A Theoretical Analysis of Fine-tuning with Linear Teachers

Anonymous Author(s)
Affiliation
Address
email

Abstract

Fine-tuning is a common practice in deep learning, achieving excellent generalization results on downstream tasks using relatively little training data. Although widely used in practice, it is not well understood theoretically. Here we analyze the sample complexity of this scheme for regression with linear teachers in several settings. Intuitively, the success of fine-tuning depends on the similarity between the source tasks and the target task. But what is the right way of measuring this similarity? We show that the relevant measure has to do with the relation between the source task, the target task and the covariance structure of the target data. In the setting of linear regression, we show that under realistic settings there can be substantial sample complexity reduction when the above measure is low. For deep linear regression, we propose a novel result regarding the inductive bias of gradient-based training when the network is initialized with pretrained weights. Using this result we show that the similarity measure for this setting is also affected by the depth of the network. We conclude with results on shallow ReLU models, and analyze the dependence of sample complexity there on source and target tasks. We empirically demonstrate our results for both synthetic and realistic data.

1 Introduction

In recent years fine-tuning has emerged as an effective approach to learning tasks with relatively little labeled data. In this setting, a model is first trained on a source task where much data is available (e.g., masked language modeling for BERT), and then the model is further tuned using gradient descent methods on labeled data of a target task [1, 2, 3, 4]. It has also been observed that fine-tuning outperforms the strategy of fixing the representation learned on the source task in some settings, mainly in natural language processing [1, 5].

Despite its empirical success, fine-tuning is poorly understood from a theoretical perspective. One apparent conundrum in this context is that finetuned models can be much larger than the number of target training points, and thus the model is heavily overparameterized and can potentially suffer from overfitting and poor generalization. Indeed it is clear that there are models that would fit the target data well but generalize poorly. Thus, the answer must lie in the fact that fine-tuning is performed with gradient descent and not an arbitrary algorithm that could potentially "ignore" the source task. Here we set out to formalize this problem and understand the factors that determine whether fine-tuning will succeed. We note that this question can be viewed as part of the general quest to understand the implicit bias of gradient based methods [6, 7, 8, 9, 10, 11], but in the particular context of fine-tuning.

We begin by highlighting the obvious link between fine-tuning and initialization. Namely, the only difference between "standard" training of a target task and fine-tuning on it, is the initial value of the model weights before beginning the gradient updates. Thus, our goal is to understand the interplay between the model parameters at initialization (namely the source task), the target distribution, and

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

the accuracy of the fine-tuned model. A natural hypothesis is that the distance between the pretrained and fine-tuned models' weights is what governs the success of fine tuning. Indeed, some works argue that this is both the key to bound the generalization error of a model and the implicit regularization of gradient-based methods [12, 13, 14, 15]. However, this approach has been discouraged both by empirical testing of generalization bounds inspired by it [16] and by theoretical works showing this is cannot be the inductive bias in deep neural networks [17]. As we shall see, our results here further establish the fact that the success of fine-tuning is affected by other factors.

Here we focus on the case where both source and target regression tasks are linear functions of the input. We first consider the case where the model architecture is one layer linear networks, and derive novel sample complexity results for fine-tuning. We then proceed to the more complex case of deep linear networks. There we derive a novel result providing the learned model as a function of initialization, and use it to derive corresponding sample complexity bounds.

Our results above provide several surprising insights. The first is that the covariance structure of the target data has a significant effect on the success of fine-tuning. In particular the relation between the vector of the source-target difference and the eigenvectors of the target covariance will affect how well fine-tuning will work. Second, we find that the depth of the network can dramatically affect the results of the fine-tuning process, since deeper networks will serve to cancel the effect of scale differences between source and target tasks. Our results are corroborated by empirical evaluations.

We conclude with results on ReLU networks, where we provide the first sample complexity result for fine-tuning. For the case of linear teachers, these provide a simple connection between the source and target models and the test error of fine-tuning.

Taken together, our results demonstrate that fine-tuning is not only affected by some notion of distance between the source and target tasks, but also by the target covariance and the architecture of the model. These results improve our understanding of the fine-tuning process, and can potentially lead to improved accuracy in this setting by appropriate design of the tasks used for pretraining and the model architecture.

2 Related work

Empirical work [18] has shown that two models trained from a pretrained initialization with different randomness are similar functionally, and close to each other in the parameter space, compared to models initialized with random initialization. This observation encourages the idea that there is some inductive bias relating to the initial weights of the model that might explain the success of fine-tuning. [19] and [20] both show that fine-tuned models generalize well when the representation used by the target task approximately similar to the one used by the source tasks.

In linear regression, [21] showed that the inductive bias of gradient descent with infinite steps is to find the solution in the span of the data that has minimal distance to the initial weights. More recently, attention has turned towards the phenomenon of "beneign overfitting" in high dimensional linear regression, where despite fitting noise in training data, population risk may be low. In [22, 23] theoretical support was provided for this phenomenon, and [22] analyzed what covariance structure was required to observe it. Beneign overfitting was also recently analyzed in the context of ridge-regression [24] and online stochastic gradient descent [25]. Our work also addresses high dimensional regression, but our focus is different from the above works in that we start from a source task, and then train on a fixed training set from a target task (unlike online SGD). Furthermore, we go beyond the linear regression case, and obtain surprising characteristics of fine-tuning in deep linear networks.

For linear regression with deep linear models, [26] have recently shown an implicit bias for a two-layer network with deterministic initialization, and [27] have shown an implicit bias for a network with arbitrary depth and near-zero random initialization. Our work generalizes the inductive bias found by [26] to a network of arbitrary depth, and analyses the generalization error of such networks for infinite depth.

For linear regression with shallow linear networks [28] have shown a generalization bound that depends only on the norm of the target task, and which we use in Section 6.

3 Preliminaries and settings

Notations Let $\|\cdot\|$ be the L^2 norm for vectors and the spectral norm for matrices. For a vector $\boldsymbol{v}$ we denote $\hat{\boldsymbol{v}} \triangleq \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|}$. For a matrix $\mathbf{M} \in \mathbb{R}^{d\times d}$ and some $0 \leq m \leq d$, we define $\mathbf{M}_{\leq m} \in \mathbb{R}^{d\times m}$ to be the matrix containing the first m columns of $\mathbf{M}$. In a similar manner we define $\mathbf{M}_{>m}$, i.e. the matrix containing the columns indexed $m+1$ to d in $\mathbf{M}$.

Let $\mathcal{D}$ be a distribution over $\mathbb{R}^d$, and let $\mathbf{X} \in \mathbb{R}^{n\times d}$ be the row matrix of $n < d$ samples drawn from $\mathcal{D}$. Let $\boldsymbol{\Sigma}$ be the covariance matrix of $\mathcal{D}$ and let $\mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top$ be the eigenvalue decomposition of $\boldsymbol{\Sigma}$ such that $\lambda_1 \geq \ldots \geq \lambda_d$. Using the notation above, we define:

$$\mathbf{P}_{\leq k} \triangleq \mathbf{V}_{\leq k}\mathbf{V}_{\leq k}^\top; \quad \mathbf{P}_{>k} \triangleq \mathbf{V}_{>k}\mathbf{V}_{>k}^\top,$$

We observe that they are the projection matrices into the span of the top k eigenvectors of $\boldsymbol{\Sigma}$, to the span of the $d-k$ bottom eigenvectors of $\boldsymbol{\Sigma}$, accordingly. We will refer to the former as the "top-k span" of $\boldsymbol{\Sigma}$, and to the latter as the "bottom-k span" of $\boldsymbol{\Sigma}$. We abuse the definition of "top span" and "bottom span" to the k which maximizes $\lambda_k - \lambda_{k+1}$.

We denote the estimated covariance matrix $\frac{1}{n}\mathbf{X}^T\mathbf{X}$ by $\tilde{\boldsymbol{\Sigma}}$. Also, define $\mathbf{P}_\parallel$ to be the projection matrix into the row space of $\mathbf{X}$, and similarly define $\mathbf{P}_\perp$ to be the projection matrix into its orthogonal counterpart. Specifically, we define:

$$\mathbf{P}_\parallel \triangleq \mathbf{X}^\top(\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}, \quad \mathbf{P}_\perp \triangleq \mathbf{I} - \mathbf{P}_\parallel.$$

Consider a set of parameters $\boldsymbol{\Theta}$. We denote the output of a model whose weights are $\boldsymbol{\Theta}$ on a vector $\mathbf{x}$ by $f(\mathbf{x};\boldsymbol{\Theta}) \in \mathbb{R}$. In the different sections of this work we will overload f with different architectures.

We consider the problem of fine-tuning based transfer learning in regression tasks with linear teachers. Let $\boldsymbol{\theta}_T \in \mathbb{R}^d$ be the ground-truth parameters of the target task, i.e. the linear teacher which we wish to learn, and $\mathbf{y} \in \mathbb{R}^n$ be the target labels of $\mathbf{X}$, s.t. $\mathbf{y} = \mathbf{X}\boldsymbol{\theta}_T$.

We define $L(\boldsymbol{\Theta})$ to be the empirical MSE loss on $\mathbf{X}, \mathbf{y}$ and define $R(\boldsymbol{\Theta})$ as the $\mathcal{D}$ population loss:

$$L(\boldsymbol{\Theta}) \triangleq \frac{1}{n}\left(f(\mathbf{x};\boldsymbol{\Theta}) - \mathbf{y}\right)^2, \quad R(\boldsymbol{\Theta}) \triangleq \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[\left(\mathbf{x}^\top\boldsymbol{\theta}_T - f(\mathbf{x};\boldsymbol{\Theta})\right)^2\right].$$

We separate the training procedure into two parts. In the first part, which we call pretraining, we train a model on samples $\mathbf{X}_S \in \mathbb{R}^{n_S\times d}$ labeled by a linear teacher $\boldsymbol{\theta}_S$ such that $\mathbf{y}_S = \mathbf{X}_S\boldsymbol{\theta}_S \in \mathbb{R}^{n_S}$, achieving the set of model weights $\boldsymbol{\Theta}_S$. In the second part, which we call fine-tuning, we initialize a model with the pretrained weights $\boldsymbol{\Theta}(0) = \boldsymbol{\Theta}_S$ and learn the target task by optimizing $L(\boldsymbol{\Theta}(t))$.

Optimization is done by either gradient descent (GD) or gradient flow (GF). Let $\boldsymbol{\theta}(t)$ be some weight vector or weight matrix in $\boldsymbol{\Theta}(t)$. The dynamics for gradient descent optimization with some learning rate $\eta > 0$, and the dynamics for gradient flow are

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \eta\frac{\partial L(\boldsymbol{\Theta}(t))}{\partial\boldsymbol{\theta}(t)}, \quad \dot{\boldsymbol{\theta}}(t) = -\frac{\partial L(\boldsymbol{\Theta}(t))}{\partial\boldsymbol{\theta}(t)},$$

accordingly.

Assumptions Next we state several assumptions about our setup.

Assumption 3.1. *We assume $\mathbf{X}\mathbf{X}^\mathbf{T}$ is non-singular.*

This assumption holds with high probability for, e.g., i.i.d sampling. We could have incorporated this high probability into the analysis, but leave it as an assumption to simplify presentation.

Assumption 3.2 (Perfect pretraining)**.** *We assume that the pretraining optimization process learns the linear teach perfectly, i.e. that $f(\mathbf{x},\boldsymbol{\Theta}_S) = \mathbf{x}^\top\boldsymbol{\theta}_S$, for $\mathbf{x} \sim \mathcal{D}$.*

Our results can be easily extended to the case where the equality $f(\mathbf{x},\boldsymbol{\Theta}_S) = \mathbf{x}^\top\boldsymbol{\theta}_S$ holds approximately and with high probability, but for simplicity we assume equality.

Assumption 3.3 (Zero train loss)**.** *We assume that the fine-tuning converges. Namely, we assume that* $\lim_{t\to\infty} L(\boldsymbol{\Theta}(t)) = 0$.

We note that when f is a standard linear regression, arbitrarily small train loss can be obtained via gradient descent. For deep linear networks, it can be shown [29] that under suitable initialization a global optimum can be reached, and thus the assumption is also true there.

Code In our code we used Pytorch [30], Numpy [31], SciPy [32], and Matplotlib [33].

4 Analyzing fine-tuning in linear regression

In this section we explore the inductive bias and generalization properties of fine-tuning with linear teachers for linear regression with gradient descent. We define $\mathbf{\Theta} = \{\mathbf{w}\} \in \mathbb{R}^d$ and overload $f(\mathbf{x}, \mathbf{\Theta}) \triangleq \mathbf{x}^\top \mathbf{w}$. In what follows we define the parameter learned in the fine-tuning process by $\boldsymbol{\theta} = \lim_{t\to\infty} \mathbf{w}(t)$.

The following known results (taken from [21, 22]) show the inductive bias of gradient descent with non-zero initialization in under-determined linear regression and the corresponding population loss.

Theorem 4.1. *[21, 22] When $f(\mathbf{x}, \mathbf{\Theta})$ is a linear function, fine-tuning with GD will result in the following model:*

$$\boldsymbol{\theta} = \mathbf{P}_\perp \boldsymbol{\theta}_S + \mathbf{P}_\parallel \boldsymbol{\theta}_T, \tag{1}$$

and

$$R(\boldsymbol{\theta}) = \|\mathbf{\Sigma}^{1/2} \mathbf{P}_\perp \left(\boldsymbol{\theta}_T - \boldsymbol{\theta}_S\right)\|^2. \tag{2}$$

For completeness, the proof of Theorem 4.1 is given in the supplementary.

Theorem 4.1 provides two interesting observations: the first is that $\boldsymbol{\theta}$ consists of two parts, one that is in the null space of $row(\mathbf{X})$, and contains projection of the initial weights $\boldsymbol{\theta}_S$ into this space, and the other which is parallel to the train data, and contains the projection of $\boldsymbol{\theta}_T$ into $row(\mathbf{X})$. The second observation is that the population risk depends solely on the difference $\boldsymbol{\theta}_T - \boldsymbol{\theta}_S$ that is projected to the null space of the data.

Theorem 4.1 depends on the data matrix $\mathbf{X}$ (via $\mathbf{P}_\parallel, \mathbf{P}_\perp$). However, to better understand the properties of fine-tuning we would like a high probability bound on R that does not depend on $\mathbf{X}$. Next, we provide such a bound, which highlights the dependence of the population risk on the source and target tasks, and the target covariance Σ.

Theorem 4.2. *Assume rows of $\mathbf{X}$ are i.i.d. weakly square integrable centered random vectors with covariance operator $\mathbf{\Sigma}$. If $\mathbf{X}$ is subgaussian, then there exists a constant $c > 0$, such that, for all $1 \le t \le n$, and for all $1 \le m \le d$ such that $\lambda_m > 0$, with probability at least $1 - e^{-t}$ over $\mathbf{X}$, the population risk $R(\boldsymbol{\theta})$ is bounded by:*

$$\left[g(\boldsymbol{\lambda}, t, n)^{3/2} \frac{\|\mathbf{P}_{\le m}(\boldsymbol{\theta}_T - \boldsymbol{\theta}_S)\|}{\lambda_m} + g(\boldsymbol{\lambda}, t, n)^{1/2} \|\mathbf{P}_{>m}(\boldsymbol{\theta}_T - \boldsymbol{\theta}_S)\|\right]^2, \tag{3}$$

where $g(\boldsymbol{\lambda}, t, n) = c\lambda_1 \max\{\sqrt{\frac{\sum_i \lambda_i}{n\lambda_1}}, \frac{\sum_i \lambda_i}{n\lambda_1}, \sqrt{\frac{t}{n}}\}$ and $\left\|\tilde{\mathbf{\Sigma}} - \mathbf{\Sigma}\right\| \le g(\boldsymbol{\lambda}, t, n)$.

To tackle the randomness of $\mathbf{P}_\parallel(\boldsymbol{\theta}_T - \boldsymbol{\theta}_S)$, we decompose $\boldsymbol{\theta}_T - \boldsymbol{\theta}_S$ to its top-k span and bottom-k span components, and then apply the Davis-Kahan sin(Θ) theorem [34] to bound the norm of the projection of the former to the null space of the data. The proof of this theorem is given in the supplementary.

The bound in Theorem 4.2 has two key components. The first is the function $g(\boldsymbol{\lambda}, t, n)$ that captures how well the covariance $\mathbf{\Sigma}$ is estimated. The second are two matrix norms of $\boldsymbol{\theta}_T - \boldsymbol{\theta}_S$ with respect to different parts of the covariance $\mathbf{\Sigma}$. Clearly, as $g(\boldsymbol{\lambda}, t, n)$ decreases, the bound improves.

The theorem nicely highlights when fine-tuning is expected to work well. For small n when $g(\boldsymbol{\lambda}, t, n) > 1$ the bound mainly depends on $\|\mathbf{P}_{\le m}(\boldsymbol{\theta}_T - \boldsymbol{\theta}_S)\|$. In this case, the bound will be low if $\boldsymbol{\theta}_T$ and $\boldsymbol{\theta}_S$ are close in the span of the *top* eigenvectors of the target distribution. On the other hand, for large n when $g(\boldsymbol{\lambda}, t, n) < 1$, the bound mainly depends on $\|\mathbf{P}_{>m}(\boldsymbol{\theta}_T - \boldsymbol{\theta}_S)\|$. Thus, the bound will be low if $\boldsymbol{\theta}_T$ and $\boldsymbol{\theta}_S$ are close in the span of the *bottom* eigenvectors of the target distribution.

We conclude with a note on the integer m that appears in the bound. Although finding the exact m that minimizes the bound is not straightforward, the tradeoff in selecting it is as follows: we should increase m as long as $\lambda_{m+1} \approx \lambda_m$, since thus we can "cover" more of $\mathbf{P}_{>m}\left(\boldsymbol{\theta}_T - \boldsymbol{\theta}_S\right)$ without a major increase in the left part of Equation (3).

Table 1: Correlation coefficient R^2 between the accuracy on different transfer tasks in MNIST and various population risk upper bounds. Each value is a mean over 10 calculations of R^2 with different initialization, and each R^2 is calculated from 20 points, each one representing a mean accuracy value of 25 random samples.

Number of Samples	10	15	20	25	30
$\|\boldsymbol{\theta}_T - \boldsymbol{\theta}_S\|^2$	0.69 ± 0.03	0.68 ± 0.04	0.66 ± 0.04	0.64 ± 0.03	0.62 ± 0.02
Bound from [22]	0.73 ± 0.03	0.75 ± 0.03	0.74 ± 0.03	0.71 ± 0.02	0.67 ± 0.02
Ours for $m = 2$	$\mathbf{0.86 \pm 0.02}$	$\mathbf{0.89 \pm 0.02}$	$\mathbf{0.84 \pm 0.02}$	$\mathbf{0.75 \pm 0.01}$	$\mathbf{0.69 \pm 0.02}$

4.1 Experiments

In Figure 1 we empirically verify the conclusions from the bound in (3). We take $d = 1000$ and design the target covariance Σ s.t. the first $m = 50$ eigenvalues are much larger than the rest (1.5 vs. 0.3). We then consider two settings for $\boldsymbol{\theta}_T - \boldsymbol{\theta}_S$. In the first, which we call "Top Eigen Align" we select $\boldsymbol{\theta}_T$ and $\boldsymbol{\theta}_S$ such that $\mathbf{P}_{\leq m}(\boldsymbol{\theta}_T - \boldsymbol{\theta}_S) = 0$. In the second which we call "Bottom Eigen Align" we set $\mathbf{P}_{>m}(\boldsymbol{\theta}_T - \boldsymbol{\theta}_S) = 0$. In both settings we use the same norm $\|\boldsymbol{\theta}_T - \boldsymbol{\theta}_S\|_2$, to show that the bound is not affected by this norm.

As discussed above, our bound suggests better generalization performance of "Bottom Eigen Align" for large n and better performance of "Top Eigen Align" for small n. Indeed, we see that while for very few samples "Top Eigen Align" has a lower population loss than "Bottom Eigen Align", the population loss of "Bottom Eigen Align" drops significantly as n grows, and drops to zero well before $n = d$.

We next evaluate the bound on fine-tuning tasks taken from the MNIST dataset [35], and compare it to alternative bounds. Specifically, since we do not expect bounds to be numerically accurate, we calculate the correlation between the actual risk in the experiment and the risk predicted by the bounds. The task we consider (both source and target) is binary classification, which we model as regression to outputs $\{-1, +1\}$. We generate K (source,target) pairs (e.g., source is 2 vs 3 and target is 5 vs 6). For each such pair we perform source training followed by fine-tuning to target. We then record both the 0-1 error on an independent test set and the value predicted by the bounds. In this way we obtain K pairs of points (i.e., actual error vs bound), and we then calculate the R^2 for these pairs, indicating the level to which the bound agrees with the actual error. For the bounds, we consider in addition to our bound in Eq. (3) the following: the norm of source-target difference $\|\boldsymbol{\theta}_T - \boldsymbol{\theta}_S\|^2$ and a bound adapted from [22] to the case of fine-tuning.[1] The results in Table 1 show that there is a strong correlation between our bound and the actual error, and the correlation is weaker for the other bounds.

5 Analyzing fine-tuning in deep linear networks

The previous section considered a student network that is a linear function of its parameters. We next turn to the case where the student is a deep linear network. Although the resulting function is still linear in its inputs, we shall see that the effect of fine-tuning is markedly different. Indeed other works (e.g. [36, 37]) have shown that linear networks exhibit many interesting properties which make them a good study case towards more complex non-linear networks.

For this section, for a L-deep linear network we define $\boldsymbol{\Theta} = \{\mathbf{W}_1, \cdots, \mathbf{W}_L\}$ and:

$$\boldsymbol{\beta}(t) = \mathbf{W}_1(t) \cdot \mathbf{W}_2(t) \cdots \mathbf{W}_L(t),$$

such that $f(\mathbf{x}; \boldsymbol{\Theta})(t) = \mathbf{x}^\top \boldsymbol{\beta}(t)$. From Assumption 3.2, we get that $\boldsymbol{\beta}(0) = \boldsymbol{\theta}_S$.

We recall the condition of perfect balancedness (or 0-balancedness) [29]: The weights of an L deep linear model at time t are called 0-balanced if:

$$\mathbf{W}_j(t)^\top \mathbf{W}_j(t) = \mathbf{W}_{j+1}(t) \mathbf{W}_{j+1}(t)^\top \quad \text{for} \quad j \in [L-1]. \tag{4}$$

[1] The adaption is straightforward: since the population loss for non-random initialization depends on $\boldsymbol{\theta}_T - \boldsymbol{\theta}_S$ instead of $\boldsymbol{\theta}_T$, we replace the ground-truth expression $\boldsymbol{\theta}^\star$ in their bound for the bias component of the loss with $\boldsymbol{\theta}_T - \boldsymbol{\theta}_S$.

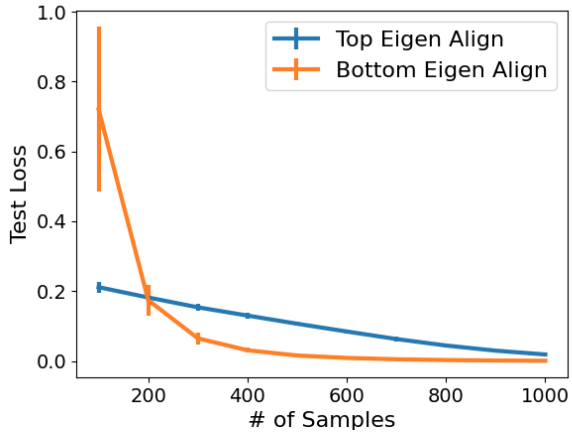


Figure 1: Comparison between different $\boldsymbol{\theta}_T - \boldsymbol{\theta}_S$. "Top" is the linear predictor initialized with $\mathbf{P}_{\leq m}(\boldsymbol{\theta}_T - \boldsymbol{\theta}_S) = 0$ and "Bottom" is the linear predictor initialized with $\mathbf{P}_{>m}(\boldsymbol{\theta}_T - \boldsymbol{\theta}_S) = 0$, for m=50. The top m eigenvalues have the value 1.5, compared to the rest which have the value 0.3.

Our analysis requires the initial random initialization (before the pretraining) to be 0-balanced, which can be achieved with a near zero random initialization, as discussed in [29]. We provide three results on the effect of fine-tuning in this setting. The first result shows the inductive bias of fine-tuning a L deep linear network (Theorem 5.1). This inductive bias holds for arbitrary L and generalizes known results for $L = 1$ Theorem 4.1 and $L = 2$ [26]. The second result analyzes the population risk of such a predictor when $L \to \infty$ for certain settings (Theorem 5.2 and Theorem 5.3). The third result shows why fixing the first layer (or any set of layers containing the first layer) after pretraining can harm fine-tuning (Theorem 5.4).

The next theorem characterizes the model learned by fine-tuning in the above setting (it can thus be viewed as the deep-linear version of the linear regression result in Theorem 4.1):

Theorem 5.1. *Assume that before pretraining, the weights of the model were 0-balanced and that Assumption 3.1, Assumption 3.2 and Assumption 3.3 hold, then*

$$\lim_{t\to\infty} \boldsymbol{\beta}(t) = \left(\frac{\|\boldsymbol{\beta}\|}{\|\boldsymbol{\theta}_S\|}\right)^{\frac{L-1}{L}} \mathbf{P}_\perp \boldsymbol{\theta}_S + \mathbf{P}_\parallel \boldsymbol{\theta}_T \tag{5}$$

and:

$$\lim_{L\to\infty} \lim_{t\to\infty} \boldsymbol{\beta}(t) = \frac{\|\mathbf{P}_\parallel \boldsymbol{\theta}_T\|}{\|\mathbf{P}_\parallel \boldsymbol{\theta}_S\|} \mathbf{P}_\perp \boldsymbol{\theta}_S + \mathbf{P}_\parallel \boldsymbol{\theta}_T. \tag{6}$$

To prove this, we focus on $\mathbf{W}_1$, and notice that the GF parameters $\dot{\mathbf{W}}_1(t)$ are in the span of $\mathbf{X}$, and hence $\mathbf{P}_\perp \mathbf{W}_1(0)$ remains static during the GF optimization, and its norm [27]. We then analyse the norm of $\lim_t t \to \infty \boldsymbol{\beta}(t)$ by using the 0-balancedness property of the weights and the min-norm solution to the equivalent linear regression problem, and achieve Equation (5). Equation (6) is achieved by calculating the limit. The proof of Theorem 5.1 is given in the supplementary. Though the expression in Equation (5) is not a closed form expression for $\boldsymbol{\beta}$ (because the norm $\boldsymbol{\beta}$ appears on the RHS), taking L to infinity (see (6)) does result in a closed form expression and demonstrates the effect of increasing model depth. As in Equation (1), we see that the end-to-end equivalent has two components: one which is parallel to the data and one which is orthogonal to the data. However, while in Equation (1) the orthogonal component has the original norm of the orthogonal projection of $\boldsymbol{\theta}_S$, the expression in Equation (6) offers a re-scaling of the norm of this component by some ratio that depends on $\boldsymbol{\theta}_S$, but also on $\boldsymbol{\theta}_T$.

5.1 When Does Depth Help Fine-Tuning?

We next wish to understand the effect of depth on the population risk of the fine-tuned model. For simplicity we focus on the case in Equation (6), and denote $\boldsymbol{\beta} = \lim_{L\to\infty} \lim_{t\to\infty} \boldsymbol{\beta}(t)$.

Since the linear network is a linear function of $\mathbf{x}$, we can derive an expression for the population risk of the network, as in Equation (2):

$$R(\boldsymbol{\beta}) = \left\| \boldsymbol{\Sigma}^{\frac{1}{2}} \mathbf{P}_{\perp} \left(\boldsymbol{\theta}_T - \frac{\|\mathbf{P}_{\|} \boldsymbol{\theta}_T\|}{\|\mathbf{P}_{\|} \boldsymbol{\theta}_S\|} \boldsymbol{\theta}_S \right) \right\|^2 . \tag{7}$$

However, since $\mathbf{P}_{\|}$ depends on the random matrix $\mathbf{X}$, without further assumptions this expression itself is not enough for us to understand the behaviour of $R(\boldsymbol{\beta})$.

Theorem 5.2 and Theorem 5.3 below provide a risk bound in this case. In both cases, we get that Equation (7) depends on $\|\boldsymbol{\theta}_T\| (\hat{\boldsymbol{\theta}}_T - \hat{\boldsymbol{\theta}}_S)$, i.e. the product of the norm of $\boldsymbol{\theta}_T$ and the difference of the *normalized* $\boldsymbol{\theta}_T$ and $\boldsymbol{\theta}_S$, compared to Equation (2) which depends on the difference between the un-normalized vectors. This observation further highlights the fact that the distance between source and target vectors is not a good predictor of fine-tuning accuracy for some architectures, because it shows that source and target can be very far, but fine-tuning can still succeed if they are aligned.

We formalize this in the next result, which holds for $\mathcal{D} = \mathcal{N}(0,1)^d$, i.e. i.i.d Gaussian.

Theorem 5.2. *Assume that the conditions of Theorem 5.1 hold, and that* $\mathbf{X} \sim \mathcal{N}(0,1)^{n \times d}$. *Then there exists a constant* $c > 0$ *such that for an* $\epsilon > 0$ *it holds that with probability at least* $1 - 4exp(-c\epsilon^2 n) - 2exp\left(-c\epsilon^2(d-n)\right)$ *the population risk is bounded as follows:*

$$R(\boldsymbol{\beta}) \leq (1+\epsilon)^2 \frac{d-n}{d} \|\boldsymbol{\theta}_T\|^2 \left\| \hat{\boldsymbol{\theta}}_T - \hat{\boldsymbol{\theta}}_S \right\|^2 + \zeta(\|\boldsymbol{\theta}_S\|)^2, \tag{8}$$

for $\zeta(\|\boldsymbol{\theta}_S\|) \approx \epsilon \|\boldsymbol{\theta}_S\|$.

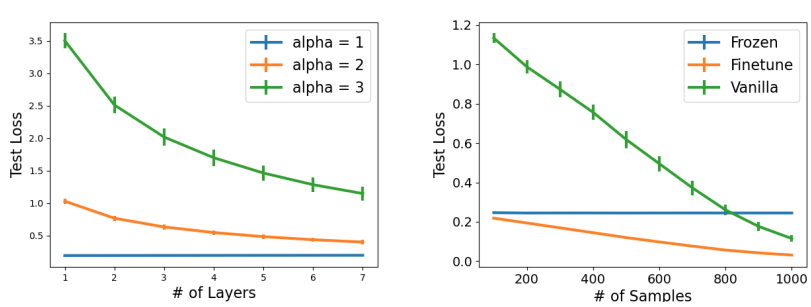
The above result is a direct analysis of Equation (7) when $\boldsymbol{\Sigma} = \mathbf{I}$ by using Lemma 5.3.2 from [38] to analyse the effects of $\mathbf{P}_{\|}, \mathbf{P}_{\perp}$. In this case Equation (2) will simplify to $R(\boldsymbol{\theta}) \leq (1+\epsilon)^2 \frac{d-n}{d} \|\boldsymbol{\theta}_T - \boldsymbol{\theta}_S\|^2$, which shows the exact difference mentioned above.

The second setting is when we assume $\boldsymbol{\theta}_T$ is identical to $\boldsymbol{\theta}_S$ in direction, but not in norm.

Theorem 5.3. *We Assume that the conditions of Theorem 5.1 hold, and that* $\boldsymbol{\theta}_T = \alpha \boldsymbol{\theta}_S$ *for* $\alpha > 0$, *then* $R(\boldsymbol{\beta}) = 0$ *and*

$$R(\boldsymbol{\theta}) = \left(\frac{\alpha - 1}{\alpha} \right)^2 \|\boldsymbol{\Sigma}^{1/2} \mathbf{P}_{\perp} \boldsymbol{\theta}_T\|^2 \neq 0 \quad \textit{for} \quad \alpha \neq 1, \alpha > 0. \tag{9}$$

The proof of Theorem 5.3 is given in the supplementary. This setting highlights our conclusion on the role of alignment in deep linear models: since the tasks are aligned, the deep linear predictor achieves zero generalization even with one sample, while the population risk of the linear predictor still depends on n.[2]



(a) **(b)**

Figure 2: (a) The effect of depth on fine-tuning when $\boldsymbol{\theta}_T$ is a α scaled, ϵ noised version of $\boldsymbol{\theta}_S$ with $d/10$ samples. (b) A frozen network has a constant generalization loss due to degeneration of the feature past the first layer.

[2]This can be seen by bounding the right part of Equation (9) with Theorem 4.2.

5.2 Deep linear fine-tuning with the first layer(s) frozen

A common trick when performing fine-tuning in practice is to "freeze" (i.e. not train) the first k layers of a model during fine-tuning on the target task. This method reduces the risk of over-fitting these layers to the small set of training sample, which is often called "Catastrophic forgetting". The next theorem shows why this method is not only non-beneficial in deep linear networks, but degenerates the training process.

Theorem 5.4. *Assume the setting of Theorem 5.1. Then, if we freeze the first layer (or any number k of first layers) during fine-tuning, the fine-tuned model will be given by* $\lim_{t\to\infty}\langle \boldsymbol{\beta}, \boldsymbol{x}\rangle = c\langle \mathbf{x}, \boldsymbol{\theta}_S\rangle$, *for some constant c.*

The key idea in the proof is to show that the product of the k first layers will be equal to $\boldsymbol{\theta}_S$ up to scaling factor. This is a result of [27]. The result implies that after fine-tuning the model will still be equal to the source task, and will not depend on the target task at all. Thus, fine-tuning essentially fails completely, and its error cannot be reduced with additional target data.

5.3 Experiments

We next provide empirical validation for the results in this section. Theorem 5.3 predicts that deeper nets will successfully learn a case where source and target vectors are aligned, but with different norms. This is demonstrated in Figure 2a where source and target tasks are related via $\boldsymbol{\theta}_T = \alpha\boldsymbol{\theta}_S + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ is a standard Gaussian vector whose norm is not too small compared to $\|\boldsymbol{\theta}_T\|$. It can be seen that when $\alpha \approx 1$, there is no difference between models of different depth. However, as α increases, adding depth has a positive effect on fine-tuning accuracy.

Theorem 5.4 states that freezing the first layer in deep linear nets can result in failure to fine-tune. We illustrate this empirically in Figure 2b, where we compare three two-layer deep linear models on the same target task: 1) A "Frozen" model that freeze the first layer after pretraining. 2) A "Vanilla" model that trains the network from scratch on the target, ignoring the source pre-training. 3) A "Finetune" model that first trains on source and fine-tunes to target. It can be seen that the "Frozen" model completely fails, as predicted by theory, and that fine-tuning has better sample complexity.

6 Analyzing fine-tuning in shallow ReLU networks

Analyzing optimization and generalization in non-linear networks is challenging. However, analysis in the Neural Tangent Kernel (NTK) regime is sometimes simpler [39, 28]. Thus, here we take a first step towards understanding fine-tuning in non-linear networks by analyzing this problem in the NTK regime. Specifically, we consider the setting of a two-layer ReLU network with m neurons in the hidden layer. Hence, we consider $\boldsymbol{\Theta} = \{\mathbf{W}, \boldsymbol{a}\}$ and $f(\mathbf{x}; \boldsymbol{\Theta}) = \frac{1}{\sqrt{m}}\sum_{r=1}^{m} a_r\sigma(\mathbf{x}^\top\mathbf{w})$ where $\mathbf{w}_1, \ldots, \mathbf{w}_m \in \mathbb{R}^d$ are vectors in the first layer, and $\boldsymbol{a} \in \{-1, 1\}^m$ is the vector of weights in the second layer. We initialize $\boldsymbol{a}$ uniformly and fix it during optimization as in [39]. For pretraining, first layer parameters are initialized from a standard Gaussian with variance κ^2. We also assume that $\|\mathbf{x}\| = 1$ for all $\mathbf{x}$ and $|y| \leq 1$. We let $f(\mathbf{X}, \boldsymbol{\Theta})\,\mathbb{R}^n$ be the vector of predictions of f on the data $\mathbf{X}$.

For the next theorem we do not assume linear teachers. We assume an arbitrary labeling function g_S such that $\mathbf{y}_S = g_S(\mathbf{X}_S)$, for $\mathbf{X}_S \in \mathbb{R}^{n_S\times d}, \mathbf{y}_S \in \mathbb{R}^{n_S}$ the pretraining data and labels, respectively. We also assume that $\mathbf{y} = g_T(\mathbf{X})$ for some arbitrary function g_T. We consider a setting where the pretraining phase is done using a two-layer network in the NTK regime, under the assumptions of Theorem 4.1 from [28] with respect to the variables m, κ, η and sufficiently many iterations.[3] Next, in the finetuning phase, we train a network initialized with the weights given by the pretraining phase. We use the same values of m, κ and η for the finetuning phase.

We rely on the analysis given in [39, 28] and achieve an upper bound on the population risk of the fine-tuned model:

Theorem 6.1. *Fix a failure probability $\delta \in (0, 1)$. We assume that Assumption 3.1 holds. Suppose $\kappa = O\left(\frac{\lambda_0\delta}{n}\right)$, $m \geq \kappa^{-2}\,\mathrm{poly}\left(n, n_S, \lambda_0^{-1}, \delta^{-1}\right)$. Consider any loss function $\ell : \mathbb{R}\times\mathbb{R} \to [0, 1]$ that is 1-Lipschitz in the first argument such that $\ell(y, y) = 0$. Then with probability at least $1 - \delta$,[4] the*

[3] See supplementary for a bound on the number of iterations.

[4] Over the random initialization of the pretraining network and the training samples used for finetuning.

two-layer neural network $f(\cdot, \mathbf{\Theta}(t))$ fine-tuned by GD for $t \geq \Omega\left(\frac{1}{\eta\lambda_0}\log\|\tilde{\mathbf{y}}\|_2^{-1}\right)$ iterations has population loss:

$$R\left(\mathbf{\Theta}(t)\right) \leq 2\sqrt{\frac{\tilde{\mathbf{y}}^\top\left(\mathbf{H}^\infty\right)^{-1}\tilde{\mathbf{y}}}{n}} + O\left(\sqrt{\frac{\log\frac{n}{\lambda_0\delta}}{n}}\right), \tag{10}$$

for $\tilde{\mathbf{y}} \equiv \mathbf{y} - f\left(\mathbf{X}, \mathbf{\Theta}(0)\right)$.

The above result shows that the true risk of the fine-tuned model is related to the distance of learned outputs $\mathbf{y}$ from the initial outputs $f\left(\mathbf{X}, \mathbf{\Theta}(0)\right)$ (after pretraining). In what follows, we use this to analyze fine-tuning with linear teachers. The proof of Theorem 6.1 is given in the supplementary. As in previous NTK analysis, the proof approach is to bound the distance between the Gram matrix $\mathbf{H}(t)$ and the infinite-width gram matrix $\mathbf{H}^\infty$ with a decreasing function in m. The main challenge is that the weights $\mathbf{W}(0)$ are not initialized i.i.d as described above. To address this we provide a careful analysis of the dynamics and show that $\mathbf{H}(t)$ is close to $\mathbf{H}(0)$ which in turn is close to $\mathbf{H}^\infty$ by the pretraining phase.

We next apply our results to the case of linear source and target tasks. We thus assume that g_S, g_T are linear functions with parameters $\boldsymbol{\theta}_S, \boldsymbol{\theta}_T$. For simplicity of exposition we assume $f\left(\mathbf{x}, \mathbf{\Theta}(0)\right) = \mathbf{x}^\top\boldsymbol{\theta}_S$ exactly (Assumption 3.2).

Before bounding the risk of fine-tuning we bound the RHS of (10) in the linear case.

Corollary 6.2. *Suppose that $g_S(\mathbf{X}) \triangleq \mathbf{X}^\top\boldsymbol{\theta}_S$, $g_T(\mathbf{X}) \triangleq \mathbf{X}^\top\boldsymbol{\theta}_T$, and assume Assumption 3.2 holds. Then, $\sqrt{\tilde{\boldsymbol{y}}^\top(\mathbf{H}^\infty)^{-1}\tilde{\boldsymbol{y}}} \leq 3\left\|\boldsymbol{\theta}_T - \boldsymbol{\theta}_S\right\|_2$.*

This is a direct corollary of Theorem 6.1 from [28] on $\tilde{\mathbf{y}}$ defined above. Theorem 6.1 and Corollary 6.2 result in the a bound on the risk of the finetuned model:

Corollary 6.3. *Under the conditions of Theorem 6.1 and Corollary 6.2, it holds that $R(\mathbf{\Theta}(t)) \leq \frac{6\|\boldsymbol{\theta}_T - \boldsymbol{\theta}_S\|_2}{\sqrt{n}} + O\left(\sqrt{\frac{\log\frac{n}{\lambda_0\delta}}{n}}\right)$*

It can be seen that fine-tuning is improved as the distance between source and target decreases. In our analysis of linear networks (Theorem 4.2 and Theorem 5.2) we obtained a more fine-grained result depending on the covariance structure. We anticipate that the non-linear case has similar results, but these will be more elaborate as they are likely to involve the covariance structure in the NTK feature space. We leave these for future work.

7 Discussion

This paper gives a fine-grained analysis of the process of fine-tuning with linear teacher in several different architectures. This analysis offers the insights into the inductive bias of gradient-descent optimization and the implied relation between the source task, the target task and the target covariance that is needed for this process to succeed. We believe our conclusions pave a way towards understanding why some pretrained models work better than others and what biases are transferred from that model during fine-tune.

A limitation of our work is the simplicity of the models analysed, and it would certainly be interesting to extend these. Our setting deals only with linear teachers, and assumes the label noise to be zero. Furthermore, we only show upper bounds on the population loss, and not matching lower bounds. For deep linear networks we assume a certain initialization which is less standard than normalized initializers like Xavier. For non-linear models, we analyse the simple model of a Shallow-ReLU network, and only in the NTK regime.

An interesting direction to explore is to try a formulate a bound similar to $Theorem\ 4.2$ for regression in the RKHS space given by the NTK, where the covariance is now over the RKHS space and thus more challenging to analyze. Another interesting setting is classification with exponential losses. Since the classifier learned by GD in this case has diverging norm, it is not clear how fine-tuning is beneficial, although in practice it often is. We leave these questions for future work.

References

[1] 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, 2019.

[2] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, 2019.

[3] Mor Geva, Ankit Gupta, and Jonathan Berant. Injecting numerical reasoning skills into language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 946–958, 2020.

[4] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, 2020.

[5] Matthew E Peters, Sebastian Ruder, and Noah A Smith. To tune or not to tune? adapting pre-trained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 7–14, 2019.

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

[7] Blake Woodworth, Suriya Gunasekar, Jason D Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. Kernel and rich regimes in overparametrized models. In *Conference on Learning Theory*, pages 3635–3673. PMLR, 2020.

[8] Lenaic Chizat and Francis Bach. Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss. In *Conference on Learning Theory*, pages 1305–1338. PMLR, 2020.

[9] Jingfeng Wu, Difan Zou, Vladimir Braverman, and Quanquan Gu. Direction matters: On the implicit bias of stochastic gradient descent with moderate learning rate.

[10] Edward Moroshko, Blake E Woodworth, Suriya Gunasekar, Jason D Lee, Nati Srebro, and Daniel Soudry. Implicit bias in deep linear classification: Initialization scale vs training accuracy. *Advances in Neural Information Processing Systems*, 33, 2020.

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

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

[13] Mingchen Li, Mahdi Soltanolkotabi, and Samet Oymak. Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 4313–4324. PMLR, 2020.

[14] Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. Towards understanding the role of over-parametrization in generalization of neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

[15] Gintare Karolina Dziugaite and Daniel M Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017.

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

[17] Noam Razin and Nadav Cohen. Implicit regularization in deep learning may not be explainable by norms. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[18] Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? *arXiv preprint arXiv:2008.11687*, 2020.

[19] Kurtland Chua, Qi Lei, and Jason D Lee. How fine-tuning allows for effective meta-learning. *arXiv preprint arXiv:2105.02221*, 2021.

[20] Daniel McNamara and Maria-Florina Balcan. Risk bounds for transferring representations with and without fine-tuning. In *International Conference on Machine Learning*, pages 2373–2381. PMLR, 2017.

[21] Suriya Gunasekar, Jason Lee, Daniel Soudry, and Nathan Srebro. Characterizing implicit bias in terms of optimization geometry. In *International Conference on Machine Learning*, pages 1832–1841. PMLR, 2018.

[22] Peter L Bartlett, Philip M Long, Gábor Lugosi, and Alexander Tsigler. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, 117(48):30063–30070, 2020.

[23] Trevor Hastie, Andrea Montanari, Saharon Rosset, and Ryan J Tibshirani. Surprises in high-dimensional ridgeless least squares interpolation. *arXiv preprint arXiv:1903.08560*, 2019.

[24] Alexander Tsigler and Peter L Bartlett. Benign overfitting in ridge regression. *arXiv preprint arXiv:2009.14286*, 2020.

[25] Difan Zou, Jingfeng Wu, Vladimir Braverman, Quanquan Gu, and Sham M Kakade. Benign overfitting of constant-stepsize sgd for linear regression. *arXiv preprint arXiv:2103.12692*, 2021.

[26] Shahar Azulay, Edward Moroshko, Mor Shpigel Nacson, Blake Woodworth, Nathan Srebro, Amir Globerson, and Daniel Soudry. On the implicit bias of initialization shape: Beyond infinitesimal mirror descent. *arXiv preprint arXiv:2102.09769*, 2021.

[27] Chulhee Yun, Shankar Krishnan, and Hossein Mobahi. A unifying view on implicit bias in training linear neural networks. *arXiv preprint arXiv:2010.02501*, 2020.

[28] Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pages 322–332. PMLR, 2019.

[29] Sanjeev Arora, Nadav Cohen, Noah Golowich, and Wei Hu. A convergence analysis of gradient descent for deep linear neural networks. In *International Conference on Learning Representations*, 2018.

[30] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[31] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[32] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A.

Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[33] John D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science Engineering*, 9(3):90–95, 2007.

[34] Chandler Davis and William Morton Kahan. The rotation of eigenvectors by a perturbation. iii. *SIAM Journal on Numerical Analysis*, 7(1):1–46, 1970.

[35] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[36] Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In *International Conference on Machine Learning*, pages 244–253. PMLR, 2018.

[37] Ziwei Ji and Matus Telgarsky. Gradient descent aligns the layers of deep linear networks. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[38] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

[39] Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] In the discussion.
   (c) Did you discuss any potential negative societal impacts of your work? [No] Our work is completely theoretical. If anything, it can benefit by advancing the understanding of the biases in pretrained models.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] In preliminaries and settings
   (b) Did you include complete proofs of all theoretical results? [Yes] Yes, in the supplementary
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Yes in the supplementary
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Yes in the supplementary
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Yes.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Yes in the supplementary
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] in the preliminaries
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] Using a benchmark dataset (MNIST)

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] Using a benchmark dataset (MNIST)

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] we didn't use crowdsourcing or conducted research with human subjects.

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] we didn't use crowdsourcing or conducted research with human subjects.

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] we didn't use crowdsourcing or conducted research with human subjects.