What can linearized neural networks actually say about generalization?

Anonymous Author(s)
Affiliation
Address
email

Abstract

For certain infinitely-wide neural networks, the neural tangent kernel (NTK) theory fully characterizes generalization. However, for the networks used in practice, the empirical NTK represents only a rough first-order approximation of these architectures. Still, a growing body of work keeps leveraging this approximation to successfully analyze important deep learning phenomena and derive algorithms for new applications. In our work, we provide strong empirical evidence to determine the practical validity of such approximation by conducting a systematic comparison of the behaviour of different neural networks and their linear approximations on different tasks. We show that the linear approximations can indeed rank the learning complexity of certain tasks for neural networks, albeit with important nuances. Specifically, we discover that, in contrast to what was previously observed, neural networks do not always perform better than their kernel approximations, and reveal that their performance gap heavily depends on architecture, number of samples and training task. In fact, we show that during training, deep networks increase the alignment of their empirical NTK with the target task, which explains why linear approximations at the end of training can better explain the dynamics of deep networks. Overall, our work provides concrete examples of novel deep learning phenomena which can inspire future theoretical research, as well as provides a new perspective on the use of the NTK approximation in deep learning.

1 Introduction

Due to their excellent practical performance, deep learning based methods are today the *de facto* standard for many visual and linguistic learning tasks. Nevertheless, despite this practical success, our theoretical understanding of how, and what, can neural networks learn is still in its infancy [1]. Recently, a growing body of work has started to explore the use of linear approximations to analyze deep networks, giving rise to the neural tangent kernel (NTK) framework [2].

The NTK framework is based on the observation that for certain initialization schemes, the infinite-width limit of many neural architectures can be exactly characterized using kernel tools [2]. This reduces key questions in deep learning theory to the study of linear methods and convex functional analysis, for which a rich set of theories exist [3]. This intuitive approach has been proved very fertile, leading to important results in generalization and optimization of very wide networks [4–9].

The NTK theory, however, can only fully describe certain infinitely wide neural networks. For the narrow architectures used in practice, it only provides a first-order approximation of their training dynamics (see Fig. 1). Despite these limitations, the intuitiveness of the NTK, which allows to use a powerful set of theoretical tools to exploit it, has lead to a rapid increase in the amount of work that successfully leverages the NTK in applications, such as predicting generalization [10] and training speed [11], explaining certain inductive biases [12–15] or designing new classifiers [16, 17].

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

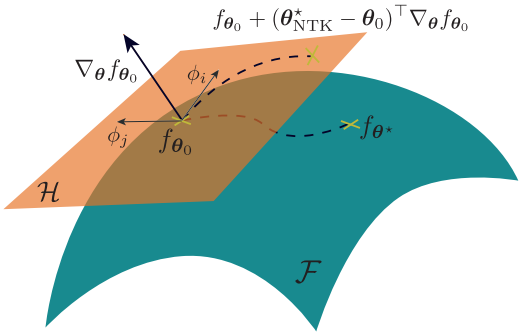

Figure 1: Conceptual illustration of the NTK approximation (see Sec 2). The empirical NTK defines a linear function space tangent $\mathcal{H}$ to the non-linear function space $\mathcal{F}$ defined by the network. In the limit of infinite width, the neural network space loses its curvature and coincides with the tangent space. Training a linearized network restricts the optimization trajectory to lie in the tangent space.

Recent reports, however, have started questioning the effectiveness of this approach, as one can find multiple examples in which kernel methods are provably outperformed by neural networks [18–20]. Most importantly, it has been observed empirically that linearized models – computed using a first-order Taylor expansion *around the initialization* of a neural network – perform much worse than the networks they approximate on standard image recognition benchmarks [21]; a phenomenon which has been coined the *non-linear advantage*. However, it has also been observed that, if one linearizes the network at a later stage of training, the non-linear advantage is greatly reduced [21]. The reasons behind this phenomenon are poorly understood, but are key to explain the success of deep learning.

Building from these observations, in this work, we delve deeper into the source of the non-linear advantage, trying to understand why previous work could successfully leverage the NTK in some applications. In particular, we shed new light on the question: *When can the NTK approximation be used to predict generalization, and what does it actually say about it?* We propose, for the first time, to empirically study this problem from the perspective of the characteristics of the training labels. To that end, we conduct a systematic analysis comparing the performance of different neural network architectures with their kernelized versions, on different problems with the same data support, but different labels. Doing so, we identify the alignment of the target function with the NTK as a key quantity governing important aspects of generalization in deep learning. Namely, one can rank the learning complexity to solve certain tasks using deep networks based solely on their kernel alignment.

Moreover, we study the evolution of this alignment during training. Interestingly, we discover that optimizing a deep network significantly increases the alignment of its empirical NTK with the target function. This phenomenon allows networks to explore functions beyond the limitations imposed to their linearization, and improves their training speed. Nonetheless, we also show that these kernel dynamics do not always have a positive effect in generalization, and we provide concrete examples where deep networks exhibit a *non-linear disadvantage* compared to their kernel approximations.

The main contributions of our work are:

- We show that the alignment with the empirical NTK at initialization can provide a good measure of relative learning complexity in deep learning for a diverse set of tasks.
- We identify, for the first time, a set of non-trivial tasks in which neural networks perform worse than their linearized approximations. This highlights the strong dependency of the *non-linear advantage* on the properties of the target function.
- We, hence, provide a fine-grained analysis of the evolution of the empirical NTK during training, showing that training increases its alignment with the training labels.
- Finally, we show that this adaptation towards the target function during training is responsible for the rapid convergence of neural networks, but that it also abides by its own inductive bias which can either benefit, or hurt generalization, depending on the target task.

We see our empirical findings as an important step forward in our understanding of deep learning. Our work paves the way for new research avenues based on our newly observed phenomena, but it also provides a fresh perspective to understand how to use the NTK approximation in several applications.

2 Preliminaries

Let $f_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}$ denote a neural network parameterized by a set of weights $\boldsymbol{\theta} \in \mathbb{R}^n$. Without loss of generality, but to reduce the computational load, in this work, we only consider the binary classification setting, where the objective is to learn an underlying target function $f : \mathbb{R}^d \to \{\pm 1\}$ by training the weights to minimize the empirical risk $\hat{\mathcal{R}}(f_{\boldsymbol{\theta}}) = \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}_{\text{sign}(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) \neq f(\boldsymbol{x}_i)}$ over a finite set of i.i.d. training data $\mathcal{S} = \{(\boldsymbol{x}_i, f(\boldsymbol{x}_i))\}_{i=1}^{m}$ from an underlying distribution $\mathcal{D}$. We broadly say that a model *generalizes* whenever it achieves a low expected risk $\mathcal{R}(f_{\boldsymbol{\theta}}) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[\mathbb{1}_{\text{sign}(f_{\boldsymbol{\theta}}(\boldsymbol{x}) \neq f(\boldsymbol{x})}]$.

In a small neighborhood around the weight initialization $\boldsymbol{\theta}_0$, a neural network can be approximated using a first-order Taylor expansion (see Fig. 1)

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) \approx \hat{f}_{\boldsymbol{\theta}}(\boldsymbol{x}; \boldsymbol{\theta}_0) = f_{\boldsymbol{\theta}_0}(\boldsymbol{x}) + (\boldsymbol{\theta} - \boldsymbol{\theta}_0)^\top \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_0}(\boldsymbol{x}), \tag{1}$$

where $\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_0}(\boldsymbol{x}) \in \mathbb{R}^n$ denotes the Jacobian of the network with respect to the parameters evaluated at $\boldsymbol{\theta}_0$. Here, the model $\hat{f}_{\boldsymbol{\theta}}$ represents a *linearized network* which maps weight vectors to functions living in a reproducible kernel Hilbert space (RKHS) $\mathcal{H} \subseteq L_2(\mathbb{R}^d)$, determined by the empirical neural tangent kernel (NTK) at $\boldsymbol{\theta}_0$, $\boldsymbol{\Theta}_{\boldsymbol{\theta}_0}(\boldsymbol{x}, \boldsymbol{x}') = \langle \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_0}(\boldsymbol{x}), \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_0}(\boldsymbol{x}') \rangle$ [2,4]. Unless stated otherwise, we will generally drop the dependency on $\boldsymbol{\theta}_0$ and use $\boldsymbol{\Theta}$ to refer to the NTK at initialization.

In most contexts, the NTK evolves during training by following the trajectory of the network Jacobian $\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_t}$ computed at a checkpoint $\boldsymbol{\theta}_t$ (see Fig. 1). Remarkably, however, it was recently discovered that for certain types of initialization, and in the limit of infinite network-width, the approximation in (1) is provably exact [2], making the NTK constant throughout training. In this regime, one can, therefore, provide generalization guarantees for neural networks using generalization bounds from kernel methods [3], and show [22], for instance, that with high probability

$$\mathcal{R}(f^\star) \leq \hat{\mathcal{R}}(f^\star) + \mathcal{O}\left(\sqrt{\frac{\|f\|_{\boldsymbol{\Theta}}^2 \mathbb{E}_{\boldsymbol{x}}[\boldsymbol{\Theta}(\boldsymbol{x}, \boldsymbol{x})]}{m}}\right), \tag{2}$$

where $f^\star = \arg\min_{h \in \mathcal{H}} \hat{\mathcal{R}}(h) + \mu \|h\|_{\boldsymbol{\Theta}}^2$, with $\mu > 0$ denoting a regularization constant, and $\|f\|_{\boldsymbol{\Theta}}^2$ being the RKHS norm of the target function, which for positive definite kernels can be computed a

$$\|f\|_{\boldsymbol{\Theta}}^2 = \sum_{j=1}^{\infty} \frac{1}{\lambda_j} \left(\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[\phi_j(\boldsymbol{x}) f(\boldsymbol{x})\right]\right)^2. \tag{3}$$

Here, the couples $\{(\lambda_j, \phi_j)\}_{j=1}^{\infty}$ denote the eigenvalue-eigenfunction pairs, in order of decreasing eigenvalues, of the Mercer's decomposition of the kernel, i.e., $\boldsymbol{\Theta}(\boldsymbol{x}, \boldsymbol{x}') = \sum_{j=1}^{\infty} \lambda_j \phi_j(\boldsymbol{x}) \phi_j(\boldsymbol{x}')$. This means that, in kernel regimes, the difference between the empirical and the expected risk is smaller when training on target functions with a lower RKHS norm. That is, whose projection on the eigenfunctions of the kernel is mostly concentrated along its largest eigenvalues. One can then see the RKHS norm as a metric that ranks the complexity of learning different targets using the kernel $\boldsymbol{\Theta}$.

Estimating (3) in practice is challenging as it requires access to the smallest eigenvalues of the kernel. However, one can use the following lemma to compute a more tractable bound of the RKHS norm, which shows that a high target-kernel alignment is a good proxy for a small RKHS norm.

Lemma 1. *Let $\alpha(f) = \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim\mathcal{D}}\left[f(\boldsymbol{x})\boldsymbol{\Theta}(\boldsymbol{x}, \boldsymbol{x}')f(\boldsymbol{x}')\right]$ denote the alignment of the target $f \in \mathcal{H}$ with the kernel $\boldsymbol{\Theta}$. Then $\|f\|_{\boldsymbol{\Theta}}^2 \geq \|f\|_2^2/\alpha(f)$. Moreover, for the NTK, $\alpha(f) = \|\mathbb{E}_{\boldsymbol{x}}\left[f(\boldsymbol{x})\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_0}(\boldsymbol{x})\right]\|_2^2$.*

Proof See supp. material.

At this point, it is important to highlight that in most practical applications we do not deal with infinitely-wide networks, and hence (2) can only be regarded as a learning guarantee for linearized models, i.e. $\hat{f}_{\boldsymbol{\theta}}$. Furthermore, from now on, we will interchangeably use the terms NTK and empirical NTK [2, 4] to simply refer to the finite-width kernels derived from (1). In our experiments, we compute those using the `neural_tangents` library [23] built on top of the JAX framework [24], which we also use to generate the linearized models. Similarly, as it is commonly done in the kernel literature, we will use the eigenvectors of the kernel Gram matrix to approximate the values of the eigenfunctions $\phi_j(\boldsymbol{x})$ over a finite dataset. We will, thus, use the terms eigenvector and eigenfunction interchangeably. In particular, we will use $\boldsymbol{\Phi} \in \mathbb{R}^{m \times m}$ to denote the matrix containing the jth Gram eigenvector $\boldsymbol{\phi}_j \in \mathbb{R}^m$ in its jth row, where the rows are ordered according to the vector of decreasing eigenvalues $\boldsymbol{\lambda} \in \mathbb{R}_+^m$. A more detailed description of this setup can be found in the supp. material.

3 Linearized models can predict relative task complexity for deep networks

As mentioned in Sec. 1 a growing body of work is using the linear approximation of neural networks as kernel methods to analyze and build novel algorithms [10–14, 16, 17]. Meanwhile, recent reports, both theoretical and empirical, have started to question if the NTK approximation can really tell anything useful about generalization for finite-width networks [18–21]. In this section, we try to demystify some of these confusions and aim to shed light on the question: *What can the empirical NTK actually predict about generalization?*

To that end, we conduct a systematic study with different neural networks and their linearized approximations given by (1), which we train to solve a structured array of predictive tasks with different complexity. Our results indicate that for many problems the linear models and the deep networks do agree in the way they *order* the complexity of learning certain tasks, even if their performance on the same problems can greatly differ. This highlights why the NTK approximation can be successfully applied in many contexts in which the main goal is to predict *just* relative differences in performance across tasks for neural networks.

3.1 Learning NTK eigenfunctions

In kernel theory, the sample and optimization complexity required to learn a given function is normally bounded by its kernel norm [3], which intuitively measures the alignment of the target function with the eigenfunctions of the kernel. The eigenfunctions themselves, thus, represent a natural set of target functions with increasingly high learning complexity – according to the increasing value of their associated eigenvalues – for kernel methods. Since our goal is to find if the kernel approximation can indeed predict generalization for neural networks, we evaluate the performance of these networks when learning the eigenfunctions of their NTKs.

In particular, we generate a sequence of datasets constructed using the standard CIFAR10 [25] samples, which we label using different binarized versions of the NTK eigenfunctions. That is, to every sample $\boldsymbol{x}$ in CIFAR10 we assign it the label $\mathrm{sign}(\phi_j(\boldsymbol{x}))$, where ϕ_j represents the jth eigenfunction of the NTK at initialization (see Sec. 2). In this construction, the choice of CIFAR10 as supporting distribution makes our experiments close to real settings which might be conditioned by low dimensional structures in the data manifold [19, 26]; while the choice of eigenfunctions as targets guarantees a progressive increase in complexity, *at least*, for the linearized networks. Specifically, for ϕ_j the alignment is given by $\alpha(\phi_j) = \lambda_j$ (see supp. material for the detailed proof).

We train different neural network architectures – selected to cover the spectrum of small to large models [27–29] – and their linearized models given by (1). Unless stated otherwise, we use the same standard training procedure consisting of the use of stochastic gradient descent (SGD) to optimize a binary cross-entropy loss, with a decaying learning rate starting at 0.05 and momentum set to 0.9, in all our experiments. Our results report values of different metrics at the end of 100 epochs of training.

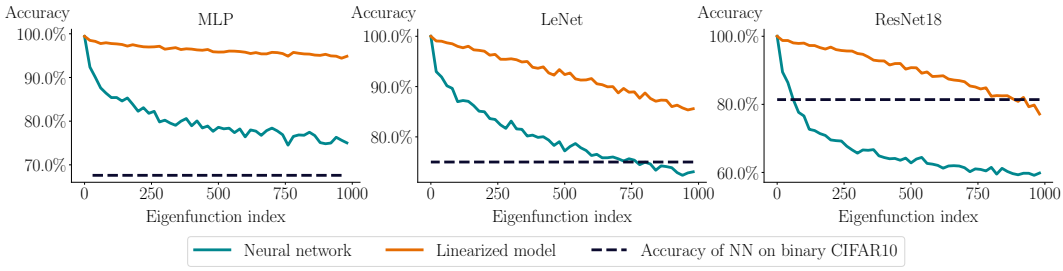

Figure 2: Validation accuracy of different neural network architectures and their linearizations when trained on binarized eigenfunctions of the NTK at initialization, i.e., $\boldsymbol{x} \mapsto \mathrm{sign}(\phi_j(\boldsymbol{x}))$. As a baseline, we also provide the accuracies on CIFAR2 (see Sec. 4).

Fig. 2 summarizes the main results of our experiments[1]. Here, we can clearly see how the validation accuracy of networks trained to predict targets aligned with $\{\phi_j\}_{j=1}^{\infty}$ progressively drops with

[1]Results with equivalent findings for other training schemes and datasets can be found in the supp. material.

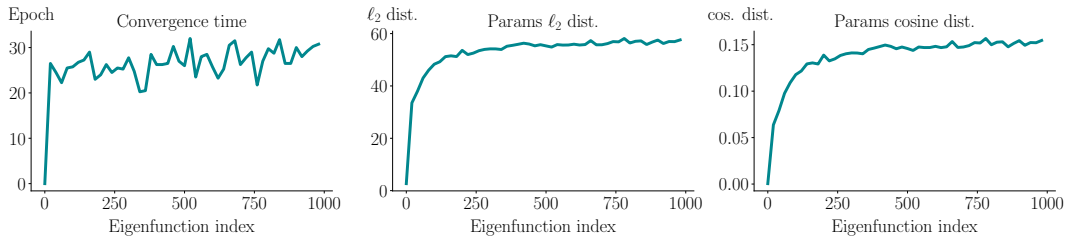

Figure 3: Correlation of different training metrics with the index of the eigenfunction the network is trained on. Plots show the number of training iterations taken by the network to achieve a 0.01 training loss, and the ℓ_2 and cosine distances between initialization and final parameters for a ResNet18 trained on the binarized eigenfunctions of the NTK at initialization.

decreasing eigenvalues for both linearized models – as predicted by the theory – as well as for neural networks. Similarly, Fig. 3 shows how the training dynamics of these networks also correlate with eigenfunction index. Specifically, we see that networks take more time to fit eigenfunctions associated to smaller eigenvalues, and need to travel a larger distance in the weight space to do so.

Overall, our observations reveal that sorting tasks based on their alignment with the NTK is a good predictor of learning complexity both for linear models and neural networks. Nevertheless, we also see that there are large performance gaps between neural networks and their linearized approximations. Indeed, even if the networks and their approximations agree on which eigenfunctions are harder to learn, the kernel methods seem to be much better at it. This is clearly different to what was previously observed for other targets in which neural networks performed better than kernels [18–21], and it highlights that the existence of a *non-linear advantage* is not always true.

3.2 Learning linear predictors

The NTK eigenfunctions are one example of a canonical set of tasks with increasing hardness for kernel methods, whose learning complexity for neural networks follows the same order. However, could there be more examples? And, are the previously observed correlations useful to predict other generalization phenomena? In order, to answer these questions, we propose to analyze another set of problems, but this time using a sequence of targets of increasing complexity for neural networks.

In this sense, it has recently been observed that, for convolutional neural networks (CNNs), the set of linear predictors – i.e., hyperplanes separating two distributions – represents a function class with a wide range of learning complexities among its elements [30]. In particular, it has been confirmed empirically that it is possible to rank the complexity for a neural network to learn different linearly separable tasks based on a sequence of vectors known as neural anisotropy directions (NADs) [30].

Definition (Neural anisotropy directions)**.** *The NADs of a neural network are the ordered sequence of orthonormal vectors* $\{\boldsymbol{v}_j\}_{j=1}^d$ *which form a full basis of the input space and whose order is determined by the sample complexity required to learn the linear predictors* $\{\boldsymbol{x} \mapsto \mathrm{sign}(\boldsymbol{v}_j^\top \boldsymbol{x})\}_{j=1}^d$.

In [30], the authors provided several heuristics to compute the NADs of a neural network. However, we now provide a new, more principled, interpretation of the NADs, showing one can also obtain this sequence using a kernel approximation. To that end, we will make use of the following theorem.

Theorem 1. *Let* $\boldsymbol{u} \in \mathbb{S}^{d-1}$ *be a unitary vector that parameterizes a linear predictor* $g_{\boldsymbol{u}}(\boldsymbol{x}) = \mathrm{sign}(\boldsymbol{u}^\top \boldsymbol{x})$*, and let* $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$*. The alignment of* $g_{\boldsymbol{u}}$ *with* $\boldsymbol{\Theta}$ *is given by*

$$\alpha(g_{\boldsymbol{u}}) = \left\| \mathbb{E}_{\boldsymbol{x}} \left[\nabla^2_{\boldsymbol{x},\boldsymbol{\theta}} f_{\boldsymbol{\theta}_0}(\boldsymbol{x}) \right] \boldsymbol{u} \right\|_2^2, \tag{4}$$

where $\nabla^2_{\boldsymbol{x},\boldsymbol{\theta}} f_{\boldsymbol{\theta}_0}(\boldsymbol{x}) \in \mathbb{R}^{n \times d}$ *denotes the derivative of* $f_{\boldsymbol{\theta}}$ *with respect to the weights and the input.*
Proof See supp. material.

Theorem 1 gives an alternative method to compute NADs. Indeed, in the kernel regime, the NADs are simply the right singular vectors of the matrix of mixed-derivatives $\mathbb{E}_{\boldsymbol{x}} \nabla^2_{\boldsymbol{x},\boldsymbol{\theta}} f_{\boldsymbol{\theta}_0}(\boldsymbol{x})$ of the network[2] Note however, that this interpretation is just based on an approximation, and hence there is no *explicit* guarantee that these NADs will capture the direcional inductive bias of deep networks. Our

[2] All predictors have the same L_2 norm. Hence, their alignment is inversely proportional to their kernel norm.

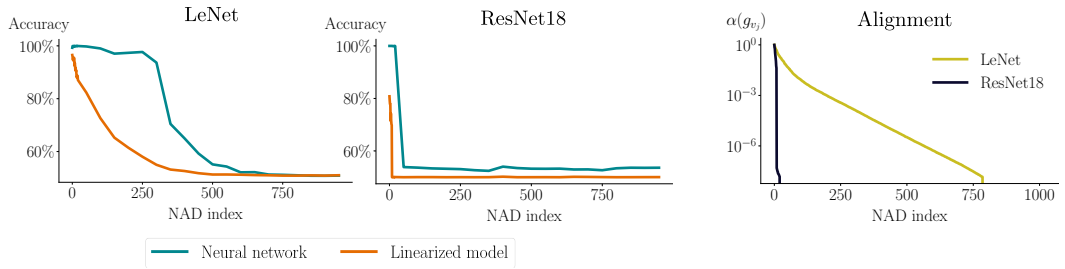

Figure 4: (Left) Performance comparison of different neural network architectures with their linearizations when learning linear target functions aligned with increasing NADs, i.e., $\boldsymbol{x} \mapsto \text{sign}(\boldsymbol{v}^\top \boldsymbol{x})$. (Right) Predicted value of the alignment of the predictors with $\boldsymbol{\Theta}$, i.e., $\alpha(g_{\boldsymbol{v}_j})$ (see Theorem 1)

experiments show otherwise, as they reveal that CNNs actually rank the learning complexity of different linear predictors in a way compatible with Theorem 1.

Indeed, as shown in Fig. 4, when trained to classify a set of linearly separable datasets[3], aligned with the NTK-predicted NADs, CNNs perform better on those predictors with a higher kernel alignment (i.e., corresponding to the first NADs) than on those with a lower one (i.e., later NADs). The fact that NADs can be explained using kernel theory constitutes another clear example that theory derived from a naïve linear expansion of a neural network can sometimes capture important trends in the inductive bias of deep networks; even when we observe a clear performance gap between linear and non-linear models. Surprisingly, neural networks exhibit a strong *non-linear advantage* on these tasks, even though these NADs were explicitly constructed to be well-aligned with the linear models.

Overall, our results explain why previous heuristics that used the NTK to rank the complexity of learning certain tasks [10, 11, 13] were successful in doing so. Specifically, by systematically evaluating the performance of neural networks on tasks of increasing complexity for their linearized approximations, we have observed that the non-linear dynamics on these networks do not change the way in which they sort the complexity of these problems. However, we should not forget that the differences in performance between the neural networks and their approximation are very significant and whether they favour or not neural networks depends on the task. In what follows, we thus delve deeper into the sources of these differences.

4 Sources of the non-linear advantage

In this section, we study in more detail the mechanisms that separate neural networks from their linear approximations, which can sometimes lead to significant performance differences. Specifically, we show that there are important nuances involved in the comparison of linear and non-linear models, which depend on number of samples, architecture or target task. To shed more light on this complex relations, we conduct a fine-grained analysis of the optimization dynamics of neural networks, studying the evolution of their empirical NTK. We show that due to the non-linear optimization dynamics of deep networks, the alignment of their NTK with the training task grows with training. This is an important phenomenon which can explain why, as it was previously observed but not explained in [21], there is a better agreement between the predictions of neural networks and those obtained by linear approximations after some epochs of training. Finally, we show that the kernel adaptation is a major driver of the faster speed of convergence of neural networks with respect to kernel methods. Nevertheless, we also show that this adaptation can sometimes be imperfect, and lead neural networks to overfit the training labels.

4.1 The non-linear advantage depends on the sample size

As we have seen, there exist multiple problems in which neural networks perform significantly better than their linearized approximations (see Sec. 3.2), but also others where they do not (see Sec. 3.1). We now show, however, that the magnitude of these differences is influenced by the training set size.

[3]Full details of the experiment can be found in the supp. material.

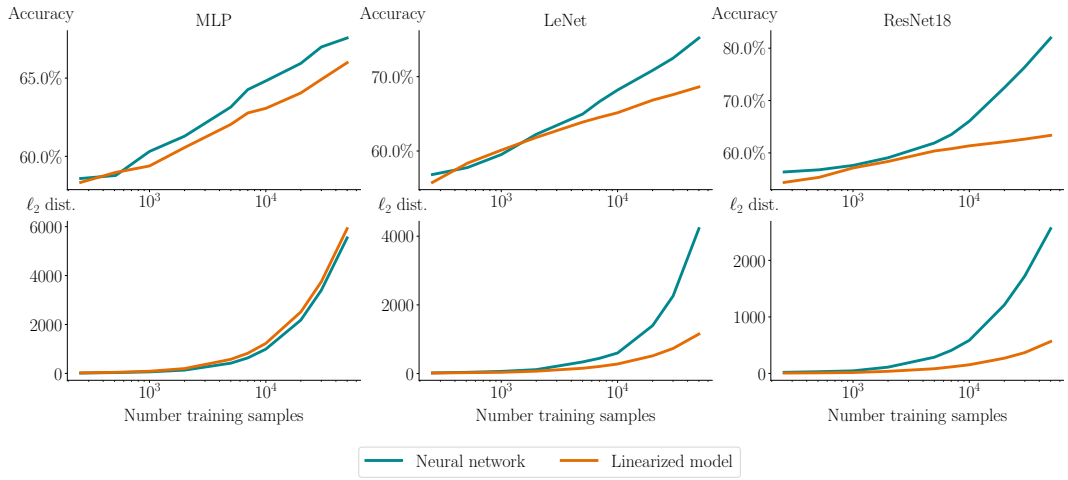

Figure 5: Comparison of test accuracy (top) and parameter distance to initialization (bottom) between neural networks and their linear approximations trained on CIFAR2 with different training set sizes. Plots show average over five different random seeds.

We can illustrate this phenomenon by training several neural networks to predict some semantically-meaningful labels of an image dataset. In particular, and for consistency with the previous two-class examples, we deal with a binary version of the popular CIFAR10 dataset, in which we assign label $+1$ to all samples from the first five classes in the dataset, and label -1 to the rest. We will refer to this version of the dataset as CIFAR2. Indeed, as seen in Fig. 5, some neural networks exhibit a large non-linear advantage on this task, but this advantage mostly appears when training on larger datasets. This phenomenon suggests that the inductive bias that boosts neural network's performance with respect to kernel methods involves an important element of scale.

One can intuitively understand this behaviour by analyzing the distance travelled by the parameters during optimization (see bottom row of Fig. 5). Indeed, for smaller training set sizes, the networks can find solutions that fit the training data closer to their initialization more easily. And, as a result, the error incurred by the linear approximation in these cases is smaller. This suggests why previous studies could not identify large performance gaps between NTK-based models and neural networks in small-data tasks [16], while it also highlights the strength of the linear approximation in this regime.

4.2 The kernel alignment with the target increases during training

So far we have mostly analyzed results dealing with linear expansions around the weight initialization $\boldsymbol{\theta}_0$. However, recent empirical studies have argued that linearizing at later stages of training induces smaller approximation errors [21], suggesting that the NTK dynamics can better explain the final training behaviour of a neural network. To the best of our knowledge, this phenomenon is still poorly understood, mostly because it hinges on understanding the non-linear dynamics of deep networks. We now show, however, that understanding the way the spectrum of the NTK evolves during training can provide important insights into these dynamics.

To that end, we first analyze the evolution of the principal components of the empirical NTK in relation to the target function. Specifically, let $\boldsymbol{\Phi}_t$ denote the matrix of first K eigenvectors of the Gram matrix of $\boldsymbol{\Theta}_t$ obtained by linearizing the network after t epochs of training, and let $\boldsymbol{y} \in \mathbb{R}^m$ be the vector of training labels. In Sec. 3.1, we have seen that both neural networks and their linear approximations perform better on targets aligned with the first eigenfunctions of $\boldsymbol{\Theta}_0$. We propose, therefore, to track the energy concentration $\|\boldsymbol{\Phi}_t \boldsymbol{y}\|_2 / \|\boldsymbol{y}\|_2$ of the labels onto the first eigenfunctions with the aim to identify a significant transformation of the principal eigenspace $\mathrm{span}(\boldsymbol{\Phi}_t)$.

Fig. 6 shows the result of this procedure applied to a CNN trained to classify CIFAR2. Strikingly, the amount of energy of the target function that is concentrated on the $K = 50$ first eigenfunctions of the NTK significantly grows during training. This is a heavily non-linear phenomenon – by definition the linearized models have a fixed kernel – and it hinges on a dynamical realignment of $\boldsymbol{\Theta}_t$ during training. That is, training a neural network rotates $\boldsymbol{\Theta}_t$ in a way that increases $\|\boldsymbol{\Phi}_t \boldsymbol{y}\|_2 / \|\boldsymbol{y}\|_2$.

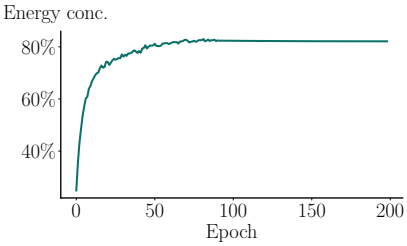

Figure 6: Energy concentration of $10,000$ CIFAR2 training labels on the first $K = 50$ eigenvectors of the kernel Gram matrices $\mathbf{\Phi}_t$ of a LeNet.

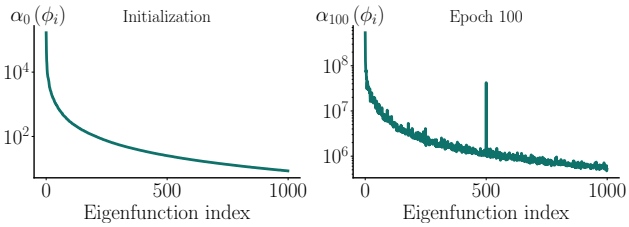

Figure 7: Alignment of the eigenfunctions of the NTK at initialization for a LeNet with the NTKs computed at the beginning (left) and at the end of 100 epochs of training (right) to predict the 500th eigenfunction at initialization.

This phenomenon is prevalent across training setups, and it can also be observed when training on other tasks. In fact, a more fine-grained inspection reveals that the kernel rotation mostly happens in a single functional axis. It maximizes the alignment of $\mathbf{\Theta}_t$ with the target function f, i.e., $\alpha_t(f) = \|\mathbb{E}_{\boldsymbol{x}}\left[f(\boldsymbol{x})\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_t}(\boldsymbol{x})\right]\|_2^2$, but does not greatly affect the rest of the spectrum. Indeed, our results show that during training α_t grows significantly more for the target than for any other function.

This is clearly illustrated in Fig. 7, where we compare α_0 and α_{100} for the first $1,000$ eigenfunctions of $\mathbf{\Theta}_0$, $\{\phi_j\}_{j=1}^{1,000}$, when training to predict and arbitrary eigenfunction[4] $\boldsymbol{x} \mapsto \text{sign}(\phi_{500}(\boldsymbol{x}))$. Strikingly, we can see that after training to predict ϕ_{500}, $\alpha(\phi_{500})$ increases much more than $\alpha(\phi_j)$ for any other eigenfunction ϕ_j. In fact, the relative alignment between all other eigenfunctions does not change much. Note, however, that the absolute values of all alignments have also grown, a phenomenon which is due to a general increase in the Jacobian norm $\mathbb{E}_{\boldsymbol{x}}\|\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_t}(\boldsymbol{x})\|_2$ during training.

These results show that there is great potential in using linear approximations to investigate important deep learning phenomena involving pretrained networks as in [10, 17]. Indeed, as $\alpha_t(f)$ is higher at the end of training, it can be expected that a linear expansion at this late stage will be able to capture the inductive bias needed to fine-tune on targets similar to f. The intuition behind this lies in the geometry of the NTK RKHS $\mathcal{H}$: Note that in $\mathcal{H}$, $\|\hat{f}_{\boldsymbol{\theta}}\|_{\mathbf{\Theta}}$=$\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|_2$ [3]; and recall that, as indicated by Lemma 1, target functions with small $\|f\|_{\mathbf{\Theta}}$ also have high $\alpha(f)$. This means that in $\mathcal{H}$ those tasks with a high $\alpha(f)$ are indeed represented by weights closer to the origin of the linearization, which thus makes the approximation error of the linearization smaller for these targets as observed in [21].

4.3 Kernel rotation improves speed of convergence, but can hurt generalization

The rotation of $\mathbf{\Theta}_t$ during training is an important mechanism that explains why the NTK dynamics can better capture the behaviour of neural networks at the end of training. However, it is also fundamental in explaining the ability of neural networks to quickly fit the training data. Specifically, it is important to highlight the stark contrast, at a dynamical level, between linear models and neural networks. Indeed, we have consistently observed across our experiments that neural networks converge much faster to solutions with a near-zero training loss than their linear approximations.

We can explain the influence of the rotation of the NTK on this phenomenon through a simple experiment. In particular, we train three different models to predict another arbitrary eigenfunction $\boldsymbol{x} \mapsto \text{sign}(\phi_{400}(\boldsymbol{x}))$: i) a ResNet18, ii) its linearization around initialization, and iii) an unbiased linearization around the solution of the ResNet18 $\boldsymbol{\theta}^\star$, i.e., $\hat{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) = (\boldsymbol{\theta} - \boldsymbol{\theta}_0)^\top \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}^\star}(\boldsymbol{x})$.

Fig. 8 compares the dynamics of these models, revealing that the neural network indeed converges much faster than its linear approximation. We see, however, that the kernelized model constructed using the pretrained NTK of the network has also a faster convergence. We conclude, therefore, that, since the difference between the two linear models only lies on the kernel they use, it is indeed the transformation of the kernel through the non-linear dynamics of the neural network training that makes these models converge so quickly.

Note, however, that the rapid adaptation of $\mathbf{\Theta}_t$ to the training labels can have heavy toll in generalization, i.e., the model based on the pretrained kernel converges much faster than the randomly

[4] Similar results for other eigenfunctions and networks can be found in the supp. material

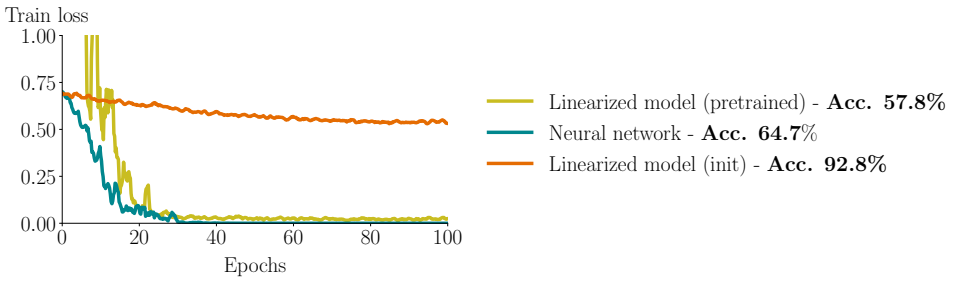

Figure 8: (Left) Evolution of training loss while learning $\boldsymbol{x} \mapsto \text{sign}(\phi_{400}(\boldsymbol{x}))$ for a ResNet18 and two linearized models based on $\boldsymbol{\Theta}_0$ and $\boldsymbol{\Theta}_{100}$ of the ResNet18. (Right) Test accuracy on $\boldsymbol{x} \mapsto \text{sign}(\phi_{400}(\boldsymbol{x}))$ for the three models.

initialized one, but has a much lower test accuracy (comparable to the one of the neural network). On other hand, on CIFAR2 (see Sec. 4.1) the kernel rotation does greatly improve test accuracy for some models. The fact that it can both boost and hurt generalization, highlights that the kernel rotation is subject to its own form of inductive bias to govern its dynamics. In this sense, we believe that explaining the non-trivial coupling between the kernel rotation, alignment, and training dynamics is an important avenue for future research, which will allow us to better understand deep networks.

5 Final remarks

Explaining generalization in deep learning [1] has been the subject of extensive research in recent years [31–35]. The NTK theory is part of this trend, and it has been used to prove convergence and generalization of very wide networks [2, 4–9]. Most of these efforts, however, still cannot explain the behavior of the models used in practice. Therefore, multiple authors have proposed to study neural networks empirically, with the aim to identify novel deep learning phenomena which can be later explained theoretically [21, 36–38].

In this work, we have followed a similar approach, and presented a systematic study comparing the behaviour of neural networks and their linear approximations on different tasks. Previous studies had shown there exist tasks that neural networks can solve but kernels cannot [18–21]. Our work complements those results, and provides examples of tasks where kernels perform better than neural networks (see Sec.3.1). This is important because in machine learning there is *no-free lunch*. Hence, any sufficiently powerful model for a set of problems must also be limited on other tasks [39]. Moving forward, knowing which tasks a neural network can and cannot solve efficiently will be fundamental to explain their inductive bias.

Similarly, our work complements the work in [21] where the authors also empirically compared neural networks and their linear approximations, but focused on a single training task. In this work, we built from these observations and studied models on a more diverse set of problems. Doing so, we have shown that the alignment with the empirical NTK can rank the learning complexity of certain tasks for neural networks, despite being agnostic to the non-linear dis/advantages. In this sense, we have revealed that important factors such as sample size, architecture, or target task can greatly influence the gap between kernelized models and neural networks. Finally, we have seen that the evolution of the NTK during training increases its alignment with the target task, thus explaining important phenomena including some of the previous observations from [21].

Finally, a similar phenomenon to the kernel rotation described in Sec. 4.2 was recently observed for small neural networks, suggesting networks adapt to the main directions of variability of the data manifold [19, 26]. Our work goes one step further and reveals that the kernel moves in the direction that maximizes the alignment with the training task. We confirmed this effect is prevalent on all sorts of models and training tasks, it improves training speed, but it is also subject to its own inductive bias.

Future work should focus on providing a better theoretical understanding to our empirical findings, and try to explain the intricate connections between the evolution of the NTK and the dynamics of training. From a practical perspective, our work paves the way for new applications that use linearized networks to predict generalization in deep learning, or exploit their more predictable inductive bias.

References

[1] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *International Conference on Learning Representations (ICLR)*, 2017.

[2] A. Jacot, C. Hongler, and F. Gabriel, "Neural tangent kernel: Convergence and generalization in neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[3] B. Schölkopf and A. J. Smola, *Learning with Kernels: support vector machines, regularization, optimization, and beyond*. Adaptive computation and machine learning series, MIT Press, 2002.

[4] J. Lee, L. Xiao, S. S. Schoenholz, Y. Bahri, R. Novak, J. Sohl-Dickstein, and J. Pennington, "Wide neural networks of any depth evolve as linear models under gradient descent," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[5] S. S. Du, J. D. Lee, H. Li, L. Wang, and X. Zhai, "Gradient descent finds global minima of deep neural networks," in *International Conferenece on Machine learning (ICML)*, 2019.

[6] A. Bietti and J. Mairal, "On the inductive bias of neural tangent kernels," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[7] D. Zou and Q. Gu, "An improved analysis of training over-parameterized deep neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[8] C. Liu, L. Zhu, and M. Belkin, "On the linearity of large non-linear models: when and why the tangent kernel is constant," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[9] S. Arora, S. S. Du, W. Hu, Z. Li, R. Salakhutdinov, and R. Wang, "On exact computation with an infinitely wide neural net," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[10] A. Deshpande, A. Achille, A. Ravichandran, H. Li, L. Zancato, C. Fowlkes, R. Bhotika, S. Soatto, and P. Perona, "A linearized framework and a new benchmark for model selection for fine-tuning," *arXiv:2102.00084 [cs]*, 2021.

[11] L. Zancato, A. Achille, A. Ravichandran, R. Bhotika, and S. Soatto, "Predicting training time without training," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[12] H. Mobahi, M. Farajtabar, and P. L. Bartlett, "Self-Distillation Amplifies Regularization in Hilbert Space," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[13] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng, "Fourier features let networks learn high frequency functions in low dimensional domains," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[14] T. Gebhart, U. Saxena, and P. Schrater, "A Unified Paths Perspective for Pruning at Initialization," *arxiv:2101.10552*, 2021.

[15] G. Bachmann, S. Moosavi-Dezfooli, and T. Hofmann, "Uniform convergence, adversarial spheres and a simple remedy," *arxiv:2105.03491*, 2021.

[16] S. Arora, S. S. Du, Z. Li, R. Salakhutdinov, R. Wang, and D. Yu, "Harnessing the power of infinitely wide deep nets on small-data tasks," in *International Conference on Learning Representations (ICLR)*, 2020.

[17] W. Maddox, S. Tang, P. G. Moreno, A. G. Wilson, and A. Damianou, "Fast adaptation with linearized neural networks," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2021.

[18] Z. Allen-Zhu and Y. Li, "What can resnet learn efficiently, going beyond kernels?," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[19] B. Ghorbani, S. Mei, T. Misiakiewicz, and A. Montanari, "When do neural networks outperform kernel methods?," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[20] E. Malach, P. Kamath, E. Abbe, and N. Srebro, "Quantifying the Benefit of Using Differentiable Learning over Tangent Kernels," *arXiv:2103.01210 [cs, stat]*, 2021.

[21] S. Fort, G. K. Dziugaite, M. Paul, S. Kharaghani, D. M. Roy, and S. Ganguli, "Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[22] P. L. Bartlett and S. Mendelson, "Rademacher and gaussian complexities: Risk bounds and structural results," in *Computational Learning Theory, 14th Annual Conference on Computational Learning Theory, COLT 2001 and 5th European Conference on Computational Learning Theory, EuroCOLT 2001, Amsterdam, The Netherlands, July 16-19, 2001, Proceedings*, 2001.

[23] R. Novak, L. Xiao, J. Hron, J. Lee, A. A. Alemi, J. Sohl-Dickstein, and S. S. Schoenholz, "Neural tangents: Fast and easy infinite neural networks in python," in *International Conference on Learning Representations (ICLR)*, 2020.

[24] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018.

[25] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," tech. rep., University of Toronto, 2009.

[26] J. Paccolat, L. Petrini, M. Geiger, K. Tyloo, and M. Wyart, "Geometric compression of invariant manifolds in neural nets," *Journal of Statistical Mechanics: Theory and Experiment*, no. 4, 2021.

[27] F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain," *Psychological Review*, no. 6, 1958.

[28] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, no. 11, 1998.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 2016.

[30] G. Ortiz-Jimenez, A. Modas, S.-M. Moosavi-Dezfooli, and P. Frossard, "Neural Anisotropy Directions," in *Advances in Neural Information Processing Systems 34 (NeurIPS)*, 2020.

[31] B. Neyshabur, Z. Li, S. Bhojanapalli, Y. LeCun, and N. Srebro, "The role of over-parametrization in generalization of neural networks," in *International Conference on Learning Representations (ICLR)*, 2019.

[32] D. Soudry, E. Hoffer, M. S. Nacson, and N. Srebro, "The implicit bias of gradient descent on separable data," in *International Conference on Learning Representations (ICLR)*, 2018.

[33] S. Gunasekar, J. D. Lee, D. Soudry, and N. Srebro, "Characterizing implicit bias in terms of optimization geometry," in *International Conferenece on Machine learning (ICML)*, 2018.

[34] P. L. Bartlett, N. Harvey, C. Liaw, and A. Mehrabian, "Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks," *Journal Machine Learning Research*, vol. 20, pp. 63:1–63:17, 2019.

[35] E. Abbe and C. Sandon, "On the universality of deep learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[36] V. Nagarajan and J. Z. Kolter, "Uniform convergence may be unable to explain generalization in deep learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[37] P. Nakkiran, B. Neyshabur, and H. Sedghi, "The deep bootstrap framework: Good online learners are good offline generalizers," in *International Conference on Learning Representations (ICLR)*, 2021.

[38] H. Maennel, I. M. Alabdulmohsin, I. O. Tolstikhin, R. J. N. Baldock, O. Bousquet, S. Gelly, and D. Keysers, "What do neural networks learn when trained with random labels?," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[39] D. H. Wolpert, "The lack of A priori distinctions between learning algorithms," *Neural Computation*, vol. 8, no. 7, 1996.

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] See discussion at the end of Sec. 3 and Sec. 4, as well as our Final remarks.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] Our paper deals with fundamental questions regarding our understanding of deep networks. In this sense, it is subject to the same ethical concerns as the machine learning field as a whole, which makes it hard to identify potential direct risks or benefits associated to our empirical findings.
   (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]
   (b) Did you include complete proofs of all theoretical results? [Yes] See supp. material.
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)? [No] The code will be realeased upon acceptance. However, if a reviewer deems it necessary we would be open to provide the code that supports our claims.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Full details of all our experiments can be found in the supp. material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] Computing the kernel Gram matrix for different random seeds is a computationally expensive procedure (see supp. material) which with our limited resources would have taken too much time to compute for all the settings in our experiments. However, because none of our claims depend on precise metrics, and are rather supported by clear trends which we validated on different settings, we are confident that our results are reproducible.
   (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] Seep supp. material.
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]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
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]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]