A FAST, WELL-FOUNDED APPROXIMATION TO THE EMPIRICAL NEURAL TANGENT KERNEL

Anonymous authors
Paper under double-blind review

ABSTRACT

Empirical neural tangent kernels (eNTKs) can provide a good understanding of a given network's representation: they are often far less expensive to compute and applicable more broadly than infinite-width NTKs. For networks with O output units (e.g. an O-class classifier), however, the eNTK on N inputs is of size $NO \times NO$, taking $\mathcal{O}\left((NO)^2\right)$ memory and up to $\mathcal{O}\left((NO)^3\right)$ computation. Most existing applications have therefore used one of a handful of approximations yielding $N \times N$ kernel matrices, saving orders of magnitude of computation, but with limited to no justification. We prove that one such approximation, which we call "sum of logits," converges to the true eNTK at initialization. Our experiments demonstrate the quality of this approximation for various uses across a range of settings.

1 INTRODUCTION

The pursuit of a theoretical foundation for deep learning has lead researches to uncover interesting connections between neural networks (NNs) and kernel methods. It has long been known that randomly initialized NNs in the infinite width limit are Gaussian processes with what is termed the NNGP kernel, and training the last layer with gradient flow under squared loss corresponds to the posterior mean (Neal, 1996; Williams, 1996; Hazan & Jaakkola, 2015; Lee et al., 2017; Matthews et al., 2018; Novak et al., 2018; Yang, 2019). More recently, Jacot et al. (2018) (building off a line of closely related prior work) showed that the same is true if we train all the parameters of the network, but using a different kernel called the Neural Tangent Kernel (NTK). Yang (2020); Yang & Littwin (2021) later showed that this connection is architecturally universal, extending the domain from fully-connected NNs to most of the currently-used networks in practice, such as ResNets and Transformers. Lee et al. (2019) also showed that the dynamics of training wide but finite-width NNs with gradient descent can be approximated by a linear model obtained from the first-order Taylor expansion of that network around its initialization. Furthermore, they experimentally showed that this approximation approximation excellently holds even for networks that are not so wide.

In addition to theoretical insights from the results themselves, NTKs have had significant impact in diverse practical settings. Arora et al. (2019b) show very strong performance of NTK-based models on a variety of low-data classification and regression tasks. The condition number of an NN's NTK has been shown correlation directly with the trainability and generalization capabilities of the NN (Xiao et al., 2018; 2020); thus, Park et al. (2020); Chen et al. (2021) have used this to develop practical algorithms for neural architecture search. Wei et al. (2022); Bachmann et al. (2022) estimate the generalization ability of a specific network, randomly initialized or pre-trained on a different dataset, with efficient cross-validation. Zhou et al. (2021) use NTK regression for efficient meta-learning, and Wang et al. (2021); Holzmüller et al. (2022); Mohamadi et al. (2022) use NTKs for active learning.

There has also been significant theoretical insight gained from empirical studies of networks' NTKs. Here are a few examples: Fort et al. (2020) use NTKs to study how the loss geometry the NN evolves under gradient descent. Franceschi et al. (2021) employ NTKs to analyze the behaviour of Generative Adverserial Networks (GANs). Nguyen et al. (2020; 2021) used NTKs for dataset distillation. He et al. (2020); Adlam et al. (2020) used NTKs to predict and analyze the uncertainty of a NN's predictions. Tancik et al. (2020) use NTKs to analyze the behaviour of MLPs in learning high frequency functions, leading to new insights into our understanding of neural radiance fields.

We thus believe NTKs will continue to be used in both theoretical and empirical deep learning.

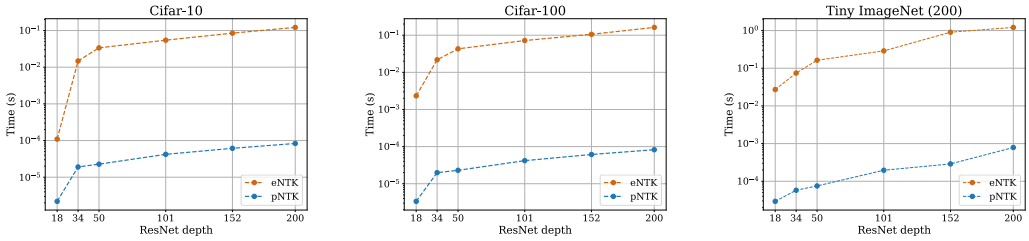

Figure 1: Comparison of **wall-clock time of evaluating the eNTK and pNTK of a pair of input datapoints** over various datasets and ResNet depths.

Unfortunately, however, computing the NTK for practical networks is extremely challenging, and most of the time not even computationally feasible. The NTK of a NN is defined as the outer product of the Jacobians of the output of the NN with respect to its parameters:

$$eNTK \coloneqq \Theta_\theta(x_1, x_2) = \left[J_\theta\left(f_\theta(x_1)\right)\right]\left[J_\theta\left(f_\theta(x_2)\right)\right]^\top, \tag{1}$$

where $J_\theta\left(f_\theta(x)\right)$ denotes the Jacobian of the function f at a point x with respect to the flattened vector of all its parameters, $\theta \in \mathbb{R}^P$. Assuming $f : \mathbb{R}^D \to \mathbb{R}^O$, where D is the input dimension and O the number of outputs, we have $J_\theta\left(f_\theta(x)\right) \in \mathbb{R}^{O \times P}$ and $\Theta_\theta(x_1, x_2) \in \mathbb{R}^{O \times O}$. Thus, computing the NTK between a set of N_1 data points and a set of N_2 data points yields $N_1 N_2$ matrices each of shape $O \times O$, which we usually reshape into an $N_1 O \times N_2 O$ matrix.

When computing an eNTK on tasks involving large datasets and with multiple output neurons, e.g. in a classification model with O classes, the eNTK quickly becomes impractical regardless of how fast each entry is computed due to its $NO \times NO$ size. For instance, the full eNTK of a classification model even on the relatively mild CIFAR-10 dataset (Krizhevsky, 2009), stored in double precision, takes over 1.8 terabytes in memory. For practical usage, we need to do something better.

In this work, we present a simple trick for a strong approximation of the eNTK that removes the O^2 from the size of the kernel matrix, resulting in a factor of O^2 improvement in the memory and up to O^3 in computation. Since for typical classification datasets O is at least 10 (e.g. CIFAR-10) and potentially 1 000 or more (e.g. ImageNet, Deng et al., 2009), this provides multiple orders of magnitude savings over the original eNTK (1). We prove that under appropriate initialization of the NN this approximation converges to the original eNTK at a rate of $\mathcal{O}(n^{-1/2})$ for a network of depth L and width n in each layer, and the predictions of kernel regression with the approximate kernel do the same. Finally, we present diverse experimental investigations supporting our theoretical results across a range of different architectures and settings. We hope this approximation further enables researches to employ NTKs towards theoretical and empirical advances in wide networks.

Infinite NTKs In the infinite-width limit of properly initialized NNs, Θ_θ converges almost surely at initialization to a particular kernel, and remains constant over the course of training. Algorithms are available to compute this expectation exactly, but they tend to be substantially more expensive than computing (1) directly for all but extremely wide networks. The convergence to this infinite-width regime is also slow in practice, and moreover it eliminates some of the interest of the framework: neural architecture search, predicting generalization of a pre-trained representation, and meta-learning are all considerably less interesting when we only consider infinite-width networks that do essentially no feature learning. Thus, in this paper, we focus only on the "empirical" eNTK (1).

2 Related Work

Among the numerous recent works that have used eNTKs either to gain insights about various phenomenons in deep learning or to propose new algorithms, not many have publicized the computational costs and implementation details of computing eNTKs. Nevertheless, all are in agreement about the expense of such computations (Park et al., 2020; Holzmüller et al., 2022; Fort et al., 2020).

Several recent works have, mostly "quietly," employed various techniques to avoid dealing with the full eNTK matrix; however, to the best of our knowledge, none provide any rigorous justifications.

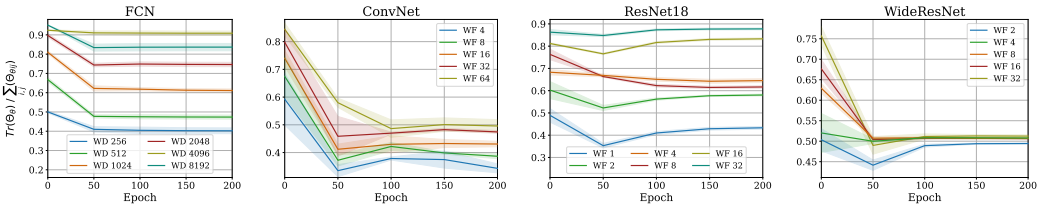

Figure 2: Evaluating the **ratio of information between on-diagonal and off-diagonal elements of** Θ_θ at initialization and throughout training, based on 1000 points from CIFAR-10. The reported numbers are the average of 1000×1000 kernels each having a shape of 10×10. The same subset has then been used to train the NN using SGD. As the NN's width grows, the eNTK converges to being diagonal at initialization among all different architectures.

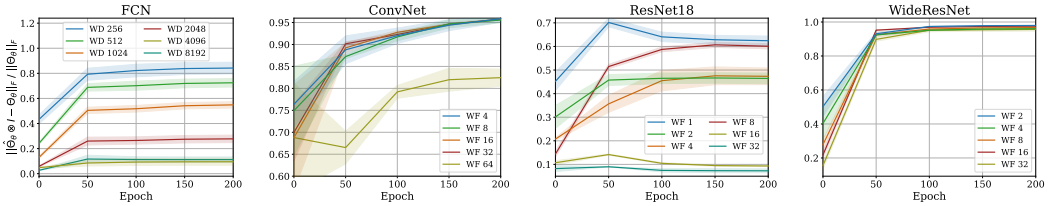

Figure 3: Evaluating the **relative difference of Frobenius norm of** $\Theta_\theta(\mathcal{D}, \mathcal{D})$ **and** $\hat{\Theta}_\theta(\mathcal{D}, \mathcal{D}) \otimes I_O$ at initialization and throughout training, based on 1000 points from CIFAR-10. As the NN's width grows, the relative difference in $\|\Theta_\theta\|_F$ and $\|\hat{\Theta}_\theta \otimes I_O\|_F$ decreases at initialization.

Wei et al. (2022, Section 2.3) point out that if the final layer of a NN is randomly initialized, the *expected* eNTK can be written as $K_0 \otimes I_C$ for some kernel K_0, where I_O is the $O \times O$ identity matrix and $\otimes$ is the Kronecker product. Thus, they use the approximation in which they only compute the eNTK with respect to one of the logits of the NN. Although their approach to approximating the eNTK is similar to ours, they don't provide any rigorous bounds or empirical study of how closely the actual $eNTK$ is approximated by its expectation in this regard. Wang et al. (2021) employs the same "single-logit" strategy, though they only mention the infinite-width limit as a motivation supporting their trick. Despite of these claims, we will see in our experiments that the eNTK is generally *not* diagonal. We will, however, prove upper bounds on distance of our approximation to the eNTK, and provide experimental support that this approximation captures the behaviour of the eNTK even when the NN's weights are not at initialization. Park et al. (2020); Chen et al. (2021) also seem to use a form of "single-logit" approximation to eNTK, without explicitly mentioning it. Lee et al. (2019), by contrast, do use the full eNTK, and hence never compute the kernel on more than 256 datapoints.

More recently, Novak et al. (2021) performed an in-depth analysis of computational and memory complexity required for computing the eNTK, and proposed two new approaches to reduce the time complexity of computing the eNTK in special cases (depending on the architecture of the NN) over explictly implementing (1). We remark that as our approaches are complementary, as we propose an approximation to the eNTK and Novak et al. (2021) proposes an algorithm for computation of the eNTK, which also applies when to computing our approximation – in fact, we use their "structured derivatives" method in our experiments. Moreover, their approach does not reduce the memory complexity of computing the eNTK, typically the largest burden in practical applications.

3 Pseudo-NTK

We define the pseudo-NTK (pNTK) of the network f_θ as

$$pNTK := \hat{\Theta}_\theta(x_1, x_2) = \underbrace{\left[\nabla_\theta \left(\frac{1}{\sqrt{O}} \sum_{i=1}^{O} f_\theta^{(i)}(x_1)\right)\right]}_{1 \times P} \underbrace{\left[\nabla_\theta \left(\frac{1}{\sqrt{O}} \sum_{i=1}^{O} f_\theta^{(i)}(x_2)\right)\right]^\top}_{P \times 1}, \tag{2}$$

where $f_\theta^{(i)}(x)$ denotes the i'th output of f_θ on the input x. Unlike eNTK, which is a matrix-valued kernel for each pair of inputs, pNTK is a traditional scalar-valued kernel.

Some recent work (Arora et al., 2019a; Yang, 2020; Wei et al., 2022; Wang et al., 2021) has pointed out that in the infinite width limit the NTK becomes a diagonal matrix. Thus, one can avoid computing the off-diagonal entries of the infinite-width NTK of each pair through using $\Theta_\theta(x_1, x_2) = \hat{\Theta}_\theta(x_1, x_2) \otimes I_O$, resulting in a drastic $\mathcal{O}(O^2)$ time and memory complexity decrease.

Practitioners have accordingly used the same approach in computing the corresponding eNTK of a finite width network, but with little to no further justification. We see in our experiments that for finite width networks, the NTK is **not** diagonal. In fact, we show that for most practical networks, it is very far from being diagonal, casting doubt on the correctness of previous results that have used this scheme to calculate the NTK of a finite-width matrix. We, however, provide the first theoretical bounds of the difference from our presented approximation (2), which we also call "sum of logits."

Before turning to the formal results and experimental evaluation, we give some intuition. First, suppose that $f_\theta^{(i)}(x) = \phi(x) \cdot v_i$, so that v_i is the ith entry of a linear read-out layer; then $\frac{1}{\sqrt{O}} \sum_{i=1}^{O} f_\theta^{(i)}(x) = \phi(x) \cdot \left[\frac{1}{\sqrt{O}} \sum_{i=1}^{O} v_i\right]$. If the vectors $v_i \sim \mathcal{N}(0, \sigma^2 I_O)$ are independent, then $\frac{1}{\sqrt{O}} \sum_{i=1}^{O} v_i \sim \mathcal{N}(0, \sigma^2 I_O)$ has the same distribution as any individual entry, say v_1. Thus, at initialization, our sum of logits approximation agrees in distribution with the first-logit approximation. We prefer the sum-of-logits form for networks that are not at random initialization (like a pretrained network with randomized read-out layer), however.

Calling this vector (whether the first logit or sum of logits) v, we can think of (2) as the NTK of a model with a single scalar output as a function of ϕ. When we linearize a network with that kernel for an O-class classification problem, getting the formula (6) discussed in Section 4.3, we end up, effectively, with a one-vs-rest classifier scheme. Thus, we can think of the pseudo-NTK as approximating the process of training O one-vs-rest classifiers, rather than a single O-way classifier.

4 Approximation Quality of Pseudo-NTK

We will now study various aspects of the approximation of (2) to (1), both in theory and empirically. To experimentally evaluate our claims, we present various experiments that compare the targeted characteristic of pNTK vs eNTK both at initialization and throughout training. We present our experiments on four widely-used architectures, namely, fully-connected networks (FCN) of depth 3 (as in Lee et al., 2019; 2020), fully-convolutional networks (ConvNet) of depth 8 (as in Arora et al., 2019a;b; Lee et al., 2020), ResNet18 (He et al., 2016) and WideResNet-16-k (Zagoruyko & Komodakis, 2016). We evaluate each architecture at different widths, as mentioned in the plot legends: we show exact widths for FCN, while for others we show a widening factor (details in Appendix A). For consistency with most other recent papers studying NTKs and properties of NNs in general, we chose CIFAR-10 (Krizhevsky, 2009) as the dataset to evaluate our experiments on. Each experiment is repeated using three seeds, and the corresponding error bars are plotted. However, in some figures the error bars are not plotted. In those cases, it is because they interfered with clear interpretation of the plots, as the error bars were too large and not informative. We used Stochastic Gradient Descent (SGD) as the optimizer of choice for training our networks. Details on the configurations used in optimization of each network can be found in the Appendix.

Each network on each experiment is trained for 200 epochs using mini-batch stochastic gradient descent (SGD), run on NVIDIA V100 GPUs with 32GB of memory. Details on the batch sizes and learning rates used for each NN can be found in Appendix A. The measured statistic for each experiment has been reported after 0, 50, 100, 150, and 200 epochs.

4.1 Pseudo-NTK converges to eNTK as the network's width grows

The first crucial thing to verify is whether the pNTK kernel matrix approximates the true eNTK as a whole. We study this first in terms of Frobenius norm.

Theorem 1 (Informal). *Let $f_\theta : \mathbb{R}^D \to \mathbb{R}^O$ be a fully-connected network with layers of width n, parameters initialized as $\mathcal{N}(0, 1/n)$, and ReLU-type activations. Let $\hat{\Theta}_\theta(x_1, x_2)$ be the corresponding pNTK of f_θ as in (2) and $\Theta_\theta(x_1, x_2)$ the corresponding eNTK as in (1) for a fixed pair of inputs x_1,*

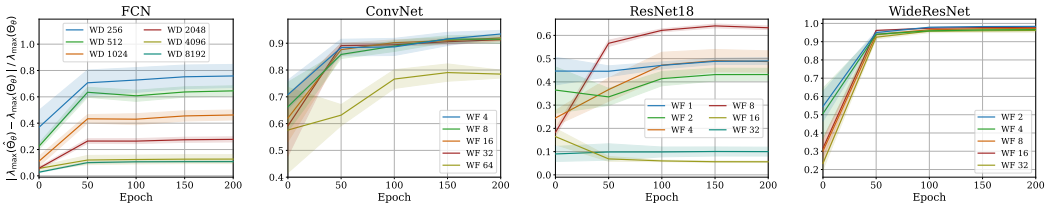

Figure 4: Evaluating the **relative difference of** $\lambda_{\max}$ **of** $\Theta_\theta(\mathcal{D},\mathcal{D})$ **and** $\hat{\Theta}_\theta(\mathcal{D},\mathcal{D})$ at initialization and throughout training, based on kernels on a subset ($|\mathcal{D}| = 1000$) of points from CIFAR-10. As the NN's width grows, the relative difference in $\lambda_{\max}(\Theta_\theta(\mathcal{D},\mathcal{D}))$ and $\lambda_{\max}(\hat{\Theta}_\theta(\mathcal{D},\mathcal{D}))$ decreases.

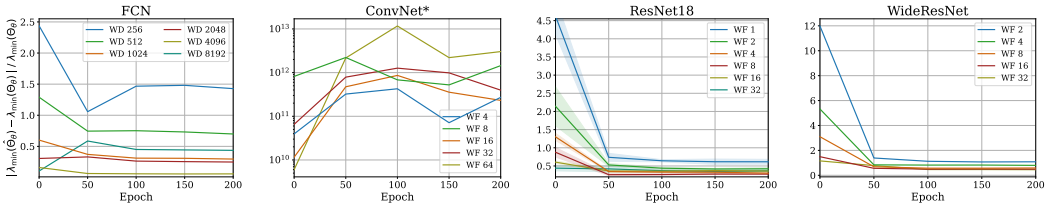

Figure 5: Evaluating the **relative difference of** $\lambda_{\min}$ **of** $\Theta_\theta(\mathcal{D},\mathcal{D})$ **and** $\hat{\Theta}_\theta(\mathcal{D},\mathcal{D})$ at initialization and throughout training. The kernels have been evaluated using a subset ($|\mathcal{D}| = 1000$) of points from CIFAR-10. As the NN's width grows, the relative difference in $\lambda_{\min}(\Theta_\theta(\mathcal{D},\mathcal{D}))$ and $\lambda_{\min}(\hat{\Theta}_\theta(\mathcal{D},\mathcal{D}))$ decreases. Note the extremely large values reported for ConvNet. As observed previously in Lee et al. (2020) and Xiao et al. (2020), CNN-GAP is ill-conditioned and $\lambda_{\min}(\Theta_\theta(\mathcal{D},\mathcal{D})) \to 0$, while $\lambda_{\min}(\hat{\Theta}_\theta(\mathcal{D},\mathcal{D})) > 0.001$, causing the huge discrepancy. More details in Appendix B.3.

x_2. *With high probability over the initialization,*

$$\frac{\|\hat{\Theta}_\theta(x_1,x_2)\otimes I_O - \Theta_\theta(x_1,x_2)\|_F}{\|\Theta_\theta(x_1,x_2)\|_F} \in \mathcal{O}(n^{-\frac{1}{2}}). \tag{3}$$

Remark 2. *All of the results in the paper can be straightforwardly extended to networks with different widths, as long as the consecutive layers' widths satisfy* $n_{l+1} = \Theta(n_l)$*. Moreover, the results can be made architecturally universal with the techniques of Yang (2020); Yang & Littwin (2021).*

Theorem 1 provides the first upper bound on the convergence rate of pNTK towards eNTK. A formal statement for Theorem 1 and its proof are in Appendix B.2.

Remark 3. *Based on the provided proof, it's straightforward that the ratio of information between on-diagonal and off-diagonal elements of the eNTK matrix converges to zero with a rate of* $\mathcal{O}(n^{-\frac{1}{2}})$ *with high probability over random initialization, as depicted in Figure 2.*

Experimental results in Figure 2 support the fact that as width grows, the ratio between off-diagonal and diagonal elements of $\Theta_\theta(x_1,x_2)$ converges to zero. Furthermore, Figure 3 provides experimental support that as width grows, $\hat{\Theta}_\theta \otimes I_O$ converges to Θ_θ in terms of relative Frobenius norm. Note that the result of Theorem 1 only applies to epoch zero of the depicted figures, as the theorem applies only to the networks whose weights are at initialization (in the *NTK parameterization*).

However, as can be seen in the figures, these results don't necessarily apply to the NNs whose readout layers are not at initialization (i.e., after a few epochs of training). This naturally gives rise to the question: *Can the pNTK be used to analyze and represent NNs whose parameters are far from initialization?* We will now take various experimental approaches towards studying this question.

4.2 Pseudo-NTK's spectrum converges to eNTK's as width grows

As discussed before, the conditioning of a network's eNTK has been shown to be closely related to generalization properties of the network, such as trainability and generalization risk (Xiao et al., 2018; 2020; Wei et al., 2022). Thus, we would like to know how well the pNTK's eigenspectrum

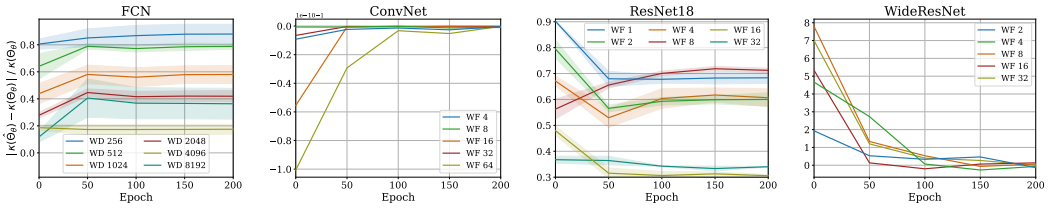

Figure 6: Evaluating the **relative difference of condition number of eNTK and pNTK** at initialization and throughout training. Condition number is defined as $\kappa(K) = \lambda_{\max}(K)/\lambda_{\min}(K)$. The kernels have been evaluated using a subset ($|\mathcal{D}| = 1000$) of points from CIFAR-10. As the NN's width grows, the relative difference in $\lambda_{\min}(\Theta_\theta(\mathcal{D},\mathcal{D}))$ and $\lambda_{\min}(\hat{\Theta}_\theta(\mathcal{D},\mathcal{D}))$ decreases. The bizarre values found in the ConvNet plot can be addressed as discussed in Figure 5; more details in Appendix B.3.

approximates that of the eNTK. The following theorem gives a bound on the rate of convergence between the maximum eigenvalues of the two kernels.

Theorem 4 (Informal)**.** *Let $f_\theta : \mathbb{R}^D \to \mathbb{R}^O$ be a fully-connected network with layers of width n, parameters initialized as $\mathcal{N}(0, 1/n)$, and ReLU-type activations. Let $\hat{\Theta}_\theta(x_1, x_2)$ be the corresponding pNTK of f_θ as in (2) and $\Theta_\theta(x_1, x_2)$ the corresponding eNTK as in (1) for a fixed pair of inputs x_1, x_2. With high probability over the initialization,*

$$\frac{\lambda_{max}\left(\hat{\Theta}_\theta(x_1, x_2) \otimes I_O\right) - \lambda_{max}\left(\Theta_\theta\left(x_1, x_2\right)\right)}{\lambda_{max}\left(\Theta_\theta\left(x_1, x_2\right)\right)} \in \mathcal{O}(n^{-\frac{1}{2}}). \tag{4}$$

Theorem 4 provides an upper bound on the difference between the maximum eigenvalue of pNTK versus the maximum eigenvalue of eNTK based on the NN's width. A formal statement for Theorem 4 and its proof are given in Appendix B.3. Experimental results in Figure 4 support the fact that as width grows, the max eigenvalue of Θ_θ converge to the max eigenvalue of $\hat{\Theta}_\theta$. Figure 5 provides similar experimental support regarding the minimum eigenvalues of $\hat{\Theta}_\theta$ and Θ_θ in terms of relative difference. Together, these two results intuitively imply that the condition number $\kappa(K) = \lambda_{\max}(K)/\lambda_{\min}(K)$ should become similar as width grows; this is also supported by results in Figure 6. Again, it shall be noted that Theorem 4 only applies to the NNs whose widths are at initialization and as in the previous subsection, these results don't necessarily apply to the NNs not at initialization (i.e., after a few epochs of training).

Interestingly, the rate of increase/decrease in the difference between maximum and minimum eigenvalues and the condition numbers between pNTK and eNTK do not have necessarily have a monotonic behaviour as the training goes on. Observing the exact values of $\lambda_{\min}$, $\lambda_{\max}$, and κ for different architectures, widths at initialization and throughout training reveals that in ConvNet, WideResNet and ResNet18 architectures, $\lambda_{\min}$ is close to zero at initialization, but grows during training; the inverse phenomenon is observed with FCNs. Further investigations of these statistics might reveal interesting insights about the behaviour of NNs trained with SGD and the connections between eNTK and trainibility of the architecture.

4.3 Kernel regression using pNTK vs Kernel regression using eNTK

Lee et al. (2019) proved that as a finite NN's width grows, its training dynamics can be well approximated using the first-order Taylor expansion of that NN around its initialization (a *linearized neural network*). Informally, they showed that when f is wide enough, its predictions after being trained using gradient descent with suitably small learning rate on the training set $\mathcal{D}$ can be approximated by those of the linearized network f^{lin}:

$$\underbrace{f^{lin}(x)}_{O\times 1} = \underbrace{f_0(x)}_{O\times 1} + \underbrace{\Theta_0(x,\mathcal{D})}_{O\times NO} \underbrace{\Theta_0(\mathcal{D},\mathcal{D})^{-1}}_{NO\times NO} \underbrace{(\mathcal{Y}_\mathcal{D} - f_0(\mathcal{D}))}_{NO\times 1}, \tag{5}$$

where $\mathcal{Y}_\mathcal{D}$ is the matrix of one-hot labels for the training points $\mathcal{D}$, and Θ_0 is the eNTK of f at initialization f_0. This is simply kernel regression on the training data $\mathcal{D}$ using the kernel Θ_0 and prior

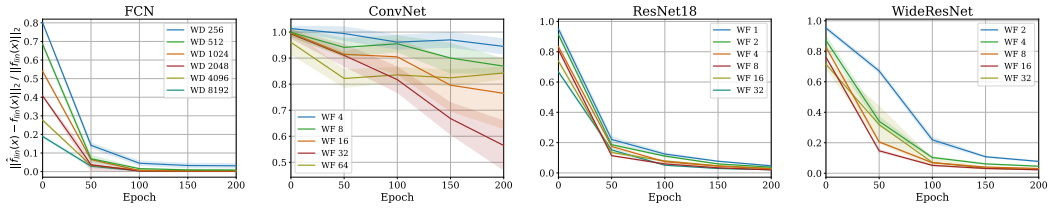

Figure 7: Evaluating the **relative norm difference of kernel regression outputs using eNTK and pNTK as in Equation (5) and Equation (6)** at initialization and throughout training. The kernel regression has been done on $|\mathcal{D}| = 1000$ training points and $|\mathcal{X}| = 500$ test points randomly selected from CIFAR-10's train and test sets. As the NN's width grows, the relative difference in $\hat{f}^{lin}(\mathcal{X})$ and $f^{lin}(\mathcal{X})$ decreases at initialization. Surprisingly, the difference between these two continues to quickly vanish throughout the training process using SGD.

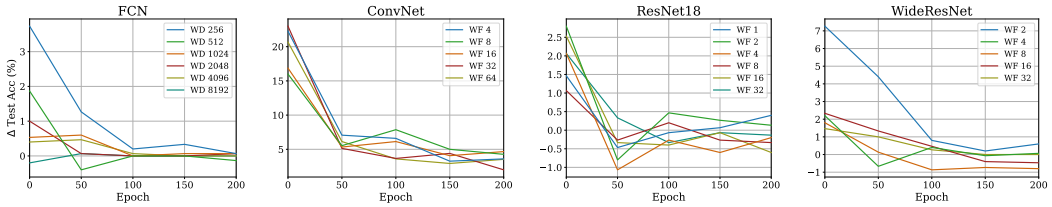

Figure 8: Evaluating the **accuracy difference of kernel regression outputs using eNTK and pNTK as in Equation (5) and Equation (6)** at initialization and throughout training. **Kernel regression using pNTK always achieves a higher test accuracy in comparison to kernel regression using eNTK.** The kernel regression has been done on $|\mathcal{D}| = 1000$ training points and $|\mathcal{X}| = 500$ test points randomly selected from CIFAR-10's train and test sets. As the NN's width grows, the prediction accuracy of in $\hat{f}^{lin}$ and f^{lin} decreases at initialization. Again, the difference between these two continues to vanish throughout the training process using SGD.

mean f_0. Wei et al. (2022) use the same kernel in a generalized cross-validation estimator (Craven & Wahba, 1978) to predict the generalization risk of the NN. As discussed before, using the eNTK in these applications is practically infeasible, due to huge time and memory complexity of the kernel, but we show the pNTK approximates $f^{lin}(x)$ with much improved time and memory complexity.

Theorem 5 (Informal)**.** *Let $f_\theta : \mathbb{R}^D \to \mathbb{R}^O$ be a fully-connected network with layers of width n, parameters initialized as $\mathcal{N}(0, 1/n)$, and ReLU-type activations. Let $\hat{\Theta}_\theta(x_1, x_2)$ be the corresponding pNTK of f_θ as in (2) and $\Theta_\theta(x_1, x_2)$ the corresponding eNTK as in (1) for a fixed pair of inputs x_1, x_2. Define*

$$\underbrace{\hat{f}^{lin}(x)}_{1\times O} := \underbrace{f_0(x)}_{1\times O} + \underbrace{\hat{\Theta}_\theta(x, \mathcal{D})}_{1\times N} \underbrace{\hat{\Theta}_\theta(\mathcal{D}, \mathcal{D})^{-1}}_{N\times N} \underbrace{(\mathcal{Y}_\mathcal{D} - f_0(\mathcal{D}))}_{N\times O}. \tag{6}$$

With proper reshaping, with high probability over random initialization,

$$\|\hat{f}^{lin}(x) - f^{lin}(x)\|_F \in \mathcal{O}(n^{-\frac{1}{2}}). \tag{7}$$

Theorem 5 provides an upper bound on the norm difference of $\hat{f}^{lin}$ from (6) and f^{lin} from Equation (5) based on the NN's last layer's width. A formal statement is given and proved in Appendix B.4.

Experimental results in Figure 7 show that as width grows, the predictions of kernel regression using $\hat{\Theta}_\theta$ converge to the prediction of those obtained from using Θ_θ, *while requiring orders of magnitude of less memory and time to compute*. Figure 8 shows similar results for the difference in prediction accuracies achieved using kernel regression through $\hat{\Theta}_\theta$ and Θ_θ kernels. Appendix B.4 also shows further analysis of how well the *linearized network* predicts the final accuracy of the trained model for each architecture and width pair. Although $\|\hat{f}^{lin}(x) - f^{lin}(x)\|_F$ decreases with width of the network in Figure 7 at initialization, this does not *necessarily* translate to a monotonic behaviour in prediction accuracies, a non-smooth function of the vector of predictions; we do see that the expected pattern more or less holds, however.

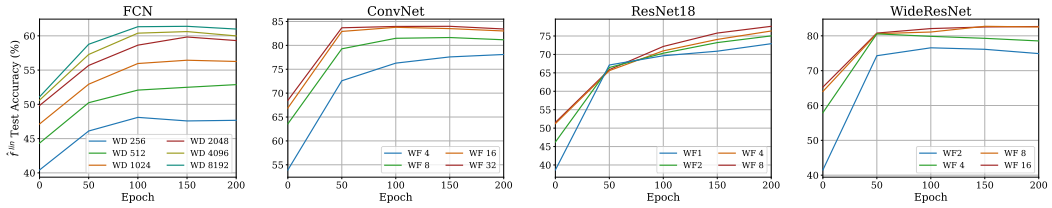

Figure 9: Evaluating the **test accuracy of kernel regression predictions using pNTK as in Equation (6) on the full CIFAR-10 dataset** at initialization and throughout training. As the NN's width grows, the test accuracy of $\hat{f}^{lin}$ is also improved, but eventually saturates with the growing width. Using trained weights in computation of pNTK results in improved test accuracy of $\hat{f}^{lin}$.

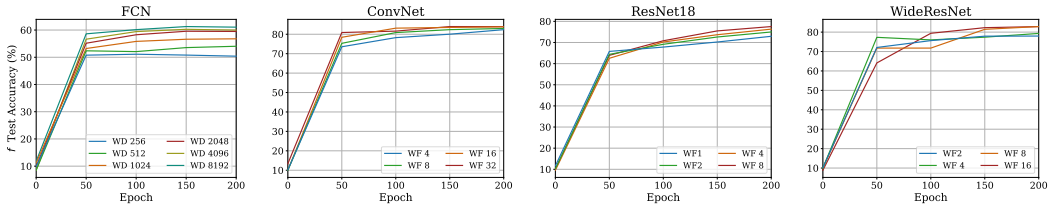

Figure 10: Evaluating the **test accuracy of model f throughout SGD training on the full CIFAR-10 dataset**. In contrast to $\hat{f}^{lin}$, the test accuracy of f does not significantly improve with growing width.

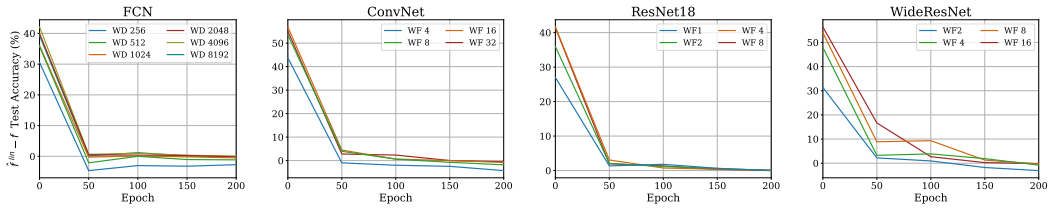

Figure 11: Evaluating the **difference in test accuracy of kernel regression using pNTK as in (6) vs the current model f** throughout SGD training on the full CIFAR-10 dataset: how much does a linearized predictor with the current representation improve prediction accuracy over SGD?

A surprising outcome depicted in Figures 7 and 8 is that while training the model's parameters, predictions of $\hat{f}^{lin}$ and f^{lin} converge very quickly. This is particularly intriguing as it's in contrast with the experimental results depicted in Figures 2 to 4 and 6. In other words, although the kernels Θ_θ and $\hat{\Theta}_\theta \otimes I_O$ seem to be diverging in Frobenius norm, eigenspectrum, and so on, kernel regression using those two kernels converges quickly, so that after 50 epochs the difference in predictions almost totally vanishes. We believe further investigation of why this phenomenon is observed could lead to new interesting insights about the training dynamics of NNs.

5 Kernel Regression using pNTK on full CIFAR-10 dataset

Thanks to the reduction in time and memory complexity of pNTK over eNTK, motivated by Theorem 5 and experimental findings in Figure 8, we finally evaluate the corresponding pNTKs of the four architectures that we have used in our experimental evaluations in different widths using full CIFAR-10 data, at initialization and throughout training the models under SGD. As mentioned previously, running kernel regression with eNTK on all of CIFAR-10 would require evaluating 25×10^{10} Jacobian-vector products and more than ≈ 1.8 terabytes of memory; using pNTK, this can be done with a far more reasonable 25×10^8 Jacobian-vector products and ≈ 18 gigabytes of memory. This is still a heavy load compared to, say, direct SGD training, but is within the reach of standard compute nodes.

Figure 9 shows the test accuracy of $\hat{f}^{lin}$ on the full train and test sets of CIFAR-10. In the infinite-width limit, the test accuracy of $\hat{f}^{lin}$ at initialization (and later, because the kernel stays constant in this regime) should match the final test accuracy of f: that is, the epoch 0 points in Figure 9 would agree with roughly the epoch 200 points in Figure 10. This comparison is plotted directly in Appendix C. Furthermore, the test accuracies of predictions of kernel regression using the pNTK are lower than those achieved by the NTKs of infinite-width counterparts for fully-connected and fully-convolution networks. This is consistent with results on eNTK by Arora et al. (2019a); Lee et al. (2020), although Arora et al. (2019a) studied only a "CIFAR-2" version.

It is worth noting from Figures 9 and 11 that, in contrast to the findings of Fort et al. (2020), we observe that the corresponding pNTK of the NN f continues to change even after epoch 50 of SGD training. Although for fully-connected networks and some versions of ResNet18 this change is not significant, in fully-convolutional networks and WideResNets the pNTK continues to exhibit changes until epoch 150, where the training error has vanished. We remark that Fort et al. (2020) analyzed eNTKs based on only 500 random samples from CIFAR-10, while the pNTK approximation has enabled us to run our analysis on the 100-times larger full dataset.

Lastly, to help the community better analyze the properties of NNs and their training dynamics and avoid wasting computation by redoing this work, we plan to share computed pNTKs for all the mentioned architectures and widths, as well as pNTKs of ResNets with 34, 50, 101 and 152 layers on CIFAR-10 and CIFAR-100 (Xiao et al., 2017) datasets, both at initialization and using pretrained ImageNet (Deng et al., 2009) weights. We hope that our contribution will enable further analyses and breakthroughs towards a stronger theoretical understanding of the training dynamics of deep neural networks.

6 Discussion

Our pseudo-NTK approach to approximating the empirical Neural Tangent Kernel has provable bounds, good empirical performance, and multiple orders of magnitude improvement in runtime speed and memory requirements over the direct empirical NTK. We evaluate our claims and the quality of the approximation under diverse settings, giving new insights into the behaviour of the empirical NTK with trained representations. We help justify the correctness of recent approximation schemes, and hope that our rigorous results and thorough experimentation will help researchers develop a deeper understanding of the training dynamics of finite networks, and develop new practical applications of the NTK theory.

One major remaining question is to theoretically analyze what happens to the pNTK or eNTK during SGD training of the network. In particular, the fast convergence of $\hat{f}^{lin}$ and f^{lin} when training the network, as seen in Figures 7 and 8, runs counter to our expectations based on the approximation worsening in Frobenius norm (Figure 7), maximum eigenvalue (Figure 4), and condition number (Figure 6). This seems likely to be very important to practical use of the pNTK.

Perhaps relatedly, it is also unclear why pNTK consistently results in higher prediction accuracies than when kernel regression is done using eNTK, given that our motivation for pNTK is entirely in terms of approximating the eNTK (Figure 8). Intuitively, this may be related to a regularization-type effect: the pNTK corresponds to a particularly limited choice of a "separable" operator-valued kernel (e.g. Álvarez et al., 2012). Separable kernels are a common choice in that literature for both computational and regularization reasons; by enforcing this particularly simple form, we remove many degrees of freedom relating to the interaction between "tasks" (different classes) that may be unnecessary or hard to estimate accurately with the eNTK. This might, in some sense, correspond to a one-vs-one rather than one-vs-rest framework in the intuitive sense discussed in Section 3. Understanding this question in more detail might require a more detailed understanding of the structure of the eNTK at finite width, and/or a much more detailed understanding of the interaction between classes in the dataset with learning in the NTK regime.

Finally, even the pNTK is still rather expensive compared to running SGD on neural networks. It might make for a better starting point than the full eNTK for other speedup methods, however, like kernel approximation schemes or advanced linear algebra solvers (e.g. Rudi et al., 2017).

REFERENCES

Ben Adlam, Jaehoon Lee, Lechao Xiao, Jeffrey Pennington, and Jasper Snoek. Exploring the uncertainty properties of neural networks' implicit priors in the infinite-width limit. 2020.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In *NeurIPS*, 2019a.

Sanjeev Arora, Simon S Du, Zhiyuan Li, Ruslan Salakhutdinov, Ruosong Wang, and Dingli Yu. Harnessing the power of infinitely wide deep nets on small-data tasks. 2019b.

Gregor Bachmann, Thomas Hofmann, and Aurélien Lucchi. Generalization through the lens of leave-one-out error. In *ICLR*, 2022.

Xiangning Chen, Cho-Jui Hsieh, and Boqing Gong. When vision transformers outperform resnets without pre-training or strong data augmentations. 2021.

Peter Craven and Grace Wahba. Smoothing noisy data with spline functions. *Numerische mathematik*, 31(4):377–403, 1978.

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

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

Jean-Yves Franceschi, Emmanuel de Bézenac, Ibrahim Ayed, Mickaël Chen, Sylvain Lamprier, and Patrick Gallinari. A neural tangent kernel perspective of gans. 2021.

Tamir Hazan and Tommi Jaakkola. Steps toward deep kernel methods from infinite neural networks. 2015.

Bobby He, Balaji Lakshminarayanan, and Yee Whye Teh. Bayesian deep ensembles via the neural tangent kernel. *Advances in Neural Information Processing Systems*, 33:1010–1022, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015. URL https://arxiv.org/abs/1502.01852.

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

David Holzmüller, Viktor Zaverkin, Johannes Kästner, and Ingo Steinwart. A framework and benchmark for deep batch active learning for regression. 2022.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *NeurIPS*, 2018.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes. 2017.

Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *NeurIPS*, 2019.

Jaehoon Lee, Samuel Schoenholz, Jeffrey Pennington, Ben Adlam, Lechao Xiao, Roman Novak, and Jascha Sohl-Dickstein. Finite versus infinite neural networks: an empirical study. *Advances in Neural Information Processing Systems*, 33:15156–15172, 2020.

Alexander G de G Matthews, Mark Rowland, Jiri Hron, Richard E Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. 2018.

Mohamad Amin Mohamadi, Wonho Bae, and Danica J. Sutherland. Making look-ahead active learning strategies feasible with neural tangent kernels, 2022.

Radford M Neal. Priors for infinite networks. In *Bayesian Learning for Neural Networks*, pp. 29–53. Springer, 1996.

Timothy Nguyen, Zhourong Chen, and Jaehoon Lee. Dataset meta-learning from kernel ridge-regression. 2020.

Timothy Nguyen, Roman Novak, Lechao Xiao, and Jaehoon Lee. Dataset distillation with infinitely wide convolutional networks. *Advances in Neural Information Processing Systems*, 34, 2021.

Roman Novak, Lechao Xiao, Jaehoon Lee, Yasaman Bahri, Greg Yang, Jiri Hron, Daniel A Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Bayesian deep convolutional networks with many channels are gaussian processes. 2018.

Roman Novak, Jascha Sohl-Dickstein, and Samuel Stern Schoenholz. Fast finite width neural tangent kernel. In *Fourth Symposium on Advances in Approximate Bayesian Inference*, 2021.

Daniel S Park, Jaehoon Lee, Daiyi Peng, Yuan Cao, and Jascha Sohl-Dickstein. Towards nngp-guided neural architecture search. 2020.

Alessandro Rudi, Luigi Carratino, and Lorenzo Rosasco. FALKON: An optimal large scale kernel method. In *NeurIPS*, 2017.

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547, 2020.

Martin J. Wainwright. *High-Dimensional Statistics: A Non-Asymptotic Viewpoint*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2019. doi: 10.1017/9781108627771.

Haonan Wang, Wei Huang, Andrew Margenot, Hanghang Tong, and Jingrui He. Deep active learning by leveraging training dynamics. 2021.

Alexander Wei, Wei Hu, and Jacob Steinhardt. More than a toy: Random matrix models predict how real-world neural representations generalize. 2022.

Christopher Williams. Computing with infinite networks. *Advances in neural information processing systems*, 9, 1996.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Lechao Xiao, Yasaman Bahri, Jascha Sohl-Dickstein, Samuel Schoenholz, and Jeffrey Pennington. Dynamical isometry and a mean field theory of cnns: How to train 10,000-layer vanilla convolutional neural networks. In *International Conference on Machine Learning*, pp. 5393–5402. PMLR, 2018.

Lechao Xiao, Jeffrey Pennington, and Samuel Schoenholz. Disentangling trainability and generalization in deep neural networks. In *International Conference on Machine Learning*, pp. 10462–10472. PMLR, 2020.

Greg Yang. Wide feedforward or recurrent neural networks of any architecture are gaussian processes. *Advances in Neural Information Processing Systems*, 32, 2019.

Greg Yang. Tensor programs ii: Neural tangent kernel for any architecture, 2020.

Greg Yang and Etai Littwin. Tensor programs iib: Architectural universality of neural tangent kernel training dynamics, 2021.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *BMVC*, 2016.

Yufan Zhou, Zhenyi Wang, Jiayi Xian, Changyou Chen, and Jinhui Xu. Meta-learning with neural tangent kernels. 2021.

Mauricio A. Álvarez, Lorenzo Rosasco, and Neil D. Lawrence. Kernels for vector-valued functions: A review. *Foundations and Trends® in Machine Learning*, 4(3):195–266, 2012. doi: 10.1561/2200000036.