_k\big(f^{(t-1)}\big)\right]$$

for some $\nu > 0$. The group weights are normalized so that $q_1^{(t)} + \cdots + q_n^{(t)} = 1$. [20] showed (in their Proposition 2) that for convex settings, the Group DRO risk of iterates converges to the global minimum with the rate $O(t^{-1/2})$ if ν is sufficiently small.

In a nutshell, the idea of GRW is to give larger weights to domains that are "harder" to learn, such as small domains, or domains where the current model has a high risk.

4.2 Mitigating Sensitivity to Outliers

Consider GRW algorithms that assign larger weights to samples where the model has higher loss. One family of such algorithms is distributionally robust optimization (DRO), which defines an uncertainty set $\mathcal{U}(P) = \{Q \mid D(Q \parallel P) \le \rho\}$ for some divergence function D and some $\rho > 0$. Then, DRO minimizes the *worst-case expected risk* defined as $\mathcal{R}_{D,\rho}(h; P) = \sup_Q \big\{\mathbb{E}_{(X,Y)\sim Q}[\ell(h(X), Y)] \;\big|\; D(Q \parallel P) \le \rho\big\}$. We also call this the DRO risk. The uncertainty set contains distributions that are close to the training distribution, and DRO trains the model on the worst distribution in the uncertainty set.

The problem is that real data sets contain outliers, and intuitively models have higher loss on outliers. As a result, DRO pays more attention to outliers during training, causing instability and a performance drop. Prior work [10, 12, 29] has already observed that DRO is highly sensitive to outliers. In my work, I also use experiments to showcase that DRO methods are very sensitive to outliers in real data sets.

To this end, we propose a method called DORO, which stands for Distributionally and Outliers Robust Optimization. The idea of DORO is to ignore the samples where the model has the highest loss, because these samples are potential outliers. Figure 2 compares between DRO and DORO. DRO gives more weights to the samples with the

highest loss, as labeled by "DRO", but there are lots of outliers within these samples. On the other hand, DORO will ignore a small portion of the worst samples (for example 10%) and then apply DRO, so it essentially gives more weights to samples in the middle, as labeled by "DORO".

Now let us formulate DORO. Consider the Huber's ϵ-contamination model $P = (1-\epsilon)P' + \epsilon\tilde{P}'$, where P is the observed training distribution, P' is the "clean" training distribution without outliers, and $\tilde{P}'$ is an arbitrary distribution. DORO trains the model on the "easiest" P' that satisfies this model and minimizes the risk. Specifically, the expected ϵ-DORO risk is defined as

$$\mathcal{R}_{D,\rho,\epsilon}(h;P) = \inf_{P'} \left\{ \mathcal{R}_{D,\rho}(h;P) \;\middle|\; \exists\, \tilde{P}' \text{ s.t. } P = (1-\epsilon)P' + \epsilon\tilde{P}' \right\}. \tag{9}$$

Here ϵ is a hyperparameter, and we only need to make sure that it is not less than the real noise level. It is easy to show that this risk corresponds to Figure 2.

One might ask what if the model has low loss on some outliers, and high loss on some inliers? Will DORO still work in this case? The answer is positive, and we prove two theoretical results:

- The minimizer of the DORO risk on the contaminated distribution achieves a close-to-minimum DRO risk on the clean distribution assuming the Huber's contamination model. We also show that this bound is information-theoretically optimal.
- The DORO risk is a surrogate loss of $\mathcal{R}_{\max}$ defined in Eqn. (8). Specifically, $\mathcal{R}_{\max}$ on the clean distribution is upper bounded by the DORO risk on the contaminated distribution multiplied by a constant factor.

We test the proposed DORO algorithm on real data sets, including COMPAS, CelebA and CivilComments-Wilds. On all data sets, we find that DORO can lead to more stable training and better performance than DRO and ERM.

4.3 Is GRW Better than ERM?

Prior work empirically showed that GRW needs a much larger regularization than ERM, and is not better than ERM on a variety of tasks. In this part, we focus on a comparison between GRW and ERM. We will show for both regression and classification tasks, and for both linear models and wide neural networks that GRW is not necessarily better than ERM. This negative result is shown from an optimization perspective. Specifically, we will show that when starting from the same initialization, GRW and ERM will converge to two very close points, and thus GRW cannot be better.

Let us start from regression with the squared loss $\ell(\hat{y}, y) = \frac{1}{2}(\hat{y}-y)^2$. We first demonstrate this negative result on linear models, thereby providing a key proof intuition. Let the linear model be denoted by $f(\boldsymbol{x}) = \langle \theta, \boldsymbol{x} \rangle$, where $\theta \in \mathbb{R}^d$. We consider the overparameterized setting where $d > n$. The weight update rule of GRW under gradient descent (GD) is $\theta^{(t+1)} = \theta^{(t)} - \eta \sum_{i=1}^n q_i^{(t)} (f^{(t)}(\boldsymbol{x}_i) - y_i)\boldsymbol{x}_i$. Therefore, $\theta^{(t+1)} - \theta^{(t)}$ is always a linear combination of $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n$. Suppose that $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n$ are linearly independent. Then, if $\theta^{(t)} \to \theta^*$, then $\theta^* - \theta^{(0)} \in \text{span}\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n\}$.

Meanwhile, since optimizing the squared loss for a linear model is a convex problem, given that $q_i^{(t)} \to q_i$ for some q_i as $t \to \infty$, we know that $\theta^{(t)}$ will always converge to some θ^*. And if $q_i > 0$, then this θ^* should achieve zero squared loss, which means that it is an interpolator such that $\langle \theta^*, \boldsymbol{x}_i \rangle = y_i$ for all $i \in [n]$.

Now here is the key. We know that $\theta^* - \theta^{(0)} \in \text{span}\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n\}$, which is an n-dimensional linear space. We also know that $\langle \theta^*, \boldsymbol{x}_i \rangle = y_i$ for all $i \in [n]$, which are n linear equations. By Cramer's rule, there is exactly one θ^* that satisfies both conditions. This unique θ^* does not depend on the sample weights $q_i^{(t)}$. Also note that ERM is a special case of GRW, with $q_i^{(t)} \equiv 1/n$. Thus, we have essentially proved for this setting that GRW and ERM will always converge to the same model, given that there is no regularization and no early stopping. We can extend the above result to wide neural networks in the NTK regime [13], since these networks can be approximated by their linearized version.

Moreover, we also study the effect of L^2 regularization, with which the GRW learning objective becomes

$$\hat{\mathcal{R}}^{\mu}_{\boldsymbol{q}^{(t)}}(f) = \sum_{i=1}^{n} q_i^{(t)} \ell(f(\boldsymbol{x}_i), y_i) + \frac{\mu}{2} \left\| \theta - \theta^{(0)} \right\|_2^2. \tag{10}$$

Adding regularization does make a difference regardless of how big μ is. For instance, for the linear model case we discussed earlier, with regularization the problem is still convex, so θ will converge to the global minimum θ^* that satisfies $\nabla_\theta \hat{\mathcal{R}}^{\mu}_{\boldsymbol{q}^{(t)}}(f(\boldsymbol{x}; \theta^*)) = 0$. For static GRW, the solution is $\theta^* = \theta^{(0)} + (\boldsymbol{X}\boldsymbol{Q}\boldsymbol{X}^\top + \mu \boldsymbol{I})^{-1} \boldsymbol{X}\boldsymbol{Q}(\boldsymbol{Y} - f^{(0)}(\boldsymbol{X}))$, where $\boldsymbol{Q} = \text{diag}(q_1, \cdots, q_n)$. Clearly, θ^* depends on the sample weights.

Our argument, however, is that the regularization must be large enough to *significantly lower the training performance*, or the final model would still be close to the unregularized ERM model. Specifically, we prove that if the empirical training risk of the final model can go below ϵ for some $\epsilon > 0$, then for all $\boldsymbol{x}$ such that $\|\boldsymbol{x}\|_2 \le 1$, we have

$$\limsup_{t \to \infty} \left| f^{(t)}_{\text{REG}}(\boldsymbol{x}) - f^{(t)}_{\text{ERM}}(\boldsymbol{x}) \right| = O(d_w^{-1/4} + \sqrt{\epsilon}) \to O(\sqrt{\epsilon}) \qquad \text{as } d_w \to \infty.$$

Here, $f^{(t)}_{\text{REG}}$ is the regularized GRW model, f_{ERM} is the ERM model, and d_w is the width of the network. This means that if ϵ is very small, then $f^{(t)}_{\text{REG}}(\boldsymbol{x}) \approx f_{\text{ERM}}(\boldsymbol{x})$ for all $\boldsymbol{x}$, and thus regularized GRW cannot be better than ERM.

We now move on to classification. For simplicity we focus on binary classification where $\mathcal{Y} = \{+1, -1\}$, but our results can be easily extended to multi-class classification. Let the loss be the logistic loss $\ell(\hat{y}, y) = \log(1 + \exp(-\hat{y}y))$. The big difference here is that *the logistic loss does not have finite minimizers*. The logistic loss converging to zero means that the model weight "explodes" to infinity instead of converging to a finite point.

Let us again provide the key proof intuition by looking at the linear model $f(\boldsymbol{x}) = \langle \theta, \boldsymbol{x} \rangle$. We show that when the training error goes to zero, $\theta^{(t)}$ will converge to the *max-margin classifier* defined as $\hat{\theta}_{\text{MM}} = \arg\max_{\theta: \|\theta\|_2 = 1} \{\min_{i=1,\cdots,n} y_i \cdot \langle \theta, \boldsymbol{x}_i \rangle\}$, as long as $\liminf_{t \to \infty} q_i^{(t)} > 0$. Note that $\hat{\theta}_{\text{MM}}$ only depends on the training samples, and does not depend on either the initial point or the sample weights. Thus, again we have shown that all GRW, including ERM, will converge to the same final model, and thus GRW is no better than ERM.

Similarly, we can extend this negative result to wide neural networks, and regularized logistic loss. Define the max-margin linearized NN as $f_{\text{MM}}(\boldsymbol{x}) = \left\langle \hat{\theta}_{\text{MM}}, \nabla_\theta f^{(0)}(\boldsymbol{x}) \right\rangle$, where $\hat{\theta}_{\text{MM}} = \arg\max_{\|\theta\|_2 = 1} \left\{ \min_{i=1,\cdots,n} y_i \cdot \left\langle \theta, \nabla_\theta f^{(0)}(\boldsymbol{x}_i) \right\rangle \right\}$. Under the same conditions as the previous result, we can show that if the training error can converge to somewhere less than ϵ, then for any $\boldsymbol{x}$ such that $|f_{\text{MM}}(\boldsymbol{x})| = \Omega(\sqrt{-\log 2\epsilon})$, the output of the wide neural network trained with the regularized loss will have the same sign as $f_{\text{MM}}(\boldsymbol{x})$, which means that they predict the same class.

4.4 Enhancing the OOD Generalization

We have shown the sobering result that for both regression and classification, for both linear models and wide neural networks, and for both unregularized and regularized loss that does not significantly lower the training performance, GRW does not have better OOD generalization than ERM. The natural following question is how we can improve this. While this is still an open problem at this moment and we have called upon our community to look into this issue more deeply, here we list three approaches that could help enhance the OOD performance.

The first approach is to create more synthetic data with data augmentation or generative models. We have shown that giving more weights to small domains does not actually help, but instead we can enlarge these small domains with synthetic data. There has been a huge breakthrough in generative models recently, and people have been trying to apply them to OOD generalization [21, 26].

For classification tasks, there are two other approaches. The second approach is to design a new class of algorithms other than GRW, such as *logit adjustment* [17]. We have proved that for classification, all GRW models converge to the max-margin classifier. The idea of logit adjustment is to adjust the margin of each domain, such that smaller domains have larger margins. The way to achieve this is instead of multiplying the sample weights to the loss on the samples, we multiply it to the logits predicted by the model. If we multiply a small number to the logits of one particular domain, then the model will be encouraged to have a large margin on this domain.

The third approach is to stay within the class of GRW algorithms, but to use a new family of loss functions. For instance, [25] showed that by changing an exponentially tailed loss such as the logistic loss to a polynomially-tailed loss such as

$$\ell_{\alpha,\beta}(\hat{y}, y) = \begin{cases} \ell_{\text{left}}(\hat{y}y) & \text{, if } \hat{y}y < \beta; \\ \dfrac{1}{[\hat{y}y - (\beta - 1)]^{\alpha}} & \text{, if } \hat{y}y \geq \beta, \end{cases}$$

where ℓ_{left} is a suitable function that makes this loss monotonic and smooth, then GRW will converge to a different point than ERM. However, being different does not necessarily mean that GRW is actually better than ERM. [25] showed for a toy example that GRW is better, but for a more general case this is still unknown.

5 Proposed Future Work

As summarized above, my prior work studied the generalization and transferability of representation learning especially with big models, figured out some issues and challenges of existing methods, and proposed several new methods. In my last year of PhD, I intend to apply these new methods to practical applications, especially on tabular data. I will investigate how to combine multiple sources of prior knowledge, how to extract eigenfunctions of kernels via variational objectives, and how to evaluate representations.

5.1 Representation Learning on Tabular Data

Despite the great success of deep learning, tabular data remains a bastion it has yet to conquer. Tabular data is the most common type of data in industrial applications. However, at this point, the state-of-the-art method on tabular data is still gradient boosted

decision trees (GBDT), as shown in recent empirical studies [7, 16]. Real-world tabular data has a number of complexities, such as categorical features, missing values, etc. Moreover, the relationship between the response and the predictor variables is not necessarily smooth, in which case neural networks cannot perform as good as decision trees.

On the other hand, using deep models to learn representations for tabular data is a promising direction to explore. Using deep models for prediction might be bad due to the non-smoothness of the underlying pattern, but using them for representation learning could extract useful information that helps improve the performance.

Current progress. My team has already built the infrastructure for large-scale experiments, with over 200 data sets, 5 benchmarks, and a number of baselines. We have implemented contrastive and non-contrastive learning for tabular data, such as SCARF [1], VICReg [2], global contrastive learning [27], etc. We have also implemented some recent deep learning based methods such as TabPFN [11]. At this point, GBDT still achieves the best performances on all benchmarks, but by combining more features learned by neural networks, we are confident that we can achieve further improvement.

5.2 Extracting Eigenfunctions of a General Kernel

Let $k(\cdot, \cdot)$ be a Mercer kernel. My work showed that the top eigenfunctions of k constitute a good representation. The problem is how to extract the eigenfunctions of k in a scalable way. The classical algorithm is kernel PCA, which proceeds by constructing a gigantic Gram matrix of k on all samples, and then computing its eigenvectors. Clearly, kernel PCA is not scalable, and we want a new method that uses big models.

Let us assume that we have cheap access to $k(x, x')$ for all x, x', and k is non-negative. Then, we can use an algorithm similar to non-contrastive learning. For each step:

1. Sample a batch $\{x_1, \cdots, x_m\}$.
2. Compute the degree of x_i for all $i \in [m]$: $D(x_i) = \sum_{j=1}^m k(x_i, x_j)$.
3. Compute the conditional probability $P^+(x_j|x_i) = \tilde{k}(x_i, x_j) = \frac{k(x_i,x_j)}{D(x_i)}$.
4. Sample $x_i^+, x_i^{++} \sim P^+(\cdot|x_i)$ independently.
5. Minimize $\frac{1}{m} \sum_{i=1}^m \left\|\Phi(x_i^+) - \Phi(x_i^{++})\right\|_2^2$, subject to $\mathrm{Cov}_{P(X^+)}(\Phi) = \boldsymbol{I}$.

Note that in the final step, the covariance is computed on $X^+ \sim P^+(\cdot|X)$, where $X \sim P_{\mathcal{X}}$. My prior work showed that the minimizer of this loss is the top-d eigenfunctions. It remains to investigate the optimization behavior of this algorithm, and apply it to some real data sets and see its performance.

There are two other questions that need to be studied. First, what if k is not always non-negative, in which case we cannot define the conditional probability P^+? Second, what if we cannot access $k(x, x')$ for all x, x'?

5.3 Combining Multiple Sources of Prior Knowledge

Let k be a similarity kernel, such as the contrastive kernel. Non-contrastive learning can be formulated as maximizing $\left\langle k, \Phi\Phi^\top \right\rangle$ subject to $\mathrm{Cov}(\Phi) = \boldsymbol{I}$. What if we have multiple kernels $k_1, \cdots, k_r$, where each kernel encodes one piece of prior knowledge?

For instance, there are multiple types of data augmentation the target function should be invariant to, and they induce multiple contrastive kernels.

The most classical way of learning with multiple kernels is using a linear combination $k = \sum_{j=1}^{r} w_j k_j$, where $\boldsymbol{w} = (w_1, \cdots, w_r)$ is on the r-dimensional unit simplex Δ^r. Then, we can maximize $\min_{\boldsymbol{w}} \left\langle \sum_j w_j k_j, \Phi\Phi^\top \right\rangle$, which can be done via a bi-level optimization procedure, similar to an EM algorithm. For each step, we first update Φ just like the single kernel case; then, we update $\boldsymbol{w}$, probably using an exponential update algorithm.

Apart from linear combination, we can also consider kernel convolution. Formally, for two kernels k_1 and k_2, we define $(k_1 \star k_2)(x, x') = \int k_1(x, z) k_2(z, x') dP_{\mathcal{X}}(z)$. This is not symmetric, but we can always consider $k_1 \star k_2 + k_2 \star k_1$. This gives us more types of interaction between the kernels.

5.4 New Ways of Evaluating Representations

At this point, the most widely used way of evaluating an encoder is to test it on a bunch of downstream tasks. Such evaluation, on its own, is innocuous since after all, high downstream performance is what we want. However, the problem emerges when people start to optimize their pretraining algorithm towards the downstream performance. It is widely reported that tricks that work well on some downstream tasks might not work well on others, so it is hard to justify using these tricks to train foundation models when we don't know what the real downstream tasks would be.

Therefore, we need more general ways of evaluating and comparing pretrained representations. [14] discussed several ways of comparing representations, such as CCA and CKA. However, these metrics only compare the similarity between two representations, but when they are not similar, these metrics say nothing about which one is better.

There are some existing principles regarding the quality of a representation. For instance, InfoMax [15] states that a good representation should have high mutual information with the input, and contrastive learning is derived from InfoMax. There are two problems here. First, estimating the mutual information is notoriously hard. Second, when both signal and noise exist in the data, a high mutual information contain both, and it says little about the signal-to-noise ratio of the representation, which is crucial for applications where the data contains lots of noise such as financial data.

In this direction, I plan to investigate InfoMax more deeply. Specifically, I want to know why InfoMax is a good principle, whether it leads to identifiable representations, and how to estimate the mutual information for a pretrained encoder.

References

[1] Dara Bahri, Heinrich Jiang, Yi Tay, and Donald Metzler. Scarf: Self-supervised contrastive learning using random feature corruption. *arXiv preprint arXiv:2106.15147*, 2021. 5.1

[2] Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations*, 2022. 5.1

[3] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013. 2

[4] 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. *arXiv preprint arXiv:2108.07258*, 2021. 1

[5] Leo Breiman. Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical science*, 16(3):199–231, 2001. 3.1

[6] 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)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. 2

[7] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? *Advances in neural information processing systems*, 35:507–520, 2022. 5.1

[8] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. In *International Conference on Learning Representations*, 2021. 4

[9] Jeff Z HaoChen, Colin Wei, Adrien Gaidon, and Tengyu Ma. Provable guarantees for self-supervised deep learning with spectral contrastive loss. *Advances in Neural Information Processing Systems*, 34:5000–5011, 2021. 3.4

[10] Tatsunori Hashimoto, Megha Srivastava, Hongseok Namkoong, and Percy Liang. Fairness without demographics in repeated loss minimization. In Jennifer Dy and Andreas Krause, editors, *International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1929–1938, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. 4.2

[11] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *The Eleventh International Conference on Learning Representations*, 2023. 5.1

[12] Weihua Hu, Gang Niu, Issei Sato, and Masashi Sugiyama. Does distributionally robust supervised learning give robust classifiers? In *International Conference on Machine Learning*, pages 2029–2037. PMLR, 2018. 4.2

[13] Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. 4.3

[14] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMLR, 2019. 5.4

[15] R. Linsker. Self-organization in a perceptual network. *Computer*, 21(3):105–117, 1988. 5.4

[16] Duncan McElfresh, Sujay Khandagale, Jonathan Valverde, Vishak Prasad C, Ganesh Ramakrishnan, Micah Goldblum, and Colin White. When do neural nets outperform boosted trees on tabular data? *Advances in Neural Information Processing Systems*, 36, 2024. 5.1

[17] Aditya Krishna Menon, Sadeep Jayasumana, Ankit Singh Rawat, Himanshu Jain, Andreas Veit, and Sanjiv Kumar. Long-tail learning via logit adjustment. In *International Conference on Learning Representations*, 2021. 4.4

[18] 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*, pages 8748–8763. PMLR, 2021. 3.4

[19] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 1

[20] Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. In *International Conference on Learning Representations*, 2020. 4, 4.1

[21] Shiori Sagawa, Pang Wei Koh, Tony Lee, Irena Gao, Sang Michael Xie, Kendrick Shen, Ananya Kumar, Weihua Hu, Michihiro Yasunaga, Henrik Marklund, Sara Beery, Etienne David, Ian Stavness, Wei Guo, Jure Leskovec, Kate Saenko, Tatsunori Hashimoto, Sergey Levine, Chelsea Finn, and Percy Liang. Extending the WILDS benchmark for unsupervised adaptation. In *International Conference on Learning Representations*, 2022. 4.4

[22] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000. 4.1

[23] Martin J. Wainwright. *High-Dimensional Statistics: A Non-Asymptotic Viewpoint*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2019. 1

[24] Huan Wang, Shuicheng Yan, Dong Xu, Xiaoou Tang, and Thomas Huang. Trace ratio vs. ratio trace for dimensionality reduction. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007. 3.2

[25] Ke Alexander Wang, Niladri Shekhar Chatterji, Saminul Haque, and Tatsunori Hashimoto. Is importance weighting incompatible with interpolating classifiers? In *International Conference on Learning Representations*, 2022. 4.4

[26] Olivia Wiles, Sven Gowal, Florian Stimberg, Sylvestre-Alvise Rebuffi, Ira Ktena, Krishnamurthy Dj Dvijotham, and Ali Taylan Cemgil. A fine-grained analysis on distribution shift. In *International Conference on Learning Representations*, 2022. 4.4

[27] Zhuoning Yuan, Yuexin Wu, Zi-Hao Qiu, Xianzhi Du, Lijun Zhang, Denny Zhou, and Tianbao Yang. Provable stochastic optimization for global contrastive learning: Small batch does not harm performance. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 25760–25782. PMLR, 17–23 Jul 2022. 5.1

[28] 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*, pages 12310–12320. PMLR, 2021. 3.4

[29] Banghua Zhu, Jiantao Jiao, and Jacob Steinhardt. Generalized resilience and robust statistics. *The Annals of Statistics*, 50(4):2256 – 2283, 2022. 4.2