Pacific Northwest
NATIONAL LABORATORY

PNNL- 35833

Report on Pure Mathematics of Machine Learning

November 2023

Tony Chiang
Andrew Engel
Saad Qadeer
Max Vargas
Sutanay Choudhury
Panos Stinis

U.S. DEPARTMENT OF ENERGY

Prepared for the U.S. Department of Energy
under Contract DE-AC05-76RL01830

DISCLAIMER

PACIFIC NORTHWEST NATIONAL LABORATORY
operated by
BATTELLE
for the
UNITED STATES DEPARTMENT OF ENERGY
under Contract DE-AC05-76RL01830

Printed in the United States of America

Available to DOE and DOE contractors from
the Office of Scientific and Technical Information,
P.O. Box 62, Oak Ridge, TN 37831-0062
www.osti.gov
ph: (865) 576-8401
fox: (865) 576-5728
email: reports@osti.gov

Available to the public from the National Technical Information Service
5301 Shawnee Rd., Alexandria, VA 22312
ph: (800) 553-NTIS (6847)
or (703) 605-6000
email: info@ntis.gov
Online ordering: http://www.ntis.gov

PNNL- 35833

Report on Pure Mathematics of Machine Learning

November 2023

Tony Chiang
Andrew Engel
Saad Qadeer
Max Vargas
Sutanay Choudhury
Panos Stinis

Prepared for
the U.S. Department of Energy
under Contract DE-AC05-76RL01830

Pacific Northwest National Laboratory
Richland, Washington 99354

Abstract

We report on the findings conducted by the Pure Mathematics in Machine Learning (PMML) Project under the umbrella of the Mathematics in Artificial Reasoning in Sciences (MARS) Laboratory Initiative at the Pacific Northwest National Laboratory. The findings contained in this report spans three years of research from the project team. The objectives of the PMML project are two-fold: 1. To gain understanding of modern machine learners through empirical research via structured and sound experimental design, and 2. To prove mathematical theories based upon these empirical findings. We present both empirical results along with mathematical theories in this document.

Summary

This work establishes a fundamental relationship between Deep Neural Networks and their underlying kernel machines. Because every neural network may be viewed as a feature map, there is an associated kernel with this new feature space. Our work primarily focuses on understanding the relationship between finite width neural networks and their underlying empirical neural tangent kernels (eNTK). We show the following:

1. The block diagonal approximation of the eNTK, or the trace-NTK (tr-NTK) is a suitable surrogate model for the underlying network.

2. Building upon the work on Park et al., we apply the Johnson-Lindenstrauss theorem and find a projected trNTK whose residual decays exponentially.

3. Applying a kernel generalized linear model (kGML) to the trNTK, we look at the data attribution framework without imposing a sparsity assumption and we find that this assumption might be unwarranted for neural network analysis.

4. We prove that the empirical conjugate kernel (CK) is close in norm to the eNTK. The CK is a much more reasonable kernel to study as it relies on the final embedded feature representation upon which the network makes its decision.

5. We analyzed the CK on a variety of different state-of-the-art models: vision and language. We are able to explain outcomes via the CK machines.

We present these findings in this report.

Acknowledgments

This research was supported by the **Mathematics for Artificial Reasoning in Science** under the Laboratory Directed Research and Development (LDRD) Program at Pacific Northwest National Laboratory (PNNL). PNNL is a multi-program national laboratory operated for the U.S. Department of Energy (DOE) by Battelle Memorial Institute under Contract No. DE-AC05-76RL01830.

Acronyms and Abbreviations

NTK	Neural Tangent Kernel
NN	Neural Network
AI	Artificial Intelligence
GPU	Graphics Processing Unit
CK	Conjugate Kernel

Contents

Section 1 ..... **2**
Section 2 ..... **11**
Section 3 ..... **32**
Section 4 ..... **38**
Section 5 ..... **43**
References **54**

Figures

Figure 1 Linear Realization of Bert-base Model. Each panel shows a linearization of a Bert-base transfer model, initialized from a different seed. An invertible mapping is fit between the kGLM and NN to transform the kGLM's final activations to the NN's, described in Appendix I. Both τK and the Coefficient of Determination (R2) are shown for each model.

Figure 2 Overview of Using Kernel Functions for Data Attribution A) An image from the test dataset of CIFAR10 is chosen. B) We propagate the test image through the NN and plot the mean attribution of the training points from each class for each output neuron. C) Zooming into the neuron representing class "dog", we view the distribution of attributions as a modified boxplot with central lines the mean and outliers shown as flier points. The mean lines are always observed to be within the inner quartile, suggesting that no sparse number of datapoints dominate the central value, and therefore, do not dominate the data attribution.

Figure 3 The results of function regression for 100 different NNs, trained for 2400 epochs, and the corresponding approximations using the NTK, CK, and CKJ extracted from the NN at the end of the training.

Figure 4 The test errors for NNs over the course of being trained to approximate the given functions, and for the corresponding NTK, CK, and CKJ approximations. The errors are averaged over ten iterations to reduce the effects of random initialization.

Figure 5 Function regression results for 100 NNs with the widths of the hidden layers set to 256, and the corresponding NTK, CK, and CKJ approximations.

Figure 6 The test errors for function regression for 100 trained NNs using the ReLU activation function, and the corresponding NTK and CK approximations.

Figure 7 Averaged test errors for function regression over the course of training ten NNs using ReLU activations, and the corresponding NTK and CK approximations.

Figure 8 The test cross-entropies and accuracies for logistic regression performed with 100 NNs, and the corresponding qNTK and CK results.

Figure 9 The cross-entropies and accuracies for the test dataset for logistic regression performed with 100 NNs with hidden layer widths set to 64, and the corresponding NTK and CK results.

Figure 10 Test results for 100 NNs using the ReLU activation function for logistic regression, and the corresponding NTK and CK metrics.

Figure 11 Evolution of the test metrics averaged over ten iterations of NNs used to solve the logistic regression problem, and the corresponding NTK and CK results.

Figure 12 Kernel density estimate of the largest principal component of GPT-2's pre-trained final embedding representation of Sentiment140 P (the processed training set) and Sentiment140 M (the manually curated test set).

Figure 13 (a) Kernel Density Estimate of the two largest principal components of the pre-trained final embedding representation of P and M. (b) Normalized Histograms of generalization for different types of training and models, with state-of-the-art fine-tuned model performance shown as dashed lines for comparison.

Figure 14 KDE of 20k points of P_{YN} and 29k points of P used to generate P_{YN}

Figure 15 Kernel Density Estimates (KDEs) plotting the values of PCs 1 and 2 of the pre-trained embedded kernels for four different architectures for data corresponding to P and M .

Figure 16 Diagram showing ResNet-18 architecture with residual blocks labeled.

Figure 17 Performance of 10-NN (a), NCC (b), and SVM (c) after projecting activations from each residual block onto first d principal components.

Figure 18 For each model: number of PCs (a) and the percentage of variance explained by those PCs (b) needed to attain 90% of maximum classification accuracy at each residual block.

Figure 19 Different spectral behaviors in Table 1: (a) The initial and trained spectra of W in Case 1. The spectrum is invariant based on the Q-Q subplot. (b) The initial and trained spectra of KCK in Case 3. There is an outlier (orange arrow) in the spectrum after training. (c) The initial and trained spectra of KCK in Case 4. We refer to Appendix B.2 for other spectra of weight, CK, and NTK matrices in Case 1-4, where analogous phenomena hold for other matrices.

Figure 20 a) Alignment between teacher feature β and first PC of the trained/initial weights in Case 3 of Table 1. (b)-(d) Transitions of λmax(W tW ⊤ t /d), λmax(KCK t) and alignments (|β⊤u1|/ ∥β∥ and |y⊤v1|/ ∥y∥ where u1 and v1 are the first singular vectors of W t and either KCK t or KNTK t , respectively) when increasing the learning rate η while training the NN with SGD. In the green region, the largest eigenvalues are attached to the bulk (black horizontal lines) and the alignments are weak; in the orange one, outliers become apparent, and the alignments become stronger. For different η, we train the same NN with the same dataset until the training loss is less than 10−5. Here, "lr" in the x-axis represents varying learning rates.

Figure 21 (a) Evolution of KTA of CK defined by (15) with respect to training labels for Cases 1, 3&4 in Table 1. We normalize the epoch scales (x-axis) for better observations. Heavy-tailed phenomena: (b) Evolutions of PC angles θi between feature subspace U of (8) and top 100 eigenspace of W ⊤ t W t during training with Adam (solid line), SGD (dashed line) and GD (dash-dot). For the first PC θ1, see Figure 17 in Appendix B.4. (c) The CK spectra at two initializations for W : standard Gaussian and Cauchy distributions. (d) Weight spectra at initial and after SGD training. After training the weight reveals a heavy tail, but generalizes not as well as former examples (test loss 1.47504; R2 score −0.48).

Figure 22 Different NTK spectra for a small-CNN model on CIFAR-2. The subplots are Q-Q plots for the comparison between initial and trained spectra. Test accuracies: (a) 79%, (b) 84%, (c) 86.4%.

Figure 23 We use SGD for fine-tuning the BERT model. The training accuracy is 95.90% and the test accuracy is 84%. (a) The evolution of first and second eigenvalues of empirical CK during fine-tuning. (b) The alignments of training labels with first and second eigenvectors of CK during fine-tuning. See Figure 7 for the spectra of CK at different epochs.

Tables

Table1: Choice of $_K$ = trNTK faithfully forms a surrogate model of underlying NN. We perform each experiment with '# Models' independent seeds. For each model and dataset we train and extract the trNTK, train a kGLM, then calculate and report the τ_K correlation between the kGLM softmax probability and NN softmax probability for the correct class. The NN test accuracy column shows that training terminates with a highly performant model, and the test accuracy differential (TAD) columns reports the difference between the kGLM test accuracy and the NN test accuracy. We report the leading digit of error (standard error of the mean) as a parenthetical, when available.

Table 2: Comparison across surrogate feature spaces. For ResNet18 and Bert-base experiments we report the faithfulness as τ_K and test-accuracy-differential (TAD) for each kernel function: the trace-NTK (trNTK), unnormalized trace-NTK (trNTK$_0$, the projection trace NTK (proj-trNTK), the projection psuedo NTK (proj-pNTK), the embedding kernel (Em) and the conjugate kernel (CK). If available, we report leading digit of error (standard error of the mean) as a parenthetical.

Table 3: Poisoned data attribution forensics. We compute each kernel function between all poisoned training data and the clean test dataset. We report τ_K and TAD between the kGLM and NN for both the poisoned (poi.) and clean set of unseen test images. Finally, we evaluate each kernel as a filter for identifying unseen poisoned data through high similarity to poisoned training data and report the performance as Precision and Recall.

Table 4: Computational Complexity of Large Model Experiments. We report time to compute each of the trNTK, proj-trNTK, and proj-pNTK for the large model large dataset experiments are shown.

Table 5: Accuracy results for training and testing the LP and FT approaches on a human-curated subset of the Sentiment140 dataset. Mean values ± the standard error of the mean are shown.

Table 6: Those equipped with the F T method are fine-tuned using 20k datapoints from P and tested on all 359 points of M. Those with the L P method have a linear probe trained using 300 points of M and tested on the remaining 59 points of M. Values are shown ± the standard error of the mean.

Table 7: We present selected examples of potentially mislabeled in dataset P . The curators of Sentiment140 scrubbed the tweets in P of emoticons. A ':(' corresponds to a 0 label, ':)' corresponds to 1 label.

Table 8: Four models with the same architecture (n = 2000, h = 1500, d = 1000, and σ is normalized tanh), but different choices of initial learning rates and optimizers listed in Table 1. The training label noise $\sigma_\varepsilon = 0.3$ and the teacher model is defined by (9) with σ_* a normalized softplus and $\tau = 0.2$. We observe that simply choosing an optimizer and learning rate can affect the shapes of the final spectra and the performance of the NN, as measured by R_2 scores and test errors.

DOIs

Section 1: Faithful and Efficient Explanations for Neural Networks via Neural Tangent Kernel Surrogate Models
https://doi.org/10.48550/arXiv.2305.14585

Section 2: Efficient kernel surrogates for neural network-based regression
https://doi.org/10.48550/arXiv.2310.18612

Section 3: Foundation Model's Embedded Representations May Detect Distribution Shift
https://doi.org/10.48550/arXiv.2310.13836

Section 4 Exploring Learned Representations of Neural Networks with Principal Component Analysis
https://doi.org/10.48550/arXiv.2309.15328

Section 5: Spectral Evolution and Invariance in Linear-width Neural Networks
https://doi.org/10.48550/arXiv.2211.06506

This page is intentionally left blank.

1.0 DOI 10.48550/arXiv.2305.14585

FAITHFUL AND EFFICIENT EXPLANATIONS FOR NEURAL NETWORKS VIA NEURAL TANGENT KERNEL SURROGATE MODELS

Andrew Engel[1*] **Zhichao Wang**[2] **Natalie S. Frank**[3] **Ioana Dumitriu**[2]
Sutanay Choudhury[1] **Anand Sarwate**[4] **Tony Chiang**[1,5*]

[1]Pacific Northwest National Laboratory [2]University of California, San Diego
[3]Courant Institute, NYU [4]Rutgers University [5]University of Washington
{andrew.engel,sutanay.choudhury,tony.chiang}@pnnl.gov
{zhw036,idumitriu}@ucsd.edu
nf1066@nyu.edu
ads221@soe.rutgers.edu

ABSTRACT

A recent trend in explainable AI research has focused on surrogate modeling, where neural networks are approximated as simpler ML algorithms such as kernel machines. A second trend has been to utilize kernel functions in various explain-by-example or data attribution tasks to investigate a diverse set of neural network behavior. In this work, we combine these two trends to analyze approximate empirical neural tangent kernels (eNTK) for data attribution. Approximation is critical for eNTK analysis due to the high computational cost to compute the eNTK. We define new approximate eNTK and perform novel analysis on how well the resulting kernel machine surrogate models correlate with the underlying neural network. We introduce two new random projection variants of approximate eNTK which allow users to tune the time and memory complexity of their calculation. We conclude that kernel machines using approximate neural tangent kernel as the kernel function are effective surrogate models, with the introduced trace NTK the most consistent performer. Open source software allowing users to efficiently calculate kernel functions in the PyTorch framework is available here*.

1 INTRODUCTION

Explainability remains a critical open problem for applications of deep neural networks (NNs) (Leavitt & Morcos, 2020). Explain-by-example techniques (Lai et al., 2021; Yang et al., 2020) have emerged as a major category of algorithms for explainability, including prototype examples (Chen et al., 2018), Deep K-Nearest Neighbors (Papernot & McDaniel, 2018; Wang et al., 2021; Dziedzic et al., 2022), and Representer Points (Yeh et al., 2018). These techniques explain models by providing example(s) that capture model behavior on new data. Kernel functions (Alvarez et al., 2011) are a natural choice for building explain-by-example algorithms (Yeh et al., 2018); a kernel measures the similarity between individual data points via an inner product in a reproducing kernel Hilbert space (RKHS) (Hilbert, 1904; Ghojogh et al., 2021). A critical open issue has been to find a RKHS to represent a linearized feature space of a NN that faithfully explains its decisions (Hanawa et al., 2021).

In this work, we investigate computationally efficient approximations to the empirical neural tangent kernel (eNTK), which is a kernel function motivated by advances in the theory of deep learning (Jacot et al., 2018). Its well established that NNs trained using gradient descent are equivalent to kernel machines (Scholkopf & Smola, 2001) with a kernel constructed from a sum over eNTK (Lee et al., 2020) computed at each gradient step (Domingos, 2020; Bell et al., 2023). Given this equivalence, we would like to evaluate the eNTK as the kernel function for an explain-by-example algorithm;

*https://github.com/pnnl/projection_ntk

however, computing eNTK is computationally expensive (Novak et al., 2022; Chen et al., 2021), so low computational cost approximations have been developed instead (Mohamadi & Sutherland, 2022). We are the first to define and evaluate one such approximate kernel, the trace neural tangent kernel (trNTK). Additionally, we build from the work of Park et al. (2023) to provide software to compute random-projection variants that can be computed and stored with lower time and memory cost over traditional eNTK. Using these approximations, we build low-cost and faithful surrogate models for neural network classifiers.

Our methodology improves over the past evaluation of kernel surrogate models. We measure the faithfulness of a kernel function by assessing how well a kernel generalized linear model (kGLM) (Hofmann et al., 2007) correlates with the softmax probabilities of the original NN using a rank correlation. Previous evaluations relied on test accuracy (Mohamadi & Sutherland, 2022; Long, 2021), or having high similarity to the correct class (Hanawa et al., 2021), which are both flawed. Our approach and accompanying code-repository will allow users to evaluate how close their own NNs are to kernel machines in the PyTorch framework with limited overhead (Paszke et al., 2019).

Contributions

We make three major contributions in this work:

1. We define new kernel functions for faithful approximation of an underlying neural network; we are the first to analyze random projection variants that permit tuning the computational and memory expense of eNTK methods.
2. We are the first to show that approximate eNTK kernel surrogate models are consistently correlated to the underlying neural network across experiments including ResNet18 on CIFAR10 and Bert-base on COLA.
3. We compare explanations of NN decisions generated from each kernel function through a data attribution strategy and through an explain-by-example strategy; this is the first such qualitative evaluation between approximate eNTK.

Related Work

Surrogate Models for Explaining Neural Network Behavior. Recent work in explainable AI has focused on determining when NNs are exactly equivalent to other common ML algorithms (Lee et al., 2018; Balestriero & Baraniuk, 2018; Schmitz et al., 1999), including kernel machines. It has been shown that infinitely wide NNs are equivalent to a kernel machine with kernel function chosen as the neural tangent kernel (Jacot et al., 2018). These infinitely wide models, however, do not replicate the feature learning behavior seen in finite-width networks (Chizat et al., 2018; Yang & Hu, 2021; Wang et al., 2022). Subsequently, researchers turned to investigating properties of finite-width models with NTK computed at various checkpoints (Domingos, 2020; Bell et al., 2023) and/or after training (Long, 2021). This framework was used to explore inductive biases (Ortiz-Jiménez et al., 2021), feature learning (Radhakrishnan et al., 2022), learning dynamics (Fort et al., 2020; Atanasov et al., 2022), and adversarial faithfulness (Tsilivis & Kempe, 2023; Loo et al., 2022). Support vector machines (Vapnik, 2000) using eNTK or approximate eNTK kernels computed after training were shown to achieve the same test accuracy as the underlying NN (Atanasov et al., 2022; Long, 2021; Vyas et al., 2022; Mohamadi & Sutherland, 2022). Our work extends the evaluation of kernel surrogate models by evaluating whether general kernel machines approximate the underlying neural network function itself, rather than simply reproduce the same test accuracy.

Kernels for Explainability. Kernel functions defined from various RKHS have been proposed to explain the behavior of NN in different contexts. Prior works have studied the data counterfactual attribution task (Park et al., 2023), identifying points of high influence (Koh & Liang, 2017), or points that greatly change the value of loss when removed (Pruthi et al., 2020), and tracing knowledge to training data (Akyürek et al., 2023). Each of the aforementioned works have studied loss-based kernels and rely upon the availability of test labels. Our goals are to model the classification behavior on any new data, including unlabeled inputs, so we do not compare to these loss-based kernels. Most relevant to our work, Yeh et al. (2018) (hereafter Representer Points) used a kernel formed from the NN final embedding in what we call the data attribution task. We build from Representer Points by evaluating their assumptions under new approximate eNTK kernels.

DOI 10.48550/arXiv.2305.14585

Computationally Feasible Approximations of the eNTK The computational cost of direct eNTK is prohibitively high for large models and datasets. Advances on this issue have been two pronged: Some groups focus on algorithmic improvements to calculate the eNTK directly (Novak et al., 2022). An alternative strategy has been to avoid eNTK calculation and instead compute kernel functions that share similar structure to the eNTK (Mohamadi & Sutherland, 2022). One such approximate kernel was introduced quietly in Chen et al. (2021) which we refer to as the trace-NTK (trNTK). We are the first to explicitly investigate the trNTK's properties. Finally, Park et al. (2023), hereafter TRAK, utilized random projection matrices to scale the computation of a loss-based kernel function. We modify TRAK to compute projected variants of approximate eNTK.

Evaluating Kernel Attribution. In this paper, we use three evaluation strategies. The first focuses on evaluating the faithfulness of the surrogate model through rank correlation. The second evaluates surrogate model performance on a data-attribution task. We follow the methodology in Shan et al. (2022) to evaluate the model via precision and recall in tracing decisions on poisoned test data back to poisoned training data. Finally, we compare kernels qualitatively via explain-by-example. Previous work on kernel evaluation did not use a surrogate model-based approach; instead, they relied upon whether the attributions trace to training data of the correct class (Hanawa et al., 2021) or whether surrogate models replicate NN test accuracy (Mohamadi & Sutherland, 2022; Long, 2021). We will explain why our methodology is an improvement over these past evaluations strategies.

2 PRELIMINARIES

Neural Networks for Classification. We consider the supervised classification problem with C classes. Consider a data input $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^n$ with n the dimensionality of inputs, and a one-hot encoded data label vector $\boldsymbol{z} \in \mathcal{Z} \subseteq \mathbb{R}^C$. We define a neural network $F(\boldsymbol{x}\,;\boldsymbol{\theta}) : \mathcal{X} \to Y$ where the output space $\mathcal{Y} \subseteq \mathbb{R}^C$ is an intermediary step in our classification called a "logit." The NN $F(\boldsymbol{x}\,;\boldsymbol{\theta})$ is parameterized by the vector $\boldsymbol{\theta}$ and was learned via back-propagation to minimize the cross entropy loss between the target label vector $\boldsymbol{z}$ and softmax probability vector $\sigma(F(\boldsymbol{x}\,;\boldsymbol{\theta}))$, with $\sigma : \mathcal{Y} \to \mathcal{Z}$ the softmax function. We denote the c-th scalar output of the network as F^c. We interpret the predicted confidence for the c-th class for input $\boldsymbol{x}$ as $\sigma(F(\boldsymbol{x}\,;\boldsymbol{\theta}))^c$.

Kernel Functions. Kernel functions implicitly map the data vector $\boldsymbol{x}$ to a feature vector $\rho(\boldsymbol{x})$ in a higher dimensional RKHS $\mathcal{V}$ for which the kernel function $\boldsymbol{\kappa}(\cdot,\cdot)$ evaluates the inner product of two feature vectors in $\mathcal{V}$. We will notate the data matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N] \in \mathbb{R}^{N\times n}$ with N the number of training samples. With some abuse of notation, we will write $\boldsymbol{\kappa}(\boldsymbol{x}, \boldsymbol{X}) \in \mathbb{R}^N$ for the vector whose j-th component is $\boldsymbol{\kappa}(\boldsymbol{x}, \boldsymbol{x}_j)$ and $\boldsymbol{\kappa}(\boldsymbol{X}, \boldsymbol{X}) \in \mathbb{R}^{N\times N}$ for the matrix whose (i, j)-th entry is $\boldsymbol{\kappa}(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

Kernel General Linear Models as Surrogate Models We limit our investigation of surrogate models to kernel general linear models. We define a general kernel linear model kGLM : $\mathcal{X} \to \mathcal{Y}$ as:

$$\text{kGLM}(\boldsymbol{x}) := \boldsymbol{W}\boldsymbol{\kappa}(\boldsymbol{x}, \boldsymbol{X}) + \boldsymbol{b}, \tag{1}$$

where $\boldsymbol{W} \in \mathbb{R}^{C\times N}$ is a learnable weight matrix, $\boldsymbol{\kappa}$ is the kernel function, and $\boldsymbol{b} \in \mathbb{R}^C$ is a learnable bias vector. We compute classifications from kGLM by mapping the final activations to softmax confidences. The parameters $\boldsymbol{W}$ and $\boldsymbol{b}$ are learned using an optimizer to minimize the cross entropy loss on the same dataset upon which the NN is trained. Given an input $\boldsymbol{x}$, the softmax activation σ, and a NN $F(\boldsymbol{x}\,;\boldsymbol{\theta})$, the ideal surrogate modeling goal is to find a kGLM that satisfies:

$$\sigma(\text{kGLM}(\boldsymbol{x})) = \sigma(F(\boldsymbol{x}, \boldsymbol{\theta}))), \tag{2}$$

for all $\boldsymbol{x}$. Keeping this ideal in mind is useful for building intuition, but in practice, we will relax from this ideal goal for reasons described below.

Data Attribution with Kernels. Our main motivation is to explain neural networks through data attribution. Given the choice of kernel function $\boldsymbol{\kappa}$, the scalar valued data attribution for the c-th class for a test input $\boldsymbol{x}$ and a training datapoint $\boldsymbol{x}_i$ is given by:

$$A(\boldsymbol{x}, \boldsymbol{x}_i)^c := \boldsymbol{W}_{c,i}\,\boldsymbol{\kappa}(\boldsymbol{x}, \boldsymbol{x}_i) + \frac{\boldsymbol{b}_c}{N}. \tag{3}$$

Where the $\frac{\boldsymbol{b}_c}{N}$ term is necessary to ensure that the sum over the attributions for the entire training dataset is equal to the kGLM's logit for class c, $\sum_{i=1}^{N} A(\boldsymbol{x}, \boldsymbol{x}_i)^c = \text{kGLM}(\boldsymbol{x})^c$. If the kGLM is an

DOI 10.48550/arXiv.2305.14585

ideal surrogate model Eq. (2), then the softmax function applied to the vector created from each class attribution will equal the NN confidence in each class. Consequently, we will have decomposed the reasoning for the NN's specific confidence in each class to a linear combination of similarities between $\boldsymbol{x}$ and each training datapoint $\boldsymbol{x}_i$. This definition of data attribution satisfies the conceptual definition given in TRAK, and was originally introduced in Representer Points.

3 METHODS

We now turn towards the novel work of this research. In the following sections we describe our measure of faithfulness then introduce the kernel functions.

Evaluating the Faithfulness of Surrogate Models. Given many choices of kernel functions we require a measure to determine which surrogate models have higher approximation quality (i.e., faithfulness) to the NN. We relax from the ideal surrogate model goal Eq. (2) and instead evaluate kernel functions by how well they are correlated with the neural network using the Kendall-τ rank correlation. We will describe how this choice reflects our intuition of an ideal surrogate model, but first we will justify why we should move beyond the evaluation strategies of prior works.

Previous measures of faithfulness have been flawed. Prior work used test accuracy to evaluate whether an approximate eNTK formed a good kernel machine surrogate model to a NN (Mohamadi & Sutherland, 2022). Test accuracy is invariant over which specific datapoints either model is correct or incorrect upon; therefore, two models could have wildly different behavior on any $\boldsymbol{x}$ but still equivalent test accuracy when evaluated across $\boldsymbol{X}$. Other works have suggested using high similarity to the correct class to evaluate kernel functions (Hanawa et al., 2021); however, our goal is that kernel functions reflect the neural network behavior rather than necessarily be highly performant. Finally, prior work has implicitly used Pearson correlation as a faithfulness measure (Yeh et al., 2018). Pearson correlation is problematic because we observe two point clouds at mutual low confidence and mutual high confidence between the kGLM and NN models across each experiment. These point clouds serve as anchors that force the covariance, and therefore Pearson correlation, to be large. We require a measure that does not conflate the covariance with faithfulness.

Instead, we use the Kendall-τ rank correlation (Kendall, 1938). For a sequence $S_\tau = \{(a_1, b_1), \ldots, (a_N, b_N)\}$ we say that a pair (a_i, b_i) and (a_j, b_j) with $i \neq j$ are concordant if either both $a_i > a_j$ and $b_i > b_j$ or $a_i < a_j$ and $b_i < b_j$. Otherwise, the pair is discordant. We count the total number of concordant and discordant pairs, respectively denoted as NC and ND. Then, Kendall-τ is defined: $\tau_K(S_\tau) := \frac{(\mathrm{NC}-\mathrm{ND})}{\mathrm{NC}+\mathrm{ND}}$.

To assess the faithfulness of a surrogate model, we compute τ_K between the softmax probability of the neuron representing the correct class, $\sigma(F(\boldsymbol{x}\,;\boldsymbol{\theta}))^c$, and the kGLM softmax probability for the output representing the correct class, $\sigma(\mathrm{kGLM}(\boldsymbol{x}))^c$. τ_K was chosen for two reasons. First, τ_K has a range $[-1, 1]$ with $\pm$ 1 representing a monotonic relationship and a value of 0 representing no correlation. Second, if the relationship between the kGLM and NN is strictly monotonic, then an invertible mapping function exists between the kGLM softmax probabilities and the NN's (Bartle & Sherbert, 2011). Therefore, for a $\tau_K = 1$ we would recover the one-to-one ideal surrogate model relationship Eq. (2). In appendix I, we demonstrate how to find these mapping functions with iterative optimizers (Virtanen et al., 2020).

While we have argued that the test accuracy is flawed to measure faithfulness, we will report the test accuracy differential to be complete with prior works. We define test accuracy differential (TAD) as:

$$\mathrm{TAD} := \mathrm{TestAcc}_{\mathrm{kGLM}} - \mathrm{TestAcc}_{\mathrm{NN}}.$$

We now turn to defining the specific kernel functions we evaluate.

Trace Neural Tangent Kernel. For any two data inputs $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, we define the Jacobian of the NN's c-th output neuron with respect to $\boldsymbol{\theta}$ at datapoint $\boldsymbol{x}_i$ as $\boldsymbol{g}^c(\boldsymbol{x}_i;\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} F^c(\boldsymbol{x}_i;\boldsymbol{\theta})$, then the trNTK is a kernel function defined as:

$$\mathrm{trNTK}(\boldsymbol{x_i}, \boldsymbol{x}_j) := \frac{\sum_{c=1}^{C} \langle \boldsymbol{g}^c(\boldsymbol{x}_i;\boldsymbol{\theta}), \boldsymbol{g}^c(\boldsymbol{x}_j;\boldsymbol{\theta})\rangle}{\sum_{c=1}^{C} \langle \boldsymbol{g}^c(\boldsymbol{x}_i;\boldsymbol{\theta}), \boldsymbol{g}^c(\boldsymbol{x}_i;\boldsymbol{\theta})\rangle^{\frac{1}{2}} \sum_{c=1}^{C} \langle \boldsymbol{g}^c(\boldsymbol{x}_j;\boldsymbol{\theta}), \boldsymbol{g}^c(\boldsymbol{x}_j;\boldsymbol{\theta})\rangle^{\frac{1}{2}}}. \tag{4}$$

DOI 10.48550/arXiv.2305.14585

Where the inner product is the standard dot product. The normalization in the denominator of Eq. (4) makes the trNTK a kernel of cosine-similarity values. It has been suggested that this normalization helps smooth out kernel mass over the entire training dataset (Akyürek et al., 2022). The normalization ensures that two identical inputs always have maximum similarity value 1. Additional intuition about how this kernel relates to the geometry of the neural network function surface is available in appendix B. We provide additional details about these definitions in appendix C. In the following section, we relate this kernel to another approximate eNTK kernel, the psuedo neural tangent kernel.

Relationship to the Pseudo Neural Tangent Kernel. We can understand the motivation for the trNTK in the context of another approximate eNTK, called the psuedo neural tangent kernel (pNTK). The pNTK computed between inputs $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is a kernel function defined as:

$$\text{pNTK}(\boldsymbol{x}_i, \boldsymbol{x}_j) := \frac{1}{C}(\nabla_{\boldsymbol{\theta}} \sum_{c=1}^{C} F(\boldsymbol{x}_i\,; \boldsymbol{\theta})^c)^\top (\nabla_{\boldsymbol{\theta}} \sum_{c=1}^{C} F(\boldsymbol{x}_j\,; \boldsymbol{\theta})^c).$$

Mohamadi & Sutherland (2022) showed that the product of the pNTK$(\boldsymbol{x}_i, \boldsymbol{x}_j)$ with the $C \times C$ identity matrix is bounded in Frobenius norm to the eNTK by $\mathcal{O}(\frac{1}{\sqrt{n}})$, with n the width parameter of a feed forward fully connected NN with ReLU activation (Nair & Hinton, 2010; Glorot et al., 2011) and He-normal (He et al., 2015a) initialization, with high probability over random initialization.

We can frame the critical differences between the pNTK and trNTK by how each approximate the eNTK. The pNTK approximates the eNTK as a constant diagonal matrix with constant equal to the scalar kernel function given in (3). In contrast, the trNTK allows the diagonal elements of the eNTK approximation to vary, and in fact, calculates these values directly. Both the pNTK and trNTK perform a simplifying sum over the diagonal elements, which reduces the memory footprint of the approximations by a factor C^2 compared to the eNTK. We choose not to compare directly with the pNTK because the trNTK is a higher cost, but more precise, approximation of the eNTK. Instead, we focus our comparisons to much lower cost alternatives, including a projection variant of the pNTK.

Projection trNTK and Projection pNTK. For large number of parameters P and large datasets N, computing approximate eNTK remain expensive, therefore, we explore a random projection variant that allows us to effectively choose P regardless of architecture studied. Let $\boldsymbol{P}$ be a random projection matrix $\boldsymbol{P} \in \mathbb{R}^{K \times P}$, $K << P$, with all entries drawn from either the Gaussian $\mathcal{N}(0, 1)$ or Rademacher (with p=0.5 for all entries) distribution. K is a hyperparameter setting the projection matrix dimension. We set $K = 10240$ for all experiments. We use $\boldsymbol{P}$ to project the Jacobian matrices to a lower dimension, which reduces the memory needed to store the Jacobians can reduce the time complexity scaling. The Johnson-Lindenstrauss lemma ensures that most of the information in the original Jacobians is preserved when embedded into the lower dimensional space (Johnson & Lindenstrauss, 1984). We define the proj-trNTK and proj-pNTK as random projection variants of the trNTK and pNTK:

$$\text{proj-pNTK}(\boldsymbol{x}_i, \boldsymbol{x}_j) := \frac{\langle \boldsymbol{P} \sum_{c=1}^{C} \boldsymbol{g}^c(\boldsymbol{x}_i,\ \boldsymbol{\theta}), \boldsymbol{P} \sum_{c=1}^{C} \boldsymbol{g}^c(\boldsymbol{x}_j,\ \boldsymbol{\theta}) \rangle}{||\boldsymbol{P} \sum_{c=1}^{C} \boldsymbol{g}^c(\boldsymbol{x}_i,\ \boldsymbol{\theta})|| \cdot ||\boldsymbol{P} \sum_{c=1}^{C} \boldsymbol{g}^c(\boldsymbol{x}_j,\ \boldsymbol{\theta})||} \tag{5}$$

$$\text{proj-trNTK}(\boldsymbol{x}_i, \boldsymbol{x}_j) := \frac{\sum_{c=1}^{C} \langle \boldsymbol{P}\boldsymbol{g}^c(\boldsymbol{x}_i; \boldsymbol{\theta}), \boldsymbol{P}\boldsymbol{g}^c(\boldsymbol{x}_j; \boldsymbol{\theta}) \rangle}{\sum_{c=1}^{C} \langle \boldsymbol{P}\boldsymbol{g}^c(\boldsymbol{x}_i; \boldsymbol{\theta}), \boldsymbol{P}\boldsymbol{g}^c(\boldsymbol{x}_i; \boldsymbol{\theta}) \rangle^{\frac{1}{2}} \sum_{c=1}^{C} \langle \boldsymbol{P}\boldsymbol{g}^c(\boldsymbol{x}_j; \boldsymbol{\theta}), \boldsymbol{P}\boldsymbol{g}^c(\boldsymbol{x}_j; \boldsymbol{\theta}) \rangle^{\frac{1}{2}}}, \tag{6}$$

where both definitions include the cosine-normalization.

Random projection variants can improve the time complexity scaling for computing approximate eNTK under large dataset size and large number of parameters. Assuming computation via Jacobian contraction and time $[FP]$ for a forward pass and recalling that C is the number of output neurons, the eNTK time complexity is: $NC[FP] + N^2C^2P$ (Novak et al., 2022). The pNTK computation reduces this to $N[FP] + N^2P$; while the trNTK computation only reduces to $NC[FP] + N^2CP$. In contrast, the proj-pNTK costs $N[FP] + N^2K + NKP$, and the proj-trNTK costs $NC[FP] + CN^2K +$

DOI 10.48550/arXiv.2305.14585

$CNKP$. The final term in the projection variants is the cost of the extra matrix multiplication with the random projection matrix $\boldsymbol{P}$ and the Jacobian matrix. For $K << P$ and N large, projection variants reduce the time complexity.

Additional Kernel Functions. We also evaluate the conjugate kernel (CK) formed from the Gram matrix of the final embedding vector (Fan & Wang, 2020; Yeh et al., 2018), the un-normalized trNTK (trNTK^0) which is equal to the numerator of (4), and the embedding kernel (Akyürek et al., 2023), formed from a sum over the Gram matrices of embedding vectors from various layers in the network architecture. See appendix A for formal definition of these kernels.

4 Results

Experiments. Classification NNs with architectures and datasets (MNIST (Lecun et al., 1998), FMNIST (Xiao et al., 2017), CIFAR10 (Krizhevsky & Hinton, 2009), and COLA (Warstadt et al., 2018)) shown in Table 1 are trained using standard techniques. Additional details regarding datasets are provided in appendix H.1. Models that have a value of more than 1 in the column '# Models' in Table 1 are trained multiple times with different seeds to generate uncertainty estimates. The ResNet18 (He et al., 2015b), ResNet34, and MobileNetV2 (Sandler et al., 2018) models were trained by an independent research group with weights downloaded from an online repository (Phan, 2021). Bert-base (Devlin et al., 2019) weights were downloaded from the HuggingFace (Wolf et al., 2019) repository then transferred onto the COLA dataset, as is common practice for foundation models (Bommasani et al., 2021). After training, we calculate the trNTK and alternative kernels using PyTorch and automatic differentiation (Paszke et al., 2019). We train a kGLM (`sklearn.SGDclassifier`) (Pedregosa et al., 2011) for each $\boldsymbol{\kappa}$ using the same training dataset for training the NN model. All computation was completed on a single A100 GPU with 40GB memory. Details such as specifics of architecture and choice of hyperparameters are available in Appendix H.

Faithful Surrogate Modeling via trNTK. We calculate the τ_K correlation between the surrogate model and underlying NN and report the results in Table 1. We find that the efficacy of our surrogate model as measured by the correlation to the NN changes depending on architecture and dataset; though remarkably, τ_K is consistently high with a lower bound value of 0.7 across all experiments indicating high faithfulness. To demonstrate high τ_K implies we can achieve a point-for-point linear realization of the NN, we learn a non-linear mapping from the kGLM to the NN (Figure 1 for Bert-base. (Additional visualizations for the remainder of experiments available in appendix I.) Finally, we observe that the kGLM with choice of $\boldsymbol{\kappa} = \mathrm{trNTK}$ achieves comparable test accuracy as the underlying NN, which replicates the observations of prior work (Long, 2021; Vyas et al., 2022; Mohamadi & Sutherland, 2022) using our trNTK.

Data Attribution with trNTK. Accepting that the trNTK is a faithful kernel function for a kGLM surrogate model, we can use the data attribution formalism to analyze the importance of individual training datapoints to the classification. In figure 2 we present the visualization of data attribution for one test input, and provide additional visualizations in appendix J.1. The distribution of attribution follows a regular pattern in every visualization generated: the central value of attribution mass for each logit from each class is centered on the distribution of all training data from that class. We emphasize that in no cases have we observed a sparse number of training datapoints dominate the data attribution.

Comparison of Faithfulness between Kernels Functions. For ResNet18 and Bert-base models, we evaluate our choice of trNTK against alternative kernel functions, reporting τ_K and test accuracy differential in Table 2. Across both ResNet18 and Bert-base experiments, we observe that the trNTK forms surrogate models with the highest correlation to the underlying NN decision function and is furthermore consistent in replicating the performance of these networks (TAD nearly 0). The embedding kernel (Em) does not perform as consistently between both tasks, but for its intuitive connection to the internal representation of the neural network may warrant further investigation.

Faithful Surrogates in Data Poisoning Regime. Next, we evaluate whether surrogate models can be extended to analyze network behavior on poisoned data. We train a 21-layer CNN (details available in Appendix H.2.5) using BadNet CIFAR10 data (Gu et al., 2019; Shan et al., 2022). We randomly perturb training data by placing a yellow square in a tenth of training images from CIFAR10 and modify the label of these perturbed images to a targeted label (see example in appendix K). We create

DOI 10.48550/arXiv.2305.14585

Table 1: **Choice of κ = trNTK faithfully forms a surrogate model of underlying NN.** We perform each experiment with '# Models' independent seeds. For each model and dataset we train and extract the trNTK, train a kGLM, then calculate and report the τ_K correlation between the kGLM softmax probability and NN softmax probability for the correct class. The NN test accuracy column shows that training terminates with a highly performant model, and the test accuracy differential (TAD) columns reports the difference between the kGLM test accuracy and the NN test accuracy. We report the leading digit of error (standard error of the mean) as a parenthetical, when available.

Model (Dataset)	# Models	NN test acc (%)	TAD (%)	τ_K
MLP (MNIST2)	100	99.64(1)	+0.03(5)	0.708(3)
CNN (MNIST2)	100	98.4(1)	-0.2(2)	0.857(7)
CNN (CIFAR2)	100	94.94(5)	-2.1(5)	0.711(3)
CNN (FMNIST2)	100	97.95(4)	-2.2(2)	0.882(3)
ResNet18 (CIFAR10)	1	93.07	-0.28	0.776
ResNet34 (CIFAR10)	1	93.33	-0.29	0.786
MobileNetV2 (CIFAR10)	1	93.91	-0.4	0.700
BERT-base (COLA)	4	83.4(1)	-0.1(3)	0.78(2)

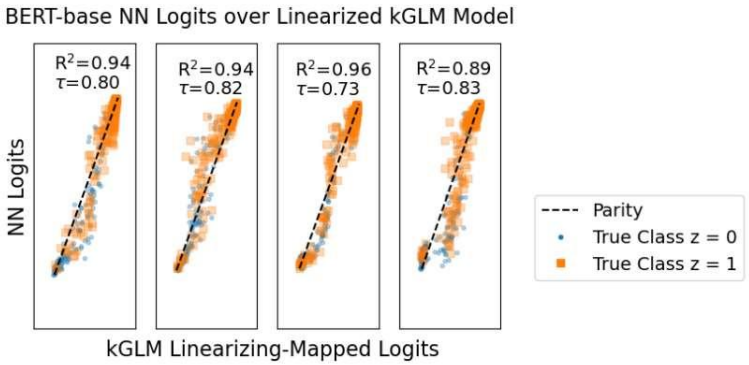


Figure 1: **Linear Realization of Bert-base Model.** Each panel shows a linearization of a Bert-base transfer model, initialized from a different seed. An invertible mapping is fit between the kGLM and NN to transform the kGLM's final activations to the NN's, described in Appendix I. Both τ_K and the Coefficient of Determination (R^2) are shown for each model.

Table 2: **Comparison across surrogate feature spaces.** For ResNet18 and Bert-base experiments we report the faithfulness as τ_K and test-accuracy-differential (TAD) for each kernel function: the trace-NTK (trNTK), unnormalized trace-NTK (trNTK^0, the projection trace NTK (proj-trNTK), the projection psuedo NTK (proj-pNTK), the embedding kernel (Em) and the conjugate kernel (CK). If available, we report leading digit of error (standard error of the mean) as a parenthetical.

Exp Name	Metric	κ					
		trNTK	trNTK^0	proj-trNTK	proj-pNTK	Em	CK
ResNet18	τ_K	0.776	0.658	0.737	0.407	0.768	0.630
	TAD (%)	-0.30	-0.52	-0.20	-0.30	-0.32	-0.20
Bert-base	τ_K	0.809(9)	0.5(1)	0.800(9)	0.72(2)	0.65(2)	0.52(4)
	TAD (%)	+0.1(3)	+0.6(2)	+0.1(2)	+0.5(2)	-0.3(5)	-0.1(1)

a "clean" test dataset from CIFAR10's normal test dataset, and a "poisoned" test dataset by placing yellow squares into each image of CIFAR10's test dataset. At test time, perturbed test data tricks the model into producing labels of the targeted label. We train a model on this poisoned dataset, compute each kernel function, measure faithfulness, and report our results in table 3. We find that the trNTK is most faithful to the NN on the clean test data, but the proj-pNTK is most faithful when evaluated on the poisoned test data. Overall in comparison to the non-poisoned set of experiments

DOI 10.48550/arXiv.2305.14585

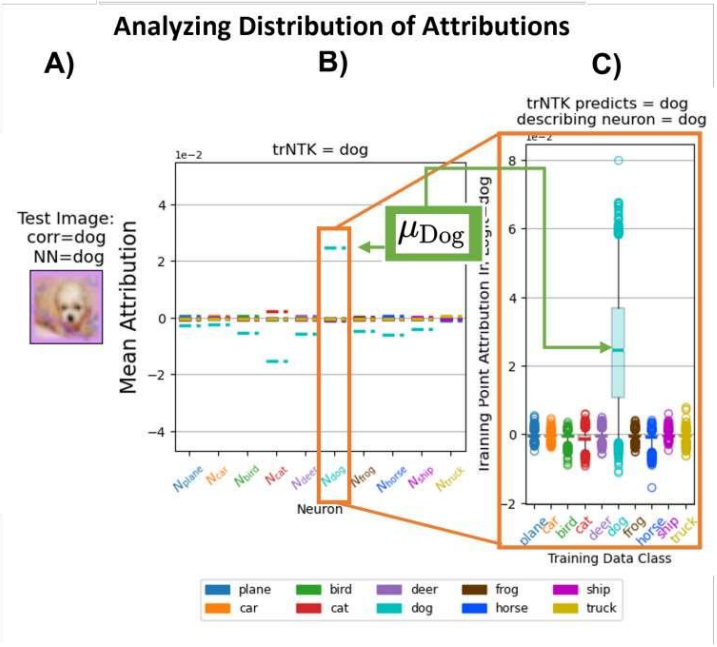


Figure 2: **Overview of Using Kernel Functions for Data Attribution** A) An image from the test dataset of CIFAR10 is chosen. B) We propagate the test image through the NN and plot the mean attribution of the training points from each class for each output neuron. C) Zooming into the neuron representing class "dog", we view the distribution of attributions as a modified box-plot with central lines the mean and outliers shown as flier points. The mean lines are always observed to be within the inner quartile, suggesting that no sparse number of datapoints dominate the central value, and therefore, do not dominate the data attribution.

each kGLM is less faithful, except for the proj-pNTK. We also point out that the kGLM with overall highest faithfulness are the kernel functions with our cosine-normalization applied.

In addition, we show an application of our surrogate modeling approach enabled by kernel-techniques. Forensics models trace NN behavior on unseen poisoned data to the poisoned data source in a training set (Shan et al., 2022). We treat each kernel as a forensic model: for each image in the clean and poisoned test dataset we compute the top 5 most similar training datapoints. If 3/5 of these training datapoints are poisoned we flag the test image as poisoned. In doing so, we can filter poisoned images from clean images. We report the performance of our forensic models using precision and recall (see appendix F) in table 3. Each kernel, except for the conjugate kernel, are all comparable in performance as forensics models. Appendix K provides examples of multiple forensic models acting on poisoned and clean versions of CIFAR10 data.

Table 3: **Poisoned data attribution forensics.** We compute each kernel function between all poisoned training data and the clean test dataset. We report τ_K and TAD between the kGLM and NN for both the poisoned (poi.) and clean set of unseen test images. Finally, we evaluate each kernel as a filter for identifying unseen poisoned data through high similarity to poisoned training data and report the performance as Precision and Recall.

Method	Precision (%)	Recall (%)	τ_K	TAD (%)	poi. τ_K	poi. TAD(%)
trNTK	99.99	99.98	0.585	+0.10	0.441	-1.08
trNTK^0	99.99	99.99	0.518	+0.12	0.543	+0.44
proj-trNTK	99.99	99.97	0.565	+0.09	0.418	+1.3
proj-pNTK	99.99	100.00	0.554	+0.07	0.665	-1.3
Embedding	99.71	100.00	0.430	-2.73	0.261	-13.98
CK	1.65	50.61	0.552	-3.50	0.454	-81.25

DOI 10.48550/arXiv.2305.14585

5 SUMMARY AND CONCLUSIONS

Impact of Linear Surrogate Modeling for Explainability. We have shown evidence supporting the choice of the trNTK as a consistently faithful choice of kernel function for a surrogate model (table 1). We made this determination by measuring the correlation between the kGLM surrogate and the NN, which is an improvement over past methodologies. Our choice of a linear model as surrogate model allows us to separate the attribution terms from each training datapoint, and ensures the central value of the attribution distribution is coupled to the kGLM's logit, and therefore the NN which it approximates (Section 2). We observed that the highest attributed images from the trNTK (and furthermore all evaluate kernel functions) have relatively small mass compared to the bulk contribution, suggesting that the properties of the bulk, rather than a few outliers, are the main source driving decision making. Therefore, presenting the top highest attribution training images without the context of the entire distribution of attribution is probably misleading.

Comparison of Kernel Functions for Surrogate Models. Our quantitative experiments showed the trNTK as more consistently correlated to the NN model compared to the unnormalized trNTK, Embedding kernel, and CK. We observe qualitative differences between these kernel's attributions (appendix J.1) and which training datapoints have highest similarity (appendix K). As a qualitative comparison between kernel functions, in appendix J.2 we visualize the top-5 most similar datapoints evaluated by each kernel function. This further reveals the similarities and differences between kernel functions. Overall, we observe that the trNTK is more sensitive to conceptual similarities between test and train examples than the CK. The embedding kernel is consistently sensitive to background pixel values, though this may be an artifact from our specific choice of layers to sample from. The proj-trNTK, as expected, follows closely with the regular trNTK. These differences could be used to tied to interesting phenomena: for example, because the CK is computed from the final embedding it is likely more sensitive to the effects of neural-collapse (Papyan et al., 2020) than the NTK, which is computed from Jacobians of weight tensors across the entire architecture. We believe this fact explains why the highest similar images measured by the trNTK are more conceptually tied to the specific test image, while the CK has collapsed that inner-class variance away.

Computational Feasibility. Finally, we comment on the computational feasibility of each of the kernel functions. Table 4 reports the time to compute each kernel, and figure 4a shows that the empirical residual distribution between the trNTK and proj-trNTK falls exponentially. The projection-trNTK and projection-pNTK have efficient computation thanks to software made available in Park et al. (2023). The full trNTK is by far the slowest. As implemented, our trNTK computation was layerwise (see appendix C), except in the Poisoning experiment, which we now believe is sub-optimal. Both the trNTK and projection-trNTK computation scales with the number of output neurons linearly, so for models with large output space the projection-pNTK may remain the only feasible option. Finally, because the residuals between the trNTK and proj-trNTK are small and decay rapidly, we believe using the projected variants are well justified. In total, we believe the differences between the trNTK and proj-trNTK are small enough that for low number of outputs, our recommendation is to utilize the proj-trNTK.

Table 4: **Computational Complexity of Large Model Experiments.** We report time to compute each of the trNTK, proj-trNTK, and proj-pNTK for the large model large dataset experiments are shown.

Exp Name	trNTK	proj-trNTK	proj-pNTK
ResNet18	389h	1.12h	7.4m
BertBase	1200h	22m	12m
Poisoning	50h	9.3m	1m

Limitations. Previous works using support vector machines (SVM) kernel surrogate models have reported limitations that we believe extend to kGLM models. We know of two such limitations. We found that SVM surrogate models fail to replicate NN behavior under gradient-based adversarial attacks Appendix G. In addition, SVM surrogate models do not have the same scaling relationships as underlying NNs (Vyas et al., 2022). Our conclusions are limited to kGLM surrogate models; an interesting follow-on work would investigate using kernel functions in K-Nearest Neighbors surrogate models which may recover a sparse explanation.

DOI 10.48550/arXiv.2305.14585

2.0 DOI 10.48550/arXiv.2310.18612

Efficient kernel surrogates for neural network-based regression

Saad Qadeer[1], Andrew Engel[1], Adam Tsou[1,2], Max Vargas[1], Panos Stinis[1,3,4], and Tony Chiang[1,3,5]

[1]Pacific Northwest National Laboratory
[2]Stony Brook University
[3]The University of Washington
[4]Brown University
[5]The University of Texas, El Paso

Abstract

Despite their immense promise in performing a variety of learning tasks, a theoretical understanding of the effectiveness and limitations of Deep Neural Networks (DNNs) has so far eluded practitioners. This is partly due to the inability to determine the closed forms of the learned functions, making it harder to assess their precise dependence on the training data and to study their generalization properties on unseen datasets. Recent work has shown that randomly initialized DNNs in the infinite width limit converge to kernel machines relying on a Neural Tangent Kernel (NTK) with known closed form. These results suggest, and experimental evidence corroborates, that empirical kernel machines can also act as surrogates for finite width DNNs. The high computational cost of assembling the full NTK, however, makes this approach infeasible in practice, motivating the need for low-cost approximations. In the current work, we study the performance of the Conjugate Kernel (CK), an efficient approximation to the NTK that has been observed to yield fairly similar results. For the regression problem of smooth functions and classification using logistic regression, we show that the CK performance is only marginally worse than that of the NTK and, in certain cases, is shown to be superior. In particular, we establish bounds for the relative test losses, verify them with numerical tests, and identify the regularity of the kernel as the key determinant of performance. In addition to providing a theoretical grounding for using CKs instead of NTKs, our framework provides insights into understanding the robustness of the various approximants and suggests a recipe for improving DNN accuracy inexpensively. We present a demonstration of this on the foundation model GPT-2 by comparing its performance on a classification task using a conventional approach and our prescription.

Keywords: Finite-width neural networks, Empirical kernel machines, Neural Tangent Kernel, Function regression, Logistic regression, Generalization errors, Foundation models

1 Introduction

Deep Neural Networks (DNNs) have shown immense promise in performing a variety of learning tasks including, among others, image classification He et al. (2016); Krizhevsky et al. (2012), natural language processing Kenton & Toutanova (2019); Kim (2014); Mikolov et al. (2013); Vaswani et al. (2017), function approximation Daubechies et al. (2022); Yarotsky (2018), solving differential equations Karniadakis et al. (2021); Kovachki et al. (2021); Lu et al. (2021), etc. However, a theoretical understanding of their effectiveness and limitations has proven elusive, thereby hindering

their universal acceptance in practical applications and limiting their usage in scientific computing. This is partly down to their somewhat unwieldy architectures and complex loss landscapes that do not permit precise closed-form characterizations of the optimally learned functions, thus making it harder to assess their precise dependence on the training data and to study their generalization properties on test sets.

Recent work has established that randomly initialized DNNs in the infinite width limit converge to kernel machines relying on a deterministic Neural Tangent Kernel (NTK) Arora et al. (2019); Lee et al. (2019). The NTK for a DNN, defined as the Gram matrix of the Jacobian of the DNN with respect to the network parameters, controls the evolution of the DNN training Jacot et al. (2018). In the infinite width limit, the NTK can be shown to not deviate from its known closed form. As a consequence, the result of fully training an infinite width DNN can be *a priori* identified as a kernel machine employing the known NTK. These insights also allow one to study the performance of the DNN architecture in the extremal over-parameterized limit and benefit from the double-descent phenomenon Bartlett et al. (2021); Belkin (2021); Liu et al. (2020a).

It should be emphasized that this limiting regime does not leverage the changes in the DNN during training and hence fails to make use of any features learned by the architecture. However, these results suggest the tantalizing possibility that empirical kernel machines could also act as surrogates for finite width DNNs. As shown elsewhere Engel et al. (2023) as well as later in the article, under certain assumptions this claim is not only corroborated by strong evidence, both mathematical (equation (27) and Sections 5 and 4) and experimental (Section 6), but also considerably improved upon. This enables a precise determination of the superior learned functions and a thorough study of their performance on unseen data.

The high computational cost of assembling the complete NTK for DNNs employed in practical applications Novak et al. (2022, 2018), however, makes this approach infeasible, motivating the need for low cost approximations. In this article, we study the performance of the Conjugate Kernel (CK) Fan & Wang (2020), a "zeroth-order" approximation to the NTK, that has been observed to yield fairly similar results (see Section 6). For the problems of regression of smooth functions and logistic regression, we prove that the CK approximations are only marginally worse than the NTK approximations and, in some cases, are much superior. In particular, we establish bounds for the errors, verify them with numerical tests, and identify the regularity of the kernel as the key determinant of performance. In addition to providing a theoretical grounding for using CKs instead of NTKs, our framework provides insights into understanding the robustness of the various approximations and suggests a recipe for boosting DNN accuracy inexpensively. We duly present a demonstration of this on GPT-2 by comparing its performance on a classification task using a conventional approach and our prescription (see Section 7).

2 Preliminaries

In this section, we introduce our notation and the define the basic terms and tools. We denote by $f_{\text{NN}} : \mathbb{R}^{d_0} \to \mathbb{R}^{d_{L+1}}$ a fully connected network (FCN) with L hidden layers of the form

$$\begin{aligned} \boldsymbol{z}^{(l)} &= \sigma\left(W^{(l)}\boldsymbol{z}^{(l-1)} + \mathbf{b}^{(l)}\right), \quad 1 \le l \le L, \\ \boldsymbol{z}^{(L+1)} &= W^{(L+1)}\boldsymbol{z}^{(L)} + \mathbf{b}^{(L+1)}. \end{aligned} \tag{7}$$

Here, σ is the activation function, $W^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}}$ and $\mathbf{b}^{(l)} \in \mathbb{R}^{d_l}$ are the trainable parameters, $\boldsymbol{z}^{(0)} \in Z$ is the input, and $\boldsymbol{z}^{(L+1)} \in \mathbb{R}^{d_{L+1}}$ is the output, i.e., $f_{\text{NN}}\left(\boldsymbol{z}^{(0)}\right) = \boldsymbol{z}^{(L+1)}$. We denote all the trainable parameters $\{W^{(l)}, \mathbf{b}^{(l)}\}_{1 \le l \le L+1}$ by θ and their number by $|\theta|$ for brevity, and indicate the dependence of the neural network on them by writing f^{θ}_{NN}. Since this is the only type of neural network (NN) we shall consider in this article, henceforth we refer to FCNs of this type as NNs.

Architectures of this type can be deployed for various tasks. For instance, let $f : Z \to \mathbb{R}$ be a given function, where Z is a compact subset of $\mathbb{R}^{d_0}$, with training input-output pairs $\{(\boldsymbol{x}_i, y_i)\}_{0 \le i \le N} \subset Z \times \mathbb{R}$ (so that $y_i = f(\boldsymbol{x}_i)$). In order to solve the function regression problem, the parameters can be trained so as to minimize the weighted squared loss

$$L_{0,\text{NN}}(\theta) = \sum_{i=0}^{N} w_i \left| y_i - f^{\theta}_{\text{NN}}(\boldsymbol{x}_i) \right|^2, \tag{8}$$

DOI 10.48550/arXiv.2310.18612

for some non-negative weights $\{w_i\}_{0\le i\le N}$, and yield an approximation $f_{\rm NN}^{\theta^*}$ to the target function. We further define the norms

$$\|g\|_0 := \left(\sum_{i=0}^{N} w_i \left|g(\boldsymbol{x}_i)\right|^2\right)^{1/2}, \qquad \|g\|_1 := \left(\sum_{i=0}^{M} v_i \left|g(\boldsymbol{y}_i)\right|^2\right)^{1/2} \tag{9}$$

for any $g : Z \to \mathbb{R}^{d_{L+1}}$, where $\{(\boldsymbol{y}_i, v_i)\}_{0\le i\le M}$ is a collection of test nodes and associated non-negative weights. The training and test errors are then $\left\|f - f_{\rm NN}^{\theta^*}\right\|_0 = [L(\theta^*)]^{1/2}$ and $\left\|f - f_{\rm NN}^{\theta^*}\right\|_1$ respectively.

Similarly, we can employ NNs for classifications tasks. Let $\{(\boldsymbol{x}_i, \chi_i)\}_{0\le i\le N} \subset Z \times \{0,1\}$ be the training set, where χ_i is the class label assigned to $\boldsymbol{x}_i$. Set $d_{L+1} = 2$ (so that $f_{\rm NN}^{\theta} : \mathbb{R}^{d_0} \to \mathbb{R}^2$) and apply the softmax function to get

$$p_{\rm NN}^{\theta}(\boldsymbol{z}) = \frac{\exp\left(f_{{\rm NN},1}^{\theta}(\boldsymbol{z})\right)}{\sum_{i=1}^{2}\exp\left(f_{{\rm NN},i}^{\theta}(\boldsymbol{z})\right)} = \frac{1}{1+\exp\left[-\left(f_{{\rm NN},1}^{\theta}(\boldsymbol{z}) - f_{{\rm NN},2}^{\theta}(\boldsymbol{z})\right)\right]}. \tag{10}$$

This is the likelihood, according to this model, of $\boldsymbol{z} \in Z$ belonging to the class labelled 1. The training requires the minimization of the logistic loss (i.e., the cross-entropy)

$$\ell_{0,{\rm NN}}(\theta) = -\sum_{i=0}^{N} \chi_i \ln\left(p_{\rm NN}^{\theta}(\boldsymbol{x}_i)\right) + (1-\chi_i)\ln\left(1 - p_{\rm NN}^{\theta}(\boldsymbol{x}_i)\right), \tag{11}$$

with the corresponding test loss

$$\ell_{1,{\rm NN}}(\theta) = -\sum_{i=0}^{M} \mu_i \ln\left(p_{\rm NN}^{\theta}(\boldsymbol{y}_i)\right) + (1-\mu_i)\ln\left(1 - p_{\rm NN}^{\theta}(\boldsymbol{y}_i)\right), \tag{12}$$

where $\{\mu_i\}_{0\le i\le M} \subset \{0,1\}$ are the class labels of the test nodes.

These tasks can also be performed with the aid of kernel methods Boser et al. (1992); Meanti et al. (2020). Given a valid kernel function $\ker : Z \times Z \to \mathbb{R}$, define the kernel matrix $H_{\ker} = (\ker(\boldsymbol{x}_i, \boldsymbol{x}_j))_{0\le i,j\le N}$, and the weighted kernel matrix $\widehat{H}_{\ker} = W^{1/2} H_{\ker} W^{1/2}$, where $W^{1/2} = {\rm diag}(\sqrt{w_0}, \ldots, \sqrt{w_N})$. The corresponding kernel approximation for the function regression problem above seeks $\boldsymbol{\delta}^{*,\ker} \in \mathbb{R}^{N+1}$ to minimize

$$L_{0,\ker}(\boldsymbol{\delta}) = \sum_{i=0}^{N} w_i \left| y_i - \sum_{j=0}^{N} \boldsymbol{\delta}_j \ker(\boldsymbol{x}_j, \boldsymbol{x}_i)\right|^2. \tag{13}$$

It possesses the closed form solution

$$f_{\ker}(\boldsymbol{z}) = \boldsymbol{y}^{\top} W^{1/2} \left(\widehat{H}_{\ker}\right)^{\dagger} W^{1/2} \left(\ker(\boldsymbol{x}_j, \boldsymbol{z})\right)_{0\le j\le N}, \tag{14}$$

where $\boldsymbol{y} = (y_i)_{0\le i\le N}$ contains the training target values, with training and test losses $\|f - f_{\ker}\|_0$ and $\|f - f_{\ker}\|_1$ respectively.

The logistic regression problem can likewise be solved by finding an $\boldsymbol{\alpha}^{*,\ker} \in \mathbb{R}^{N+1}$ that minimizes

$$\ell_{0,\ker}(\boldsymbol{\alpha}) = \sum_{i=0}^{N} \chi_i \ln\left[1 + \exp(-H_{\ker,(i,:)}\boldsymbol{\alpha})\right] + (1-\chi_i)\ln\left[1 + \exp(H_{\ker,(i,:)}\boldsymbol{\alpha})\right], \tag{15}$$

thus yielding the likelihood

$$p_{\ker}(\boldsymbol{z}) = \left[1 + \exp\left(-\sum_{j=0}^{N} \ker(\boldsymbol{z}, \boldsymbol{x}_j)\boldsymbol{\alpha}_j^{*,\ker}\right)\right]^{-1}, \tag{16}$$

DOI 10.48550/arXiv.2310.18612

according to this classifier, of $\boldsymbol{z} \in Z$ lying in class 1. The test loss $\ell_{1,\text{ker}}$ is defined similarly to (12) with (10) replaced by (16).

In case ker is induced by a feature map $\Phi_{\text{ker}} : Z \to \mathbb{R}^K$, so that $\text{ker}(\boldsymbol{x}, \boldsymbol{z}) = \Phi_{\text{ker}}(\boldsymbol{x})^\top \Phi_{\text{ker}}(\boldsymbol{z})$, (14) can be understood as an orthogonal projection on $\mathcal{S}_{\text{ker}} = \text{span}\left(\{\Phi_{\text{ker},k}\}_{k=1}^K\right)$ with respect to $\langle \cdot, \cdot \rangle_0$, the inner product that induces $\|\cdot\|_0$; a technique for obtaining an optimal representation of this projection is presented in Appendix B. Equivalently, we can interpret (14) as weighted linear regression on the data-points $\{(\Phi_{\text{ker}}(\boldsymbol{x}_i), y_i)\}_{0 \le i \le N}$, i.e., it implicitly finds $\boldsymbol{\gamma}^{*,\text{ker}} \in \mathbb{R}^K$ that minimizes the training loss

$$\widetilde{L}_{0,\text{ker}}(\boldsymbol{\gamma}) = \sum_{i=0}^{N} w_i \left| y_i - \boldsymbol{\gamma}^\top \Phi_{\text{ker}}(\boldsymbol{x}_i) \right|^2, \tag{17}$$

so the approximation (14) can also be written as

$$f_{\text{ker}}(\boldsymbol{z}) = \left(\boldsymbol{\gamma}^{*,\text{ker}}\right)^\top \Phi_{\text{ker}}(\boldsymbol{z}). \tag{18}$$

This equivalence between the apparently more general form $\boldsymbol{\gamma}^\top \Phi_{\text{ker}}(\boldsymbol{z})$ in (18) and the kernel form $\boldsymbol{\delta}^\top \left(\text{ker}(\boldsymbol{x}_j, \boldsymbol{z})\right)_{0 \le j \le N}$ in (14) can be viewed as a consequence of the representer theorem Schölkopf et al. (2001) or, equivalently, as a corollary of the (easily proven) fact that

$$\text{range}(A^\top A) = \text{range}(A^\top), \tag{19}$$

for any matrix A: applied to the weighted feature map matrix $\widehat{\Xi}_{\text{ker}} = \left(\Phi_{\text{ker},m}(\boldsymbol{x}_i) w_i^{1/2}\right)_{1 \le m \le K, 0 \le i \le N} \in \mathbb{R}^{K \times (N+1)}$, we deduce that for an optimal $\boldsymbol{\gamma}^{*,\text{ker}}$, there exists some $\boldsymbol{\delta}^{*,\text{ker}} \in \mathbb{R}^{N+1}$ such that

$$\widehat{\Xi}_{\text{ker}}^\top \boldsymbol{\gamma}^{*,\text{ker}} = \widehat{\Xi}_{\text{ker}}^\top \widehat{\Xi}_{\text{ker}} \boldsymbol{\delta}^{*,\text{ker}} = \widehat{H}_{\text{ker}} \boldsymbol{\delta}^{*,\text{ker}}, \tag{20}$$

since $\widehat{H}_{\text{ker}} = \widehat{\Xi}_{\text{ker}}^\top \widehat{\Xi}_{\text{ker}}$, and hence that we can restrict our search for $\boldsymbol{\gamma}^{*,\text{ker}}$ in range $\left(\widehat{\Xi}_{\text{ker}}\right)$. This implies that

$$\|f - f_{\text{ker}}\|_\nu^2 = \widetilde{L}_{\nu,\text{ker}}(\boldsymbol{\gamma}^{*,\text{ker}}), \tag{21}$$

for $\nu = 0, 1$, where the test loss $\widetilde{L}_{1,\text{ker}}$ is defined in the same manner as (17) with the test points and weights replacing the training ones.

Similarly, solving the kernel logistic problem can be interpreted as a linear classifier in the feature space, i.e., identifying the decision boundary

$$\left\{ \boldsymbol{x} \in \mathbb{R}^{d_0} : \left(\boldsymbol{\beta}^{*,\text{ker}}\right)^\top \Phi_{\text{ker}}(\boldsymbol{x}) = 0 \right\}, \tag{22}$$

which can be seen as a hyperplane in $\mathbb{R}^K$, by finding $\boldsymbol{\beta}^{*,\text{ker}} \in \mathbb{R}^K$ that minimizes

$$\widetilde{\ell}_{0,\text{ker}}(\boldsymbol{\beta}) = \sum_{i=0}^{N} \chi_i \ln\left[1 + \exp\left(-\boldsymbol{\beta}^\top \Phi_{\text{ker}}(\boldsymbol{x}_i)\right)\right] + (1 - \chi_i) \ln\left[1 + \exp\left(\boldsymbol{\beta}^\top \Phi_{\text{ker}}(\boldsymbol{x}_i)\right)\right]. \tag{23}$$

Following an argument similar to that used above, we have

$$\ell_{\nu,\text{ker}}(\boldsymbol{\alpha}^{*,\text{ker}}) = \widetilde{\ell}_{\nu,\text{ker}}(\boldsymbol{\beta}^{*,\text{ker}}), \tag{24}$$

for $\nu = 0, 1$.

3 Problem formulation

Given a (fully or partially trained or untrained) neural network $f_{\text{NN}} : \mathbb{R}^{d_0} \to \mathbb{R}$, the corresponding Neural Tangent Kernel (NTK) is the kernel induced by the feature map

$$\Phi_{\text{NTK},k} := \frac{\partial f_{\text{NN}}^\theta}{\partial \theta_k}, \quad 1 \le k \le |\theta|, \tag{25}$$

DOI 10.48550/arXiv.2310.18612

i.e., the Jacobian of the neural network with respect to the trainable parameters Jacot et al. (2018). In the infinite width limit, it has been established that, over the course of training a randomly initialized NN, the NTK possesses an unchanging closed form, and that the NN converges to the corresponding NTK machine Lee et al. (2019). This highlights the linear dependence of the NN on the underlying parameters, to wit,

$$f^{\theta}_{\text{NN}} = f^{\theta_0}_{\text{NN}} + (\theta - \theta_0)^{\top} \left(\frac{\partial f^{\theta}_{\text{NN}}}{\partial \theta_k} \right) \Bigg|_{\theta=\theta_0}. \tag{26}$$

In fact, since $f^{\theta_0}_{\text{NN}}$ is a linear combination of the neurons in the last hidden layer (from (7)) and which correspond the Jacobian entries corresponding to the last layer of parameters, it can be subsumed in the second term in (26) to yield

$$f^{\theta}_{\text{NN}} = \boldsymbol{\gamma}^{\top} \Phi_{\text{NTK}} \tag{27}$$

for some $\boldsymbol{\gamma} \in \mathbb{R}^{|\theta|}$. Since this is precisely the form of an NTK approximation (from the equivalence of (14) and (18)), we deduce that, in the infinite width limit, a fully trained NN is identical to an optimal regressor employing the NTK.

However, using this insight to obtain performance guarantees and assess limitations of finite width network runs into two problems. First, the assumption of linear dependence on parameters in (26) does not hold Liu et al. (2020b), so the optimal NTK machine can only be viewed as the best approximator in the tangent space of the NN at θ_0, and not on the manifold of all NNs. Secondly, computing the NTK in practice is a computationally daunting task Novak et al. (2022, 2018). In the infinite width regime and under random initialization, a closed form for the NTK can be calculated. However, empirical kernels do not possess such characterizations in general, necessitating the use of low-cost approximations.

The Conjugate Kernel (CK) is defined as the kernel induced by the Jacobian only with respect to the parameters $\{W^{(L+1)}, \mathbf{b}^{(L+1)}\}$ in the last layer Fan & Wang (2020). Consequently, its feature map can be seen as a truncation of the full Jacobian employed by the NTK. Observe that we have set $d_{L+1} = 1$ so $W^{(L+1)} \in \mathbb{R}^{d_L}$ and $\mathbf{b}^{L+1} \in \mathbb{R}$, and hence

$$\Phi_{\text{CK},k} := \begin{cases} \frac{\partial f^{\theta}_{\text{NN}}}{\partial W^{(L+1)}_k}, & \text{if } 1 \leq k \leq d_L, \\ \\ \frac{\partial f^{\theta}_{\text{NN}}}{\partial \mathbf{b}^{(L+1)}}, & \text{if } k = d_L + 1. \end{cases} \tag{28}$$

A moment's thought then reveals that

$$\text{CK}(\boldsymbol{z}, \widetilde{\boldsymbol{z}}) := \Phi_{\text{CK}}(\boldsymbol{z})^{\top} \Phi_{\text{CK}}(\widetilde{\boldsymbol{z}}) = 1 + \boldsymbol{z}^{(L)} \cdot \widetilde{\boldsymbol{z}}^{(L)}. \tag{29}$$

Remark 3.1 *For an NN trained for the binary classification problem described in the previous section, we have $d_{L+1} = 2$. In this setting, we define the NTK and CK for $f_{diff} := f^{\theta}_{NN,1} - f^{\theta}_{NN,2}$ since, as shown in* (10)*, this difference dictates the output of the logistic predictor.*

Denoting by E the contribution towards the NTK from every layer but the last allows us to write

$$\text{NTK}(\boldsymbol{z}, \widetilde{\boldsymbol{z}}) = \text{CK}(\boldsymbol{z}, \widetilde{\boldsymbol{z}}) + \text{E}(\boldsymbol{z}, \widetilde{\boldsymbol{z}}). \tag{30}$$

The CK can therefore be viewed as a "zeroth-order" approximation to the NTK. All these components are valid kernels that can be calculated from a given NN. However, the cost of computing the CK at a pair of points is the same as that of evaluating the NN at those points since, according to (29), it makes use of the neuron values at the last hidden layer. Moreover, numerical evidence suggests that, while the NTK expectedly outperforms the CK on test sets, the improvement is only marginal and is dwarfed by the superiority both kernels exhibit over the NN architecture from which they are derived (see Section 6). In other words, the CK appears to yield substantive gains in accuracy over a given finite width neural network while being easier to calculate than the NTK and much cheaper to train further than the NN. This raises the intriguing possibility that the CK can serve as a low-cost proxy for the NTK, i.e., lead to a comparable boost in performance over the NN, while being significantly less expensive to assemble than the NTK and considerably easier to train than

DOI 10.48550/arXiv.2310.18612

the NN. Since training the CK is the same as further training the last parameter layer of the NN, our work provides a recipe for boosting NN accuracy and also sheds light on the robustness (or lack thereof) conferred on the approximations by the choice of the activation function.

In the following sections, we provide a mathematical treatment of this phenomenon for two types of problems: regression of smooth functions and classification using logistic regression. For ease of exposition and analysis, we concentrate on these problems in low dimensions with structured training and test nodes. The analyses of these problems are presented in Sections 4 and 5, and are complemented by numerical experiments presented in Section 6.

4 Function regression

4.1 Preliminaries

We consider the approximation problem for a smooth function $f : [a, b] \to \mathbb{R}$ whose values are known at the equispaced training nodes $x_j = a + j\Delta x$ for $0 \le j \le N$, where $\Delta x = (b - a)/N$. The performance is assessed on the equispaced test nodes $y_j = a + j\Delta y$ for $0 \le j \le M$, where $\Delta y = (b - a)/M$; we take $M = \tau N$, for some integer $\tau > 1$, and set

$$w_i = \begin{cases} \Delta x, & 1 \le i \le N-1 \\ \Delta x/2, & i = 0, N \end{cases}, \qquad v_i = \begin{cases} \Delta y, & 1 \le i \le M-1 \\ \Delta y/2, & i = 0, M \end{cases}, \tag{31}$$

so for any $g : [a, b] \to \mathbb{R}$, we have

$$\|g\|_0 = \left(\frac{\Delta x}{2} \sum_{j=0}^{N-1} \left(g(x_j)^2 + g(x_{j+1})^2 \right) \right)^{1/2}, \quad \|g\|_1 = \left(\frac{\Delta y}{2} \sum_{j=0}^{M-1} \left(g(y_j)^2 + g(y_{j+1})^2 \right) \right)^{1/2}. \tag{32}$$

These can be recognized as the trapezoidal rule approximations to the L^2 norm on $[a, b]$; recall that this approximation is second-order accurate with respect to the step-size in general and spectrally accurate for periodic functions. Moreover, these choices serve to considerably simplify the analysis: for instance, since $y_{\tau j} = x_j$ for $0 \le j \le N$, we have, for any function g,

$$\|g\|_0^2 = \frac{\Delta x}{2} \sum_{j=0}^{N-1} \left(g(x_j)^2 + g(x_{j+1})^2 \right) \quad \le \quad \frac{\Delta x}{2} \sum_{j=0}^{N-1} \sum_{i=0}^{\tau} g(y_{\tau j+i})^2 = \left(\frac{M}{N} \right) \|g\|_1^2$$

so that

$$\|g\|_0 \quad \le \quad \tau^{1/2} \|g\|_1 . \tag{33}$$

This straightforward result allows us to describe the size of the function on the lower-resolution training grid in terms of the higher-resolution test grid. The next two results enable us to move in the opposite direction, i.e., bounding the norm on the finer grid by the norm on the coarser grid. This naturally requires some assumptions about the behaviour of the function away from the training nodes; the following lemmas consider two different settings.

4.2 Two Lemmas

Lemma 4.1 *Suppose that g is monotonic on every sub-interval $[x_i, x_{i+1}]$. Then,*

$$\|g\|_1 \le \sqrt{2} \|g\|_0 . \tag{34}$$

Proof: The monotonicity of g on every sub-interval implies

$$|g(y_{\tau i+k})| \le \max \{ |g(x_i)|, |g(x_{i+1})| \}, \quad 1 \le k \le \tau - 1, \; 0 \le i \le N - 1, \tag{35}$$

DOI 10.48550/arXiv.2310.18612

with the result

$$\begin{aligned}
\|g\|_1^2 &= \frac{\Delta y}{2}\sum_{j=0}^{M-1} g(y_j)^2 + g(y_{j+1})^2 \\
&= \frac{\Delta y}{2}\sum_{j=0}^{N-1}\left[g(x_j)^2 + g(x_{j+1})^2\right] + \frac{\Delta y}{2}\sum_{i=0}^{N-1}\sum_{k=1}^{\tau-1} g(y_{\tau i+k})^2 \\
&\leq \frac{\Delta y}{2}\sum_{j=0}^{N-1}\left[g(x_j)^2 + g(x_{j+1})^2\right] + (\tau-1)\frac{\Delta y}{2}\sum_{i=0}^{N-1}\max\left\{g(x_i)^2, g(x_{i+1})^2\right\} \\
&\leq \frac{1}{\tau}\|g\|_0^2 + \frac{2(\tau-1)}{\tau}\|g\|_0^2 \\
&\leq 2\|g\|_0^2,
\end{aligned}$$

and hence $\|g\|_1 \leq \sqrt{2}\,\|g\|_0$. ∎

Combining this with (33), we have the norm equivalence

$$\tau^{-1/2}\|g\|_0 \leq \|g\|_1 \leq \sqrt{2}\,\|g\|_0\,. \tag{36}$$

Lemma 4.2 *Suppose that g is Lipschitz on $[a,b]$ with Lipschitz constant L_g. Then,*

$$\|g\|_1^2 \leq 4\left(\|g\|_0^2 + \frac{2(b-a)^2 L_g^2}{N^2}\right). \tag{37}$$

Proof: The Lipschitz property implies that

$$|g(y_{\tau j+k})| \leq \max\left\{|g(x_j)| + kL_g\Delta y, |g(x_{j+1})| + (\tau-k)L_g\Delta y\right\}$$

for $0 \leq k \leq \tau$. It follows that

$$\begin{aligned}
g(y_{\tau j+k})^2 &\leq 2\left[g(x_j)^2 + k^2L_g^2(\Delta y)^2 + g(x_{j+1})^2 + (\tau-k)^2L_g^2(\Delta y)^2\right] \\
&\leq 2\left[g(x_j)^2 + g(x_{j+1})^2 + 2L_g^2(\Delta x)^2\right],
\end{aligned}$$

so for $0 \leq l \leq \tau-1$, we have

$$g(y_{\tau j+l})^2 + g(y_{\tau j+l+1})^2 \leq 4\left[g(x_j)^2 + g(x_{j+1})^2 + 2L_g^2(\Delta x)^2\right].$$

As a result,

$$\begin{aligned}
\sum_{l=0}^{\tau-1} g(y_{\tau j+l})^2 + g(y_{\tau j+l+1})^2 &\leq 4\tau\left[g(x_j)^2 + g(x_{j+1})^2 + 2L_g^2(\Delta x)^2\right] \\
\Rightarrow \sum_{j=0}^{N-1}\sum_{l=0}^{\tau-1} g(y_{\tau j+l})^2 + g(y_{\tau j+l+1})^2 &\leq 4\tau\sum_{j=0}^{N-1}\left[g(x_j)^2 + g(x_{j+1})^2 + 2L_g^2(\Delta x)^2\right],
\end{aligned}$$

and hence

$$M\|g\|_1^2 \leq 4N\tau\left[\|g\|_0^2 + 2L_g^2(\Delta x)^2\right] \Rightarrow \|g\|_1^2 \leq 4\left(\|g\|_0^2 + \frac{2(b-a)^2L_g^2}{N^2}\right).$$

∎

4.3 Approximation properties

For a smooth target function $f : [a,b] \to \mathbb{R}$, let f_{CK} and f_{NTK} be the CK and NTK approximations to f with respect to the training nodes. Recall that (14) can be viewed as an orthogonal projection $\mathcal{P}_{\text{ker}}$ on $\mathcal{S}_{\text{ker}}$, the span of the feature map components corresponding to the kernel. Since $\mathcal{S}_{\text{CK}} \subset \mathcal{S}_{\text{NTK}}$, we have

$$\|f - f_{\text{NTK}}\|_0 \leq \|f - f_{\text{CK}}\|_0\,. \tag{38}$$

DOI 10.48550/arXiv.2310.18612

Next, we establish that the error of the CK approximation may also be controlled by that of the NTK approximation for most target functions. More precisely, we define, for any $\beta \leq 1$

$$\mathcal{F}_\beta = \{f : [a,b] \to \mathbb{R} \text{ is smooth with } \|\mathcal{P}_{\text{NTK}} f\|_0 \leq \beta \|f\|_0\}\,. \tag{39}$$

We note that since $\mathcal{P}_{\text{NTK}}$ is an orthogonal projection, its norm is one and so $\mathcal{F}_1$ contains all smooth real-valued functions. However, most functions of interest would not lie in $\mathcal{S}_{\text{NTK}}$ and hence would belong to some $\mathcal{F}_\beta$ with $\beta < 1$.

Lemma 4.3 *Let $f \in \mathcal{F}_\beta$. Then,*

$$\|\mathcal{P}_{NTK}(f - f_{CK})\|_0 \leq \beta \|f - f_{CK}\|_0\,. \tag{40}$$

Proof: We have

$$\|f\|_0^2 = \|\mathcal{P}_{\text{NTK}} f\|_0^2 + \|(I - \mathcal{P}_{\text{NTK}})f\|_0^2 = \|f_{\text{NTK}}\|_0^2 + \|f - f_{\text{NTK}}\|_0^2 \tag{41}$$

and

$$\begin{aligned} \|f - f_{\text{CK}}\|_0^2 &= \|\mathcal{P}_{\text{NTK}}(f - f_{\text{CK}})\|_0^2 + \|(I - \mathcal{P}_{\text{NTK}})(f - f_{\text{CK}})\|_0^2 \\ &= \|f_{\text{NTK}} - f_{\text{CK}}\|_0^2 + \|f - f_{\text{NTK}}\|_0^2 \end{aligned} \tag{42}$$

since $(I - \mathcal{P}_{\text{NTK}}) f_{\text{CK}} = 0$ due to $\mathcal{P}_{\text{NTK}} \mathcal{P}_{\text{CK}} = \mathcal{P}_{\text{CK}}$. Combining (41) and (42) yields

$$\begin{aligned} \frac{\|f - f_{\text{CK}}\|_0^2}{\|f\|_0^2} &= \frac{\|f_{\text{NTK}} - f_{\text{CK}}\|_0^2 + \|f - f_{\text{NTK}}\|_0^2}{\|f_{\text{NTK}}\|_0^2 + \|f - f_{\text{NTK}}\|_0^2} \\ &\geq \frac{\|f_{\text{NTK}} - f_{\text{CK}}\|_0^2}{\|f_{\text{NTK}}\|_0^2} \end{aligned}$$

where we used the elementary inequality

$$\frac{\epsilon + \zeta}{\rho + \zeta} \geq \frac{\epsilon}{\rho}, \tag{43}$$

for $0 < \epsilon \leq \rho$ and $\zeta \geq 0$. As a result,

$$\begin{aligned} \|f_{\text{NTK}} - f_{\text{CK}}\|_0^2 &\leq \left(\frac{\|f_{\text{NTK}}\|_0^2}{\|f\|_0^2}\right) \|f - f_{\text{CK}}\|_0^2 \\ &\leq \beta^2 \|f - f_{\text{CK}}\|_0^2 \\ \Leftrightarrow \|\mathcal{P}_{\text{NTK}}(f - f_{\text{CK}})\|_0 &\leq \beta \|f - f_{\text{CK}}\|_0 \qquad (\because \mathcal{P}_{\text{NTK}} \mathcal{P}_{\text{CK}} = \mathcal{P}_{\text{CK}}). \end{aligned}$$

∎

Lemma 4.4 *For any function $f \in \mathcal{F}_\beta$ with $\beta < 1$,*

$$\|f - f_{CK}\|_0 \leq \frac{1}{1 - \beta} \|f - f_{NTK}\|_0\,.$$

Proof: We have

$$\begin{aligned} \|f - f_{\text{CK}}\|_0 &\leq \|f - f_{\text{NTK}}\|_0 + \|f_{\text{NTK}} - f_{\text{CK}}\|_0 \\ &= \|f - f_{\text{NTK}}\|_0 + \|\mathcal{P}_{\text{NTK}}(f - f_{\text{CK}})\|_0 \qquad (\because \mathcal{P}_{\text{NTK}} \mathcal{P}_{\text{CK}} = \mathcal{P}_{\text{CK}}) \\ &\leq \|f - f_{\text{NTK}}\|_0 + \beta \|f - f_{\text{CK}}\|_0 \qquad (\because \text{Lemma 4.3}) \\ \Rightarrow \|f - f_{\text{CK}}\|_0 &\leq \frac{1}{1 - \beta} \|f - f_{\text{NTK}}\|_0\,. \end{aligned} \tag{44}$$

∎

For such functions, we have the following approximation results.

Theorem 4.1 *Let $f \in \mathcal{F}_\beta$ for some $\beta < 1$ and suppose that both $(f - f_{CK})$ and $(f - f_{NTK})$ are monotonic on every sub-interval $[x_i, x_{i+1}]$ for $0 \leq i \leq N - 1$. Then, there exist constants $C_1, C_2 \geq 1$ such that*

$$C_1^{-1} \|f - f_{NTK}\|_1 \leq \|f - f_{CK}\|_1 \leq C_2 \|f - f_{NTK}\|_1\,. \tag{45}$$

DOI 10.48550/arXiv.2310.18612

Proof: We have

$$\begin{aligned} \|f - f_{\text{NTK}}\|_1 &\leq \sqrt{2}\,\|f - f_{\text{NTK}}\|_0 \qquad (\because \text{Lemma 4.1}) \\ &\leq \sqrt{2}\,\|f - f_{\text{CK}}\|_0 \qquad (\because (38)) \\ &\leq \sqrt{2\tau}\,\|f - f_{\text{CK}}\|_1 \qquad (\because (33)). \end{aligned} \tag{46}$$

Similarly,

$$\begin{aligned} \|f - f_{\text{CK}}\|_1 &\leq \sqrt{2}\,\|f - f_{\text{CK}}\|_0 \qquad (\because \text{Lemma 4.1}) \\ &\leq \frac{\sqrt{2}}{1-\beta}\,\|f - f_{\text{NTK}}\|_0 \qquad (\because \text{Lemma 4.4}) \\ &\leq \frac{\sqrt{2\tau}}{1-\beta}\,\|f - f_{\text{NTK}}\|_1 \qquad (\because (33)). \end{aligned} \tag{47}$$

Setting $C_1 = \sqrt{2\tau}$ and $C_2 = \frac{\sqrt{2\tau}}{1-\beta}$ and combining (46) and (47) yields the desired (45). ∎

Note that Theorem 4.1 requires the residuals to be monotonic which may be too stringent. The following results rests on a more relaxed assumption.

Theorem 4.2 *Let $f \in \mathcal{F}_\beta$ for some $\beta < 1$ and suppose that both $(f - f_{CK})$ and $(f - f_{NTK})$ are Lipschitz, with Lipschitz constants L_{CK} and L_{NTK}. Then, there exist constants $D_1, D_2 \geq 1$ such that*

$$\|f - f_{NTK}\|_1^2 \leq D_1\,\|f - f_{CK}\|_1^2 + \frac{8(b-a)^2 L_{NTK}^2}{N^2} \tag{48}$$

and

$$\|f - f_{CK}\|_1^2 \leq D_2\,\|f - f_{NTK}\|_1^2 + \frac{8(b-a)^2 L_{CK}^2}{N^2}. \tag{49}$$

Proof: We have

$$\begin{aligned} \|f - f_{\text{NTK}}\|_1^2 &\leq 4\left(\|f - f_{\text{NTK}}\|_0^2 + \frac{2(b-a)^2 L_{\text{NTK}}^2}{N^2}\right) \qquad (\because \text{Lemma 4.2}) \\ &\leq 4\left(\|f - f_{\text{CK}}\|_0^2 + \frac{2(b-a)^2 L_{\text{NTK}}^2}{N^2}\right) \qquad (\because (38)) \\ &\leq 4\left(\tau\,\|f - f_{\text{CK}}\|_1^2 + \frac{2(b-a)^2 L_{\text{NTK}}^2}{N^2}\right) \qquad (\because (33)). \end{aligned} \tag{50}$$

In addition,

$$\begin{aligned} \|f - f_{\text{CK}}\|_1^2 &\leq 4\left(\|f - f_{\text{CK}}\|_0^2 + \frac{2(b-a)^2 L_{\text{CK}}^2}{N^2}\right) \qquad (\because \text{Lemma 4.2}) \\ &\leq 4\left(\frac{1}{(1-\beta)^2}\,\|f - f_{\text{NTK}}\|_0^2 + \frac{2(b-a)^2 L_{\text{CK}}^2}{N^2}\right) \qquad (\because \text{Lemma 4.4}) \\ &\leq 4\left(\frac{\tau}{(1-\beta)^2}\,\|f - f_{\text{NTK}}\|_1^2 + \frac{2(b-a)^2 L_{\text{CK}}^2}{N^2}\right) \qquad (\because (33)). \end{aligned} \tag{51}$$

Setting $D_1 = 4\tau$ and $D_2 = \frac{4\tau}{(1-\beta)^2}$ yields the desired (48) and (49). ∎

5 Logistic regression

5.1 Preliminaries

In this section, we compare the performance of the two kernels under consideration for logistic regression. For brevity of exposition, we consider the problem in two dimensions. Let $\{\boldsymbol{x}_{i,j}\}_{0\leq i\leq N_1, 0\leq j\leq N_2} \subset Z = [a_1, b_1] \times [a_2, b_2]$ be the training points given by

$$\boldsymbol{x}_{i,j} = (a_1 + i\Delta x_1, a_2 + j\Delta x_2),$$

DOI 10.48550/arXiv.2310.18612

where $\Delta x_1 = (b_1 - a_1)/N_1$ and $\Delta x_2 = (b_2 - a_2)/N_2$. We further set $h = \left(\Delta x_1^2 + \Delta x_2^2\right)^{1/2}$. The test nodes $\{\boldsymbol{y}_{i,j}\}_{0 \le i \le M_1, 0 \le j \le M_2}$ are similarly chosen as

$$\boldsymbol{y}_{i,j} = (a_1 + i\Delta y_1, a_2 + j\Delta y_2),$$

where $\Delta y_1 = (b_1 - a_1)/M_1$ and $\Delta y_2 = (b_2 - a_2)/M_2$. We also assume that $\tau_k = M_k/N_k$ is an integer greater than one for $k = 1, 2$.

Following Section 2, each training and test point is assigned a class labelled 0 or 1, and denoted respectively by χ_{ij} for every $0 \le i \le N_1, 0 \le j \le N_2$ and μ_{ij} for each $0 \le i \le M_1, 0 \le j \le M_2$. Given a kernel function $\ker : Z \times Z \to \mathbb{R}$, we can define the kernel matrix

$$H_{\ker} = (\ker(\boldsymbol{x}_{i,j}, \boldsymbol{x}_{k,l}))_{0 \le i,k \le N_1, 0 \le j,l \le N_2} \in \mathbb{R}^{N_1 N_2 \times N_1 N_2},$$

with the indices arranged suitably.

We note first that the NTK must necessarily outperform the CK on the training points, i.e.,

$$\ell_{0,\mathrm{NTK}}\left(\boldsymbol{\alpha}^{*,\mathrm{NTK}}\right) \le \ell_{0,\mathrm{CK}}\left(\boldsymbol{\alpha}^{*,\mathrm{CK}}\right). \tag{52}$$

This is most easily seen by rewriting (52) in terms of the feature maps: since

$$\ell_{0,\mathrm{CK}}\left(\boldsymbol{\alpha}^{*,\mathrm{CK}}\right) = \widetilde{\ell}_{0,\mathrm{CK}}\left(\boldsymbol{\beta}^{*,\mathrm{CK}}\right),$$

from (24), and $\Phi_{\mathrm{NTK}} = \begin{pmatrix}\Phi_{\mathrm{CK}} & \Phi_{\mathrm{E}}\end{pmatrix}^{\top}$, setting $\boldsymbol{\gamma} = \begin{pmatrix}\boldsymbol{\beta}^{*,\mathrm{CK}} & \boldsymbol{0}\end{pmatrix}^{\top} \in \mathbb{R}^{|\theta|}$ yields

$$\ell_{0,\mathrm{CK}}\left(\boldsymbol{\alpha}^{*,\mathrm{CK}}\right) = \widetilde{\ell}_{0,\mathrm{CK}}\left(\boldsymbol{\beta}^{*,\mathrm{CK}}\right) = \widetilde{\ell}_{0,\mathrm{NTK}}\left(\boldsymbol{\gamma}\right) \ge \widetilde{\ell}_{0,\mathrm{NTK}}\left(\boldsymbol{\beta}^{*,\mathrm{NTK}}\right) = \ell_{0,\mathrm{NTK}}\left(\boldsymbol{\alpha}^{*,\mathrm{NTK}}\right). \tag{53}$$

In order to show that an equivalence result in the manner of Theorem 4.1 holds for the test errors, we follow a similar strategy to that for the function regression problem in Section 4:

(a) identify two cases that permit a transfer of information from training to test nodes;

(b) extend (52) to an equivalence result on the training nodes for the two kernels

These ingredients can then be combined to yield the desired conclusion.

5.2 Another pair of lemmas

We begin by noting that

$$\ell_{0,\ker}(\boldsymbol{\alpha}^{*,\ker}) \le \ell_{1,\ker}(\boldsymbol{\alpha}^{*,\ker}) \tag{54}$$

simply by virtue of the fact that every training point is also a test point. To complete (a), we need to bound the error on the finer grid by that on the coarser grid. The quantity of interest is the linear combination of the feature map components

$$\psi_{\ker}(\boldsymbol{z}) = \sum_{m=1}^{|\theta|} \Phi_{\ker,m}(\boldsymbol{z})\boldsymbol{\beta}_m^{*,\ker} \tag{55}$$

as it determines the size of the gap between $\boldsymbol{z} \in Z$ and the separating hyperplane in the feature space. However, we also need to keep track of the class label that should be associated with $\boldsymbol{z}$ as that determines how it shows up in the loss function (23). Let $\eta : Z \to \{0, 1\}$ be the true class label for every point (so that, e.g., $\chi_{ij} = \eta(\boldsymbol{x}_{i,j})$ and $\mu_{ij} = \eta(\boldsymbol{y}_{i,j})$), and set

$$\widehat{\psi}_{\ker}(\boldsymbol{z}) = (1 - 2\eta(\boldsymbol{z}))\,\psi_{\ker}(\boldsymbol{z}). \tag{56}$$

Using this function, we can write the minimized training and test losses as simply

$$\widetilde{\ell}_{0,\ker} = \sum_{i=0}^{N_1}\sum_{j=0}^{N_2} \ln\left[1 + \exp\left(\widehat{\psi}_{\ker}(\boldsymbol{x}_{i,j})\right)\right] \tag{57}$$

DOI 10.48550/arXiv.2310.18612

and

$$\widetilde{\ell}_{1,\text{ker}} = \sum_{i=0}^{M_1}\sum_{j=0}^{M_2} \ln\left[1+\exp\left(\widehat{\psi}_{\text{ker}}(\boldsymbol{y}_{i,j})\right)\right], \tag{58}$$

where we have suppressed the dependence on $\boldsymbol{\beta}^{*,\text{ker}}$ for clarity of exposition. For any $0 \le i \le N_1-1$ and $0 \le j \le N_2-1$, we also define

$$\Lambda_{ij} = \{\boldsymbol{x}_{i,j}, \boldsymbol{x}_{i+1,j}, \boldsymbol{x}_{i,j+1}, \boldsymbol{x}_{i+1,j+1}\}$$

to make it easier to navigate the training grid.

The two regimes that interest us are a corner maximum condition and Lipschitz continuity. We make these precise in the following lemmas.

Lemma 5.1 *Suppose that on every rectangle with corners Λ_{ij}, the function $\widehat{\psi}_{ker}$ achieves its maximum value on Λ_{ij}. Then,*

$$\widetilde{\ell}_{1,ker} \le \tau_1\tau_2\widetilde{\ell}_{0,ker}, \tag{59}$$

Proof: Every test point $\boldsymbol{y}_{k,l}$ belongs to some rectangle with corners $\Lambda_{k'l'}$. The contribution from this point to the test loss is then dominated by the contributions from the corner points, i.e.,

$$\ln\left[1+\exp\left(\widehat{\psi}_{\text{ker}}(\boldsymbol{y}_{k,l})\right)\right] \le \max_{k'\le i\le k'+1, l'\le j\le l'+1} \ln\left[1+\exp\left(\widehat{\psi}_{\text{ker}}(\boldsymbol{x}_{i,j})\right)\right], \tag{60}$$

with the result that

$$\sum_{k=0}^{M_1}\sum_{l=0}^{M_2} \ln\left[1+\exp\left(\widehat{\psi}_{\text{ker}}(\boldsymbol{y}_{k,l})\right)\right] \le \left(\frac{M_1}{N_1}\right)\left(\frac{M_2}{N_2}\right)\sum_{i=0}^{N_1}\sum_{j=0}^{N_2} \ln\left[1+\exp\left(\widehat{\psi}_{\text{ker}}(\boldsymbol{x}_{i,j})\right)\right],$$

and hence

$$\widetilde{\ell}_{1,\text{ker}} \le \tau_1\tau_2\widetilde{\ell}_{0,\text{ker}}. \qquad \blacksquare$$

We say that the labelling function η possesses the *matching property* with respect to the given training and test nodes if every test point $\boldsymbol{y}_{k,l}$ is contained in an enclosing rectangle with corners $\Lambda_{k'l'}$ such that η agrees with $\eta(\boldsymbol{y}_{k,l})$ on at least one corner point. In other words, we do not have the matching property if and only if there exists some test point $\boldsymbol{y}_{k,l}$ such that, for any rectangle with corners $\Lambda_{k'l'}$ that may enclose it, we have $\eta(\boldsymbol{y}_{k,l}) \notin \eta(\Lambda_{k'l'})$.

Lemma 5.2 *Suppose that ψ_{ker} is Lipschitz continuous on D, with Lipschitz constant L_{ker}, and η possesses the matching property. Then,*

$$\widetilde{\ell}_{1,ker} \le \exp(hL_{ker})\tau_1\tau_2\widetilde{\ell}_{0,ker}. \tag{61}$$

Proof: For any test point $\boldsymbol{y}_{k,l}$, let $\Lambda_{k'l'}$ contain the corners of the enclosing rectangle that respect the matching property. Without loss of generality, we can assume that $\eta(\boldsymbol{y}_{k,l}) = \eta(\boldsymbol{x}_{k',l'}) = 0$. We have

$$\psi_{\text{ker}}(\boldsymbol{y}_{k,l}) \le \psi_{\text{ker}}(\boldsymbol{x}_{k',l'}) + hL_{\text{ker}},$$

and hence

$$\begin{aligned}\ln\left[1+\exp\left(\psi_{\text{ker}}(\boldsymbol{y}_{k,l})\right)\right] &\le \ln\left[1+\exp\left(\psi_{\text{ker}}(\boldsymbol{x}_{k',l'})\right)\exp(hL_{\text{ker}})\right] \\ &\le \exp(hL_{\text{ker}})\ln\left[1+\exp\left(\psi_{\text{ker}}(\boldsymbol{x}_{k',l'})\right)\right],\end{aligned}$$

where we used the fact that

$$\ln(1+\rho x) \le \rho\ln(1+x) \tag{62}$$

whenever $\rho \ge 1$ and $x \ge 0$. In the case $\eta(\boldsymbol{y}_{k,l}) = \eta(\boldsymbol{x}_{k',l'}) = 1$, we similarly obtain

$$\ln\left[1+\exp\left(-\psi_{\text{ker}}(\boldsymbol{y}_{k,l})\right)\right] \le \exp(hL_{\text{ker}})\ln\left[1+\exp\left(-\psi_{\text{ker}}(\boldsymbol{x}_{k',l'})\right)\right],$$

and, consequently,

$$\sum_{k=0}^{M_1}\sum_{l=0}^{M_2} \ln\left[1+\exp\left(\widehat{\psi}_{\text{ker}}(\boldsymbol{y}_{k,l})\right)\right] \le \exp(hL_{\text{ker}})\left(\frac{M_1}{N_1}\right)\left(\frac{M_2}{N_2}\right)\sum_{i=0}^{N_1}\sum_{j=0}^{N_2} \ln\left[1+\exp\left(\widehat{\psi}_{\text{ker}}(\boldsymbol{x}_{i,j})\right)\right]$$

so that

$$\widetilde{\ell}_{1,\text{ker}} \le \exp(hL_{\text{ker}})\tau_1\tau_2\widetilde{\ell}_{0,\text{ker}}.$$

DOI 10.48550/arXiv.2310.18612

■

In combination with (54), Lemmas (5.1) and (5.2) allow us to move seamlessly between training and test losses. Note that the assumptions underlying these lemmas can only conceivably be satisfied by smooth ψ_{ker}, thus almost completely ruling out the suitability of ReLU activations. On the other hand, the Tanh activation function equips the kernels with Lipschitz continuity by default; the matching property is a technical assumption to rule out pathological cases where the label of a test point disagrees with those of all of its neighbouring training points.

5.3 Performance comparison

Next, we address task (b) outlined at the end of Subsection 5.1. Note that for any $0 \leq i \leq N_1$, $0 \leq j \leq N_2$, we have

$$\begin{aligned}\ln\left[1+\exp\left(\widehat{\psi}_{\text{CK}}\left(\boldsymbol{x}_{i,j}\right)\right)\right] &= \ln\left[1+\exp\left(\widehat{\psi}_{\text{CK}}\left(\boldsymbol{x}_{i,j}\right)-\widehat{\psi}_{\text{NTK}}\left(\boldsymbol{x}_{i,j}\right)\right)\exp\left(\widehat{\psi}_{\text{NTK}}\left(\boldsymbol{x}_{i,j}\right)\right)\right] \\ &\leq \max\left\{1,\exp\left(\widehat{\psi}_{\text{CK}}\left(\boldsymbol{x}_{i,j}\right)-\widehat{\psi}_{\text{NTK}}\left(\boldsymbol{x}_{i,j}\right)\right)\right\}\ln\left[1+\exp\left(\widehat{\psi}_{\text{NTK}}\left(\boldsymbol{x}_{i,j}\right)\right)\right.\end{aligned}$$

where we used (43) again. Set

$$\omega = \max_{0\leq i\leq N_1, 0\leq j\leq N_2}\exp\left(\widehat{\psi}_{\text{CK}}\left(\boldsymbol{x}_{i,j}\right)-\widehat{\psi}_{\text{NTK}}\left(\boldsymbol{x}_{i,j}\right)\right); \tag{63}$$

it follows from (52) that $\omega \geq 1$. We can therefore write

$$\widetilde{\ell}_{0,\text{CK}} \leq \omega\widetilde{\ell}_{0,\text{NTK}}. \tag{64}$$

Combining (52) and (64) with (54) and Lemmas 5.1 and 5.2 then yields the following two results.

Theorem 5.1 *Suppose that on every rectangle with corners Λ_{ij}, the functions $\widehat{\psi}_{NTK}$ and $\widehat{\psi}_{CK}$ achieve their maximum values on Λ_{ij}. Then, there exist constants $C_1, C_2 \geq 1$ such that*

$$C_1^{-1}\ell_{1,NTK} \leq \ell_{1,CK} \leq C_2\ell_{1,NTK} \tag{65}$$

Proof: We have

$$\begin{aligned}\ell_{1,\text{NTK}} &\leq (\tau_1\tau_2)\ell_{0,\text{NTK}} \quad (\because \text{Lemma 5.1}) \\ &\leq (\tau_1\tau_2)\ell_{0,\text{CK}} \quad (\because (52)) \\ &\leq (\tau_1\tau_2)\ell_{1,\text{CK}} \quad (\because (54)).\end{aligned} \tag{66}$$

Similarly,

$$\begin{aligned}\ell_{1,\text{CK}} &\leq (\tau_1\tau_2)\ell_{0,\text{CK}} \quad (\because \text{Lemma 5.1}) \\ &\leq (\tau_1\tau_2\omega)\ell_{0,\text{NTK}} \quad (\because (64)) \\ &\leq (\tau_1\tau_2\omega)\ell_{1,\text{NTK}} \quad (\because (54)).\end{aligned} \tag{67}$$

Setting $C_1 = \tau_1\tau_2$ and $C_2 = \tau_1\tau_2\omega$ and combining (66) and (67) yields (65). ■

Theorem 5.2 *Suppose that both ψ_{NTK} and ψ_{CK} are Lipschitz continuous on D and η possesses the matching property. Then, there exist constants $D_1, D_2 \geq 1$ such that*

$$D_1^{-1}\ell_{1,NTK} \leq \ell_{1,CK} \leq D_2\ell_{1,NTK} \tag{68}$$

Proof: As before, let L_{ker} be the Lipschitz constant of ψ_{ker} for both the kernels. We then have

$$\begin{aligned}\ell_{1,\text{NTK}} &\leq e^{hL_{\text{NTK}}}\ell_{0,\text{NTK}} \quad (\because \text{Lemma 5.2}) \\ &\leq e^{hL_{\text{NTK}}}\ell_{0,\text{CK}} \quad (\because (52)) \\ &\leq e^{hL_{\text{NTK}}}\ell_{1,\text{CK}} \quad (\because (54)).\end{aligned} \tag{69}$$

Similarly,

$$\begin{aligned}\ell_{1,\text{CK}} &\leq e^{hL_{\text{CK}}}\ell_{0,\text{CK}} \quad (\because \text{Lemma 5.2}) \\ &\leq e^{hL_{\text{CK}}}\ell_{0,\text{NTK}} \quad (\because (64)) \\ &\leq \omega e^{hL_{\text{CK}}}\ell_{1,\text{NTK}} \quad (\because (54)).\end{aligned} \tag{70}$$

Setting $D_1 = e^{hL_{\text{NTK}}}$ and $D_2 = \omega e^{hL_{\text{CK}}}$ and combining (69) and (70) yields (68). ■

DOI 10.48550/arXiv.2310.18612

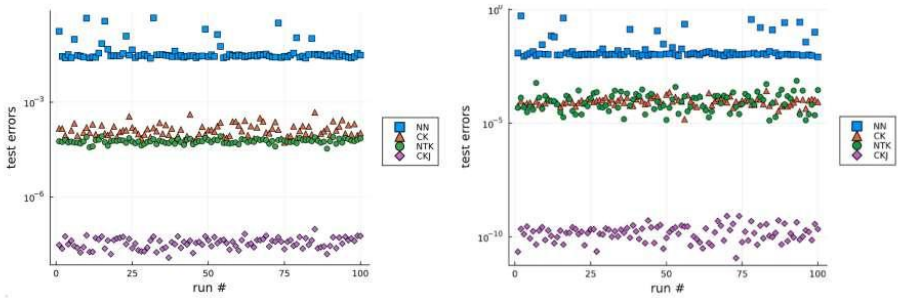
6 Numerical tests

In this section, we present numerical evidence to support the theoretical findings established in the Sections 4 and 5. In addition, we identify myriad benefits of employing CK approximations over those coming from NN and NTK, including better conditioning, inexpensive gains in accuracy, and improved robustness.

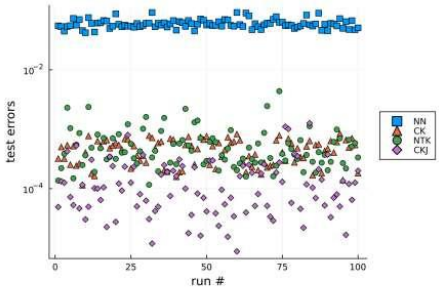
We first consider the approximation problem for smooth functions $f : [-1, 1] \to \mathbb{R}$. As examples, we use

$$f_1(x) = e^{\sin(2\pi x)}, \qquad f_2(x) = e^{3x}, \qquad f_3(x) = \cos\left(e^{3x}\right). \tag{71}$$

We note that f_3 oscillates rapidly close to $x = 1$ so it poses a particularly stiff challenge to NN and kernel approximators alike.



(a) $f_1(x) = e^{\sin(2\pi x)}$ (b) $f_2(x) = e^{3x}$



(c) $f_3(x) = \cos\left(e^{3x}\right)$

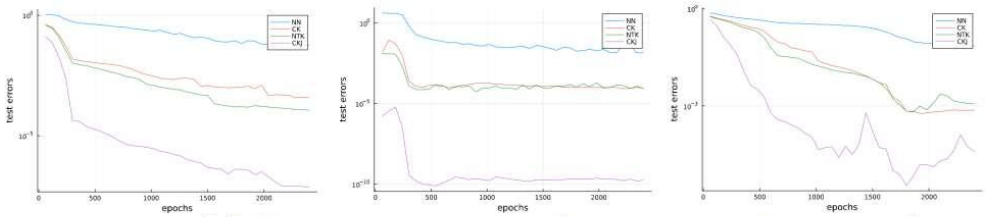
Figure 3: The results of function regression for 100 different NNs, trained for 2400 epochs, and the corresponding approximations using the NTK, CK, and CKJ extracted from the NN at the end of the training.

The sizes of the training and testing grids, as defined in Subsection 4.1, are fixed at $N = 200$ and $M = 600$ respectively, so $\tau = 3$. We train NNs of the form (7) with $L = 3$ and $d_l = d = 128$ for $1 \leq l \leq 3$ for 2400 epochs using the loss function defined in (8) and the weights given in (31). We employ the ADAM optimizer with the learning rate set at 10^{-3} and primarily use Tanh as the activation function.

After training an NN, we assemble the CK and NTK and compute the corresponding kernel approximations (14). The details of the algorithm used for extracting the NTK are given in Appendix A. The CK is easily obtained from the values of the last hidden layer and using (29). Since the feature space dimension for the CK is $(d + 1)$, assembling and using the corresponding Jacobian is also inexpensive. The results from using the Jacobian are indicated by CKJ in the following plots; a recipe for using this approximation is detailed in Appendix B.

In Figure 3, we show the test errors for the example target functions for 100 runs of the procedure described above. Owing to randomness in the initialization, the results vary over iterations; nevertheless, a clear separation is evident between the NN test errors and those of the two kernel approximations. More importantly, we note that the latter pair are fairly similar so that, while one may

DOI 10.48550/arXiv.2310.18612

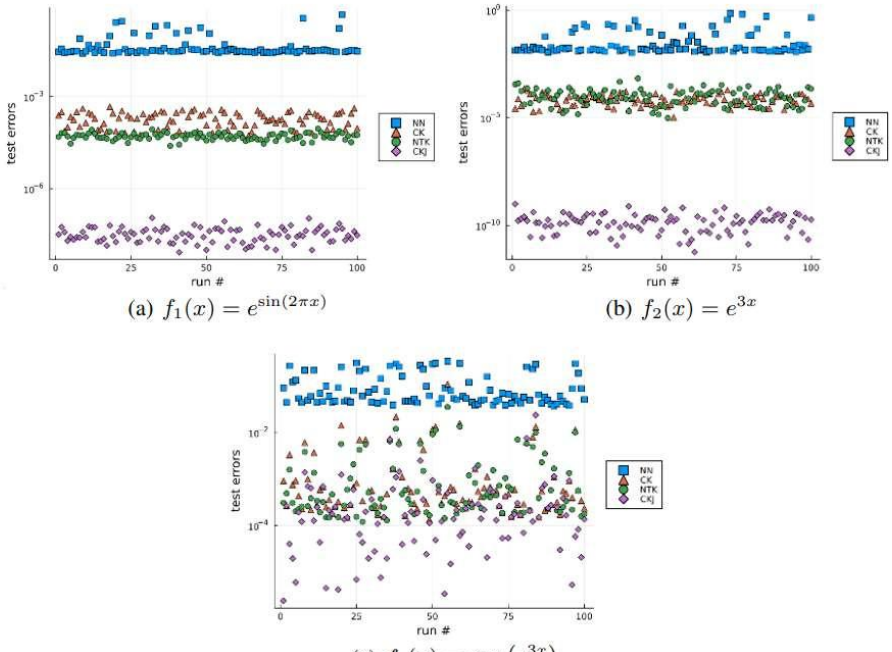


(a) $f_1(x) = e^{\sin(2\pi x)}$ (b) $f_2(x) = e^{3x}$ (c) $f_3(x) = \cos\left(e^{3x}\right)$

Figure 4: The test errors for NNs over the course of being trained to approximate the given functions, and for the corresponding NTK, CK, and CKJ approximations. The errors are averaged over ten iterations to reduce the effects of random initialization.

appear to be superior over the other for a particular example (e.g., the NTK for f_1), the difference is only marginal compared to the supremacy both enjoy over the NN.

Figure 3 also shows that the CK Jacobian yields test errors that improve on the kernel approximations by several orders of magnitude. This is a consequence of the much better conditioning possessed by the former: assembling the kernel matrix $\widehat{H}_{\text{ker}}$ squares the singular values and applying its pseudo-inverse in (14) inflates round-off errors more than the using the Jacobian does, with the result that the test errors plateau earlier. We emphasize that the use of the Jacobian is only afforded by the CK since the dimension of the feature space corresponding to the NTK is prohibitively large.



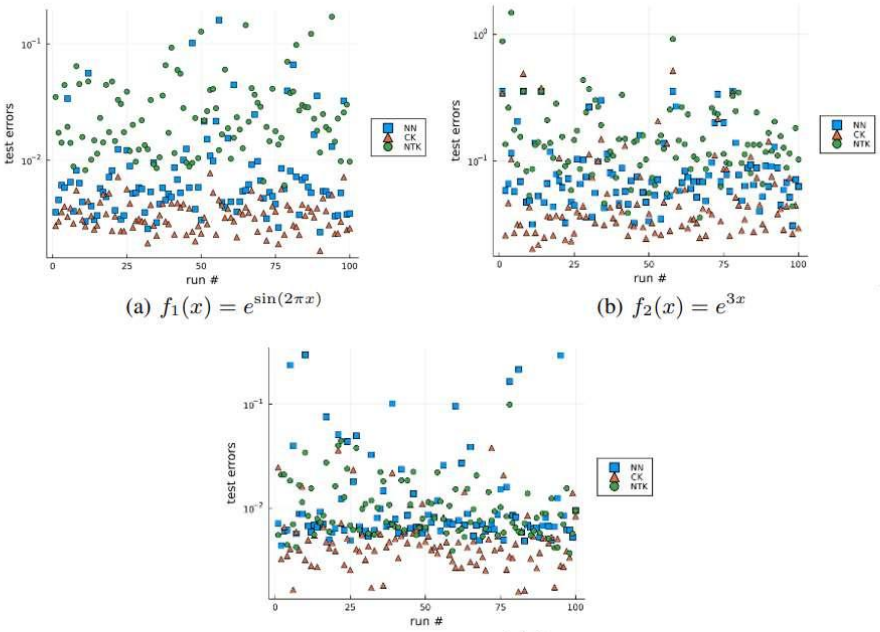
(a) $f_1(x) = e^{\sin(2\pi x)}$ (b) $f_2(x) = e^{3x}$

(c) $f_3(x) = \cos\left(e^{3x}\right)$

Figure 5: Function regression results for 100 NNs with the widths of the hidden layers set to 256, and the corresponding NTK, CK, and CKJ approximations.

These claims are further supported by the evolution of the errors over the course of the training, shown in Figure 4. The test errors are computed by averaging over ten iterations to reduce the effect of random initialization. While the NN errors can be seen to decay gradually, the kernel approximations do so more rapidly, particularly over the first few hundred epochs. The close resemblance between the two error profiles is noteworthy and supports our contention that the CK can serve as a

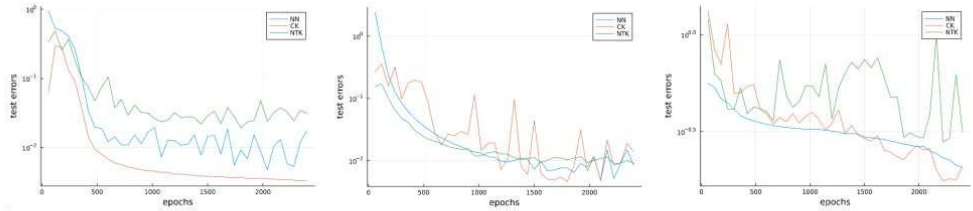
DOI 10.48550/arXiv.2310.18612



(a) $f_1(x) = e^{\sin(2\pi x)}$ (b) $f_2(x) = e^{3x}$

(c) $f_3(x) = \cos\left(e^{3x}\right)$

Figure 6: The test errors for function regression for 100 trained NNs using the ReLU activation function, and the corresponding NTK and CK approximations.



(a) $f_1(x) = e^{\sin(2\pi x)}$ (b) $f_2(x) = e^{3x}$ (c) $f_3(x) = \cos\left(e^{3x}\right)$

Figure 7: Averaged test errors for function regression over the course of training ten NNs using ReLU activations, and the corresponding NTK and CK approximations.

proxy for the NTK. Finally, we note that, as expected, the CKJ errors decay even faster and level off at a threshold several orders of magnitude lower.

In Figure 5, we show that these conclusions are also valid when we repeat the experiments with the width of the hidden layers in the NN set to $d = 256$. In particular, this assuages any concerns that the beneficial properties enjoyed by the CK and CKJ approximations may be a consequence of solving an overdetermined least-squares problem: with the larger width, this is no longer the case, and the similarity in the corresponding error plots in Figures 3 and 5 establishes that our assertions are independent of NN width.

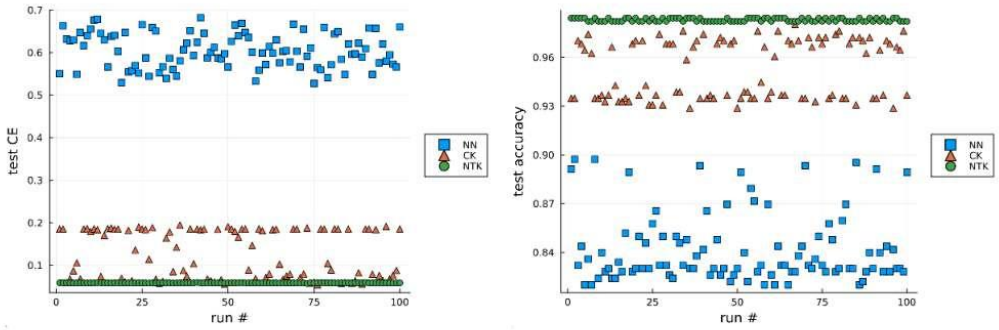
The same, however, cannot be said about the activation functions. Figure 6 shows that using ReLU activations instead of Tanh leads to the erasure of the error segregation seen in the earlier diagrams. In addition, the NTK approximators paradoxically appear to perform the worst, with the CK results only slightly bettering the NN ones. Figure 7 highlights that this is generally the case over the course of the NN training as well. These observations can be explained as a combined effect of the over-parameterization provided by the NTK and $\mathcal{S}_{\text{NTK}}$ containing discontinuous functions (due to the derivatives of the ReLU activations; see Appendix A, and (84) in particular, for more details).

DOI 10.48550/arXiv.2310.18612

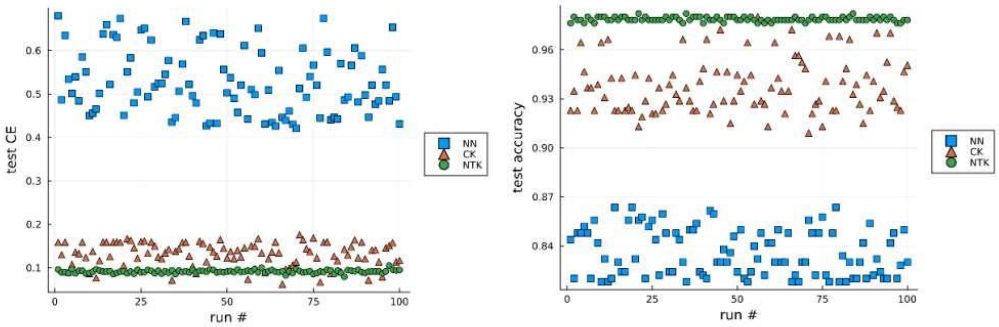
The resulting approximator manages to fit the training data very well but struggles with generalizing to the test data-points. In contrast, the $\mathcal{S}_{\mathrm{CK}}$ consists only of continuous (albeit non-differentiable) functions, with the result that, while the approximation accuracy is lower than we observed with Tanh, it does not suffer from over-fitting. This can be regarded as indicative of the dangers associated with over-parameterization, the care that must be taken with ReLU-based approximators on unseen data, and the robustness possessed by the CK.

Next, we present the results for binary logistic regression on $Z = [-1, 1]^2$. We set $N_1 = 11$, $N_2 = 7$, $M_1 = 22$ and $M_2 = 21$ (so $\tau_1 = 2$ and $\tau_2 = 3$) to specify the uniform training and testing grids (see Subsection 5.1 for the details). The labels are generated with respect to a known separating boundary $F(x_1, x_2) = 0$ for some $F : Z \to \mathbb{R}$, so the labelling function is $\eta(x_1, x_2) = (\mathrm{sign}\,(F(x_1, x_2)) + 1)/2$; we consider two examples:

$$F_1(x_1, x_2) = 4x_1^2 - 3x_1 + 5x_2 - 1, \quad F_2(x_1, x_2) = 2x_1^3 - 0.6x_1^2 - 1.94x_1 + x_2 + 0.2. \tag{72}$$



(a) $F_1(x_1, x_2) = 4x_1^2 - 3x_1 + 5x_2 - 1 = 0$



(b) $F_2(x_1, x_2) = 2x_1^3 - 0.6x_1^2 - 1.94x_1 + x_2 + 0.2 = 0$

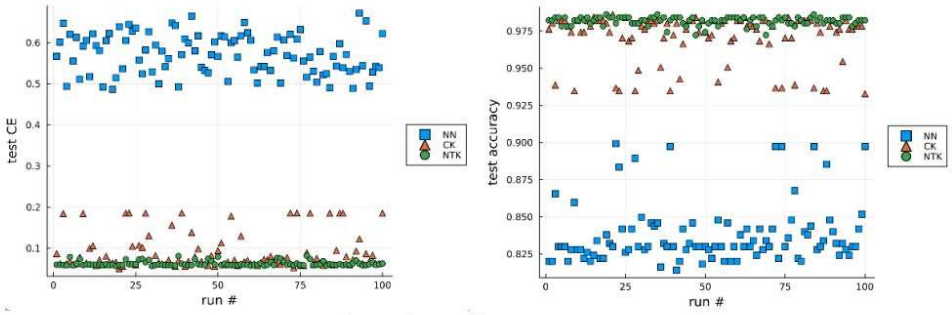
Figure 8: The test cross-entropies and accuracies for logistic regression performed with 100 NNs, and the corresponding NTK and CK results.

We train NNs of the form (7), with $L = 2$ and $d_l = d = 128$ for $l = 1, 2$, until the training accuracy (i.e., the proportion of correctly identified points) crosses 0.85, or 4000 epochs have elapsed. We again employ the ADAM optimizer, with learning rate 10^{-5}, and use Tanh as the primary activation function. The NTK and CK are extracted as detailed before and the respective training cross-entropies, defined in (15), are minimized by using Newton's method to solve $\nabla_{\boldsymbol{\alpha}} \ell_{0,\mathrm{ker}}(\boldsymbol{\alpha}) = \mathbf{0}$. We consider both cross-entropy loss values and classification accuracies as assessment metrics on the test points.

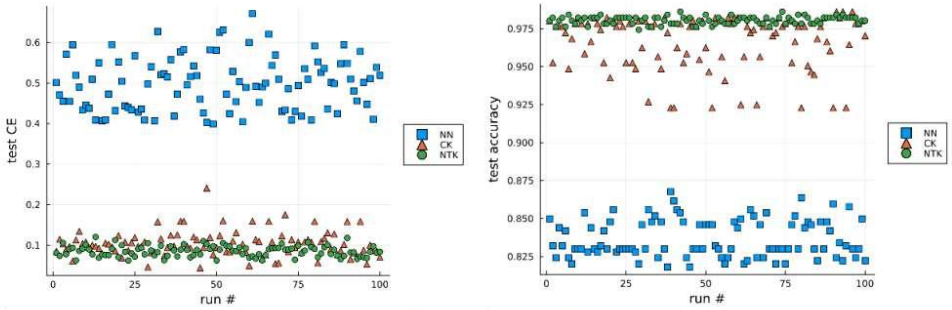
The plots in Figure 8 show the results of 100 iterations of this procedure. A clear partition is visible in the performance metrics of the NN and the two kernels on the test datasets. At the same time, while the CK is generally outperformed by the NTK, the degree of improvement is dwarfed by the separation from the NN results.

Figure 9 shows that this is still the case, even when the hidden widths of the NN are halved to 64, thus demonstrating that the CK performance is not simply a consequence of over-parameterization.

DOI 10.48550/arXiv.2310.18612



(a) $F_1(x_1, x_2) = 4x_1^2 - 3x_1 + 5x_2 - 1 = 0$



(b) $F_2(x_1, x_2) = 2x_1^3 - 0.6x_1^2 - 1.94x_1 + x_2 + 0.2 = 0$

Figure 9: The cross-entropies and accuracies for the test dataset for logistic regression performed with 100 NNs with hidden layer widths set to 64, and the corresponding NTK and CK results.

However, Figure 10 shows that the clear gap in performance vanishes when the activation function employed is ReLU. As in the case of function regression, this is explainable by over-fitting as a result of the volatile mix of over-parameterization and discontinuous feature map components.

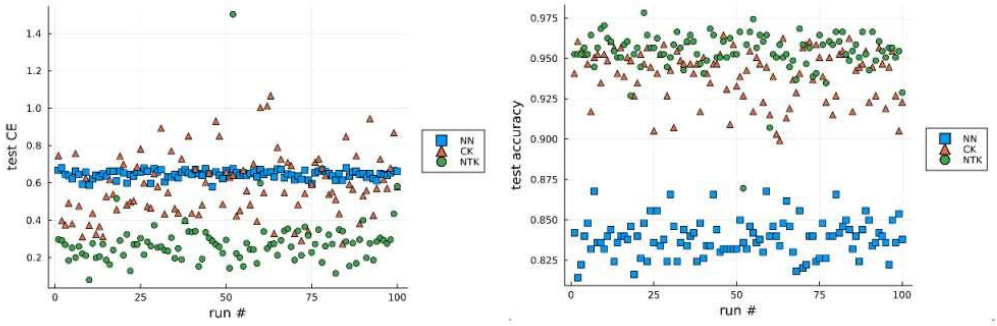
Finally, Figure 11 shows the evolution of the performance metrics over the course of the NN training. We again show the average of ten iterations to reduce the effect of random initialization. The separation alluded to earlier is evident throughout, as is the relative proximity of the loss and accuracy values of the two kernels. Moreover, we note that the gaps shrink with training, so much so that the CK metrics are barely distinguishable from the NTK ones; the NN performance improves but a sizeable gulf nevertheless persists. In particular, this highlights a crucial point: even when the two kernels have been extracted from a minimally trained NN, they yield better results than the NN does at the end of training and when its metrics have plateaued. This underscores that using the CK (i.e., explicitly retraining only the last layer of the NN) is a low-cost recipe for significantly improving the accuracy of an NN.

7 Application to Foundation Models

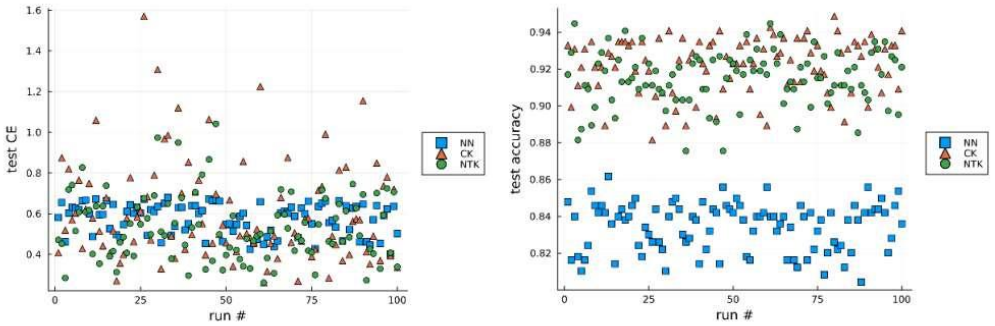
In this section, we demonstrate the effectiveness of our prescribed training strategy on a foundation model. Specifically, we make use of the pre-trained GPT-2 Radford et al. (2019) for sentiment classification on a human-annotated subset of the Sentiment140 dataset consisting of 359 tweets, all of which are labelled either positive or negative based on the expressed sentimentGo et al. (2009), rather than the entire Sentiment140 dataset due to a distribution shift between the train and test split as reported in Tsou et al. (2023) and seen in Figure 12. We evaluate two approaches:

(FT) We fine-tune by starting from pre-trained weights, attaching a randomly initialized classification head and train for ten epochs, and allowing *all* the weights to update.

DOI 10.48550/arXiv.2310.18612



(a) $F_1(x_1, x_2) = 4x_1^2 - 3x_1 + 5x_2 - 1 = 0$



(b) $F_2(x_1, x_2) = 2x_1^3 - 0.6x_1^2 - 1.94x_1 + x_2 + 0.2 = 0$

Figure 10: Test results for 100 NNs using the ReLU activation function for logistic regression, and the corresponding NTK and CK metrics.

(LP) We perform linear probing by extracting a feature vector at the final hidden layer and training a logistic regression classifier for ten epochs.

We note that these procedures can be seen as analogues of the NN and CK training respectively studied in the earlier sections. In particular, both NN and FT entail updates to all the parameters in the architecture, while the CK and LP approaches employ the feature map from the last hidden layer of a partially-trained architecture for logistic regression. Consequently, while the latter approach is significantly less expensive than the former, it can, as shown in the earlier sections, yield a marked improvement in accuracy.

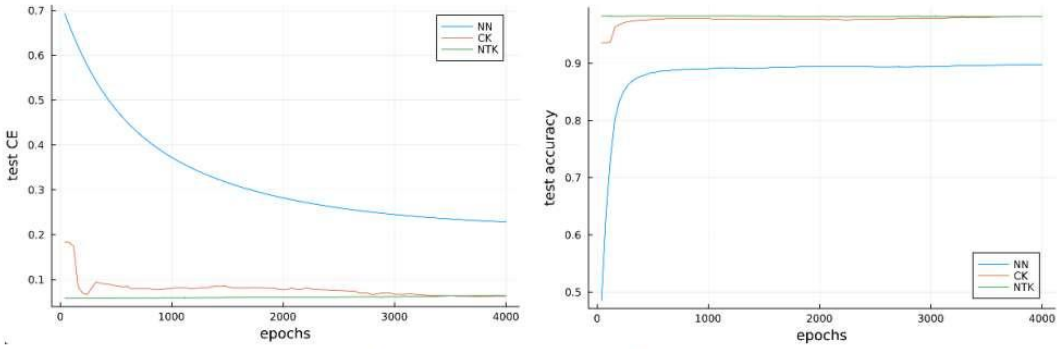
We use pre-trained weights from Hugging Face for the transformer layers Wolf et al. (2019). Furthermore, to establish a baseline, we also perform linear probing with GPT-2 initialized with random weights drawn from the truncated normal distribution $\mathcal{N}(0, 0.02)$ Radford et al. (2018),Radford et al. (2019). All experiments are performed 50 times with the training and testing data shuffled. The averaged results, along with the standard errors, are reported in Table 5 with further details about the hyperparameters provided in Appendix C, Table 6.

Model	Training Accuracy	Test Accuracy
Random feature GPT-2 + LP	$85.7 \pm 0.27\%$	$57.8 \pm 1.05\%$
Pre-trained GPT-2 + LP	$95.3 \pm 0.15\%$	$86.3 \pm 0.55\%$
Pre-trained GPT-2 + FT	$99.9 \pm 0.02\%$	$83.7 \pm 0.81\%$
Pre-trained GPT-2 + QLoRA + FT	$70.0 \pm 0.28\%$	$66.9 \pm 0.77\%$

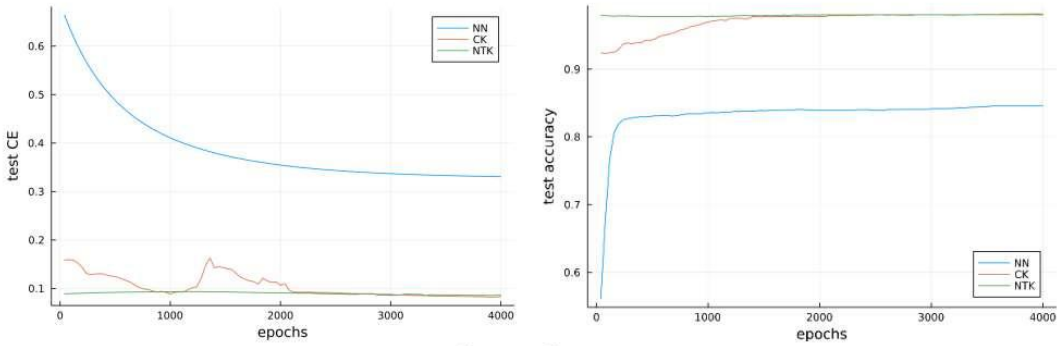
Table 5: Accuracy results for training and testing the LP and FT approaches on a human-curated subset of the Sentiment140 dataset. Mean values $\pm$ the standard error of the mean are shown.

We first note that the clear superiority (on both training and testing metrics) of linear probing following pre-training over random feature highlights the significant refinement the feature map undergoes

DOI 10.48550/arXiv.2310.18612



(a) $F_1(x_1, x_2) = 4x_1^2 - 3x_1 + 5x_2 - 1 = 0$



(b) $F_2(x_1, x_2) = 2x_1^3 - 0.6x_1^2 - 1.94x_1 + x_2 + 0.2 = 0$

Figure 11: Evolution of the test metrics, averaged over ten iterations, of NNs used to solve the logistic regression problem, and the corresponding NTK and CK results.

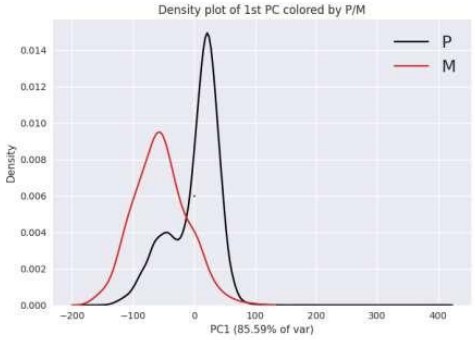


Figure 12: Kernel density estimate of the largest principal component of GPT-2's pre-trained final embedding representation of Sentiment140 P (the processed training set) and Sentiment140 M (the manually curated test set).

DOI 10.48550/arXiv.2310.18612

after being exposed to a training corpus. Next, the extremely high training accuracy from fine-tuning suggests that the architecture is overfitting and hence prone to poor generalization. This is supported by the considerably lower test accuracy. On the other hand, linear probing performs better on the test dataset, and the significantly smaller disparity between its training and testing accuracies is indicative of its robustness against overfitting. Finally, we applied QLoRA Dettmers et al. (2023) on GPT-2 to efficiently fine-tune the model on M. The trade-off for using quantized, low rank adapters is apparent compared to fine-tuning GPT-2 and the linear probing approach.

8 Discussion

Despite possessing the universal approximation property in theory Cybenko (1989); Hornik et al. (1989) and enjoying the expressiveness that accompanies increased depth Daubechies et al. (2022), in practice neural networks are stymied by optimization errors in their efforts to approximate given functions to arbitrary accuracy. Finding the optimal parameters requires the navigation of highly complex and non-convex loss landscapes and demands a large number of training epochs, thus driving up the training cost. Furthermore, the somewhat cumbersome architectures do not admit closed-form characterizations of the learned functions, making it harder to determine their generalization properties, e.g., assess them on unseen data or predict their stability under the addition of seemingly negligible noise.

Since the NTK controls the evolution during training of the learned function, it provides one way of supplying the missing characterization. In particular, an optimally trained NN, in the infinite width limit, can be shown to equal a kernel machine with a closed form known *a priori*. From (27), it follows that, for finite width networks, a kernel machine relying on the empirical NTK would necessarily outperform the NN on the training data. The numerical tests in Section 6 confirm that this is also the case on test datasets across different problems, network widths, and training stages, provided the Tanh activation function is used. However, evaluating the resulting approximations (e.g., (14) and (16)) at a new point requires computing the NTK by pairing this point with every training point, which is a highly expensive undertaking for networks used in practical applications. Worse still, the NTK performance may be inferior to that of the NN when ReLU activations are employed (e.g., Figures 6 and 10) due to discontinuities in its feature map components. Furthermore, the resulting sensitivity to small perturbations in the inputs may also leave these approximations singularly susceptible to adversarial attacks.

In this paper, we studied the properties of approximations relying on the CK and found, both theoretically and empirically, that their performance is closely tied to the NTK results, provided certain regularity conditions on the approximations are met. These are satisfied by default for the Tanh activation function, thus establishing that the gains in accuracy for the NTK over the NN are also inherited by the CK approximations. Importantly, the training cost of a CK machine is significantly lower than for an NN, yielding a closed form formula for function regression (14) and a convex optimization problem for logistic regression (15). In addition, its evaluation on new data-points is as inexpensive as evaluating the NN on the same points. As a result, the CK achieves a boost in accuracy over the NN comparable to that of the NTK while being strikingly cheaper to use than the latter and much easier to train than the former.

The CK also possesses features that, in some settings, make it even more favourable than the NTK. Due to the relatively low dimension of its feature space, the usage of its feature map form (18) for function regression is computationally viable. This approximation is much better conditioned than the corresponding kernel form, with the result that round-off errors are not magnified excessively and the approximation errors can achieve a much lower plateau, as shown in Figures 3, 4 and 5. In addition, the components of the feature map corresponding to the CK are continuous, even when using ReLU activations. As a consequence, the resulting approximations are less prone to overfitting and more robust to noise in the inputs.

We note that training a CK machine is equivalent to finding the optimal weights to attach to the neurons in the last hidden layer. Our work then suggests that a low-cost strategy for improving the accuracy of a trained NN is to simply retrain the last layer of parameters. This recipe does not alter the form of the neurons (i.e., the feature map components for the CK) but, as we have shown, can lead to significant enhancements in approximation accuracy. The regularity of the activation function is a key determinant in the success of this strategy. Observe that the choice of the activation

DOI 10.48550/arXiv.2310.18612

function does not appear to affect the NN approximation errors too much, but the kernel performance is closely tied to it (e.g., compare Figures 3 and 6, and Figures 8 and 10). This indicates that the effects of the regularity of the activation function are realized only on the kernel machines, and that the NN is largely agnostic to it.

Deploying our strategy on a foundation model for classification illustrates that freezing the feature map and training only the last layer of parameters yields performance that compares favourably with full architecture training. In addition, this approach might even lend improved robustness to the predictive capabilities in that the limited degrees of freedom resist overfitting on the training data. Our results therefore hint at the possibility that early stopping for model fitting might be beneficial in more ways than is currently surmised and that there might be subtle procedures to modulate stopping times across various parts of the network. Our recommended approach offers a computational speed up without the large trade-off in accuracy as seen in other efficient fine-tuning methods.

Finally, our results can also offer an explanation as to why feed-forward regression networks perform so well. In essence, the information captured in the last hidden layer's feature representation uniquely determines the CK. While lossy, the CK still preserves enough information from the global feature representation encoded in the NTK. And so by construction, these neuro-architectures learn and then distill the relevant information by the final hidden layer for the regression tasks. Our use of approximation theory not only offers an elegant reason as to why these networks work but also a clever way to enhance them as well.

Acknowledgements

The work of SQ is supported by the Department of Energy (DOE) Office of Advanced Scientific Computing Research (ASCR) through the Pacific Northwest National Laboratory Distinguished Computational Mathematics Fellowship (Project No. 71268). The work of AD, MV, AE, and TC were partially supported by the Mathematics for Artificial Reasoning in Science (MARS) initiative via the Laboratory Directed Research and Development (LDRD) Program at PNNL. The work of PS is partially supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research program under the Scalable, Efficient and Accelerated Causal Reasoning Operators, Graphs and Spikes for Earth and Embedded Systems (SEA-CROGS) project (Project No. 80278). Pacific Northwest National Laboratory is a multi-program national laboratory operated for the U.S. Department of Energy by Battelle Memorial Institute under Contract No. DE-AC05-76RL01830.

3.0 DOI 10.48550/arXiv.2310.13836

Foundation Model's Embedded Representations May Detect Distribution Shift

Max Vargas[1,*] **Adam Tsou**[1,2,*] **Andrew Engel**[1] **Tony Chiang**[1,3,4]
[1]Pacific Northwest National Laboratory [2]Stony Brook University
[3]University of Washington [4]University of Texas, El Paso
{max.vargas,andrew.engel,tony.chiang}@pnnl.gov
{adam.tsou}@stonybrook.edu
* Equal Contribution

Abstract

Distribution shifts between train and test datasets obscure our ability to understand the generalization capacity of neural network models. This topic is especially relevant given the success of pre-trained foundation models as starting points for transfer learning (TL) models across tasks and contexts. We present a case study for TL on a pre-trained GPT-2 model onto the Sentiment140 dataset for sentiment classification. We show that Sentiment140's test dataset M is not sampled from the same distribution as the training dataset P, and hence training on P and measuring performance on M does not actually account for the model's generalization on sentiment classification. In fact, we show that linear probes trained on the final embedding of pre-trained GPT-2 are surprisingly robust and are competitive with and may even outperform fine-tuning where all model weights update. This indicates that GPT-2's pre-existing representation of language is already able to classify sentiment.

1 Introduction

Prototypical examples in transfer learning involve training on data with potential misalignment with the testing distribution. Such instances include training with machine-generated labels Radford et al. (2022), across geographical shifts, temporal drift, linguistic drift and more Chen et al. (2020a). Due to distributional shifts, optimizations for the training distribution do not necessarily translate to optimizations on the testing distribution. If one starts with a pre-trained foundation model, further training at the level of full fine-tuning may not be necessary to achieve robust generalization Kumar et al. (2022), Zhou et al. (2023). We examine this possibility using random feature models Adlam & Pennington (2020) and linear probes Kumar et al. (2022) as baselines, alongside exploratory analysis with PCA to show that foundation models may are able to discern distribution shifts and distill common textual features.

This paper is primarily a case study of Sentiment140 Go et al. (2009) and how an *a priori* understanding of the distributional shift between its automatically processed training set (P) and manually curated testing set (M) via a selection bias allowed us to analyze the utility of full fine-tuning (FT) on a base model.

We conduct a series of experiments to investigate the internal representations of P and M for the GPT-2 architecture, showing the surprising ability of the model to separate the distribution shift, and to leverage its pre-trained features for sentiment classification. We find:

1. Dimensionality reduction on the final embedding vector of GPT-2 can differentiate the distributional shift between datasets P and M.

2. A linear probe (on pre-trained GPT-2) fit to a small sample of M provides comparable performance for sentiment classification to state-of-the-art models which have been fine-tuned using P. This indicates that a model's ability to classify sentiment may be attributed to existing features learned in pre-training and not to features learned from P.

2 Related Work

Sentiment analysis is a well studied task where the objective is to determine the emotional tone of a sequence of natural language Medhat et al. (2014),D'Andrea et al. (2015),Zhang et al. (2018), Radford et al. (2017) *et al.*

Transfer learning is most commonly applied by fine-tuning, where a randomly initialized task specific set of layers are appended to a pre-trained model. In full fine-tuning, all layers are allowed to update Yosinski et al. (2014). Many large language models are pre-trained on self-supervised tasks like masked language completion, then evaluated on performance on a suite of downstream fine-tuning tasks Devlin et al. (2019),Yang et al. (2019),Liu et al. (2019),Raffel et al. (2019),Wang et al. (2018),Go et al. (2009),Wang et al. (2019),Maas et al. (2011),Socher et al. (2013) *et al.*

Linear probing investigates the features associated with a layer by using its activation as an input feature vector to a linear classifier Belinkov (2021). A number of groups have studied linear probing: 1. for its explanatory uses Belinkov (2021), Alain & Bengio (2016), Chen et al. (2020b), 2. to provide relative baselines Kumar et al. (2022), Rosenfeld et al. (2022), and 3. for its relationship to zero-shot prompting Radford et al. (2021). In some cases, linear probing out-performs full fine-tuning Evci et al. (2022).

Random feature classification is a form of linear probing on a random initialization of weights. Jarett *et. al.* Jarrett et al. (2009) was one of the first groups to pioneer a random feature baseline for vision models. A growing number of groups have established random feature baselines for language models, achieving surprisingly robust empirical results in language and sentiment analysis Conneau et al. (2017), Conneau et al. (2018), Zhang & Bowman (2018), and addressed the challenge of establishing theoretical bounds for generalization Mei & Montanari (2019), Yehudai & Shamir (2019).

3 Methods

Data Preparation. All experiments utilize the Sentiment140 Dataset, a collection of tweets scraped from Twitter in the year 2009 Go et al. (2009). The full dataset is comprised of an automatically processed dataset P and manually curated set M. Here, P has 1,600,000 examples with labels inferred from sentimentally relevant emoticons scrubbed from the original tweets; for example ":)" indicates a positive tweet and ":(" is negative. In contrast, M consists of 359 examples with human annotated sentiment labels. Critically, P and M are sampled from different distributions: P from the population of tweets containing emoticons, and M from a population of tweets containing references to a list of subjects generated by the dataset authors. The intent of the original authors were to view P and M as testing and training sets, respectively. We choose not to adopt this perspective for concern that M and P are drawn from different distributions.

We sample either P or M for model training. In the first case, we generate a balanced subset of 20k points, a disjoint subset of 20k points for validation, and we evaluate on all of M (359 points). We call these sets P_{train}, P_{val}, and P_{test}, respectively. In the second case, we generate sets M_{train} and M_{test}; M_{train} contains 150 of each class label and M_{test} contains the 59 remaining samples for M. We emphasize that M_{train} and M_{test} contain no common data points.

Pre-trained and Random Feature Neural Network Models. Our experiments use the GPT-2 architecture Radford et al. (2019). For random feature classification, we initialize GPT-2 with random weights drawn from the truncated normal distribution $\mathcal{N}(0, 0.02)$ Radford et al. (2018),Radford et al. (2019). Otherwise, we use pre-trained weights from Hugging Face for the transformer layers Wolf et al. (2019).

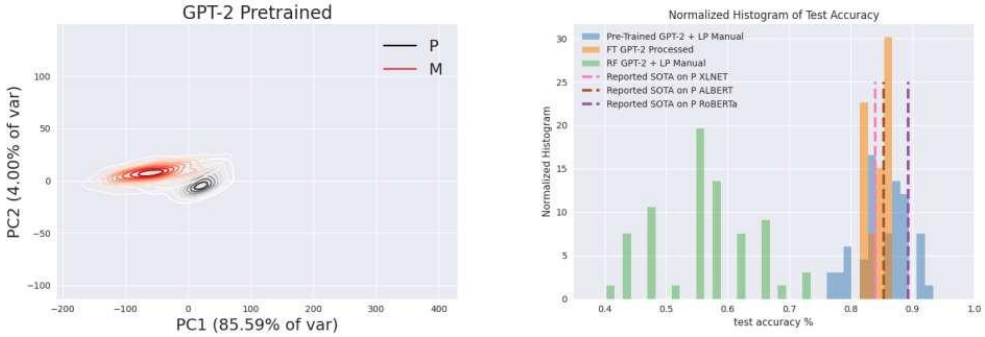
Training. We use two different training methodologies in Section 4. Further experiments in the Appendix deviate from the two methods listed here; more details can be found therein.

DOI 10.48550/arXiv.2310.13836

$\mathcal{FT}$) Starting from pre-trained weights, we attach a randomly initialized classification head and train for ten epochs, allowing *all* weights to update. We use P_{val} to choose an early stopping epoch. Evaluation is performed on M.

$\mathcal{LP}$) We perform linear probing for randomly re-initialized and pre-trained weights of GPT-2. We extract a feature vector at the final hidden layer and train a logistic regression classifier with training data M_{train} for ten epochs. We use M_{train} to determine an epoch to evaluate test accuracy on M_{test}.

4 Results



(a) KDE of PCs 1 and 2

(b) Test Accuracy

Figure 13: (a) Kernel Density Estimate of the two largest principal components of the pre-trained final embedding representation of P and M. (b) Normalized Histograms of generalization for different types of training and models, with state-of-the-art fine-tuned model performance shown as dashed lines for comparison.

LLMs can separate P and M with pre-trained weights. We examine the distributions of P and M through the lens of GPT-2. As P and M are curated by different means, we hypothesize that P and M come from inherently different distributions of natural language. To investigate this hypothesis, we construct a local-linearization of GPT-2 to quantify how the neural network represents features from these datasets. This local-linearization comes from the embedded kernel — the Gram matrix of the activations from the last hidden layer. PCA of this kernel, using the pre-trained weights of GPT-2, we can separate P and M through just the first two principal components. Figure 13a is strong evidence of our hypothesis. This analysis indicates that M is unsuitable to test generalization after training on P.

Intuitively, using P to evaluate on M confounds emoticon presence for sentiment. This is problematic if the emoticons do not perfectly match onto the notion of language-sentiment; in Table 7 we present a list of datapoints from P that are likely confounders.

Our techniques extend those by Fort *et. al* Fort et al. (2021), who perform PCA of the embedded kernel to visualize out-of-distribution data, to LLMs. Our additional experiments using linear classification to confirm the separability of P and M are available in Appendix Table 10.

Fine-tuning on datasets similar to M does not generalize to M. From the perspective of transfer learning, we can ask: can one use information learned from P to predict trends on M? The results in Table 6 indicate that this question is far more subtle than simply fine-tuning on P and then predicting on M. Starting with a pre-trained GPT-2 model and using a small amount of data from M to train a linear probe on the final hidden layer, one can separate the classes of M with notable accuracy. With this as a point of comparison, changes in the model due to fine-tuning on P are not reflected in sentiment classification on M.

Figure 13b compares the distributions of test accuracies reported in Table 6. Visually, the distribution for fine-tuning on P is aligned with that of linear probing on M. Even the accuracy for a state-of-the-art transfer model on RoBERTa (trained on the entirety of P) overlaps with the

DOI 10.48550/arXiv.2310.13836

Table 6: Comparing Fine-Tuned GPT-2 with Targeted Linear Probes

Base Model	Method	Train Acc	Test Acc
Pre-Trained GPT-2	$\mathcal{FT}$ (fine-tuning)	91.6 ± 0.93 %	84.1 ± 0.49%
Random Features GPT-2	$\mathcal{LP}$ (linear probe)	85.7 ± 0.27%	57.8 ± 1.05%
Pre-Trained GPT-2	$\mathcal{LP}$ (linear probe)	95.3 ± 0.15%	86.3 ± 0.55 %

Those equipped with the $\mathcal{FT}$ method are fine-tuned using 20k datapoints from P and tested on all 359 points of M. Those with the $\mathcal{LP}$ method have a linear probe trained using 300 points of M and tested on the remaining 59 points of M. Values are shown $\pm$ the standard error of the mean.

distribution for pre-trained GPT-2 linear probes— however, both the QQ-plot and a two-sided Kolmogorov–Smirnov test (p-value of 0.007) show that the linear probes on M differ on fine-tuning on P (Figure 21). This suggests that linear probes on M may slightly outperform fine-tuning on P (Figure 13b) in general. At best, FT, with a high cost in computation, seems to have minimal effect on the pre-trained representations, and at worst, these observations indicate that the fine-tuning procedure may actually cause the model to forget relevant semantic information it had learned during pre-training.

5 Examining P for Examples of Mislabeled Data

Table 7: Possibly Mislabeled Data in P

Label	ID	Tweet
:(	1467863072	@twitterhandle1
:(	1467871226	@twitterhandle2 Congrats!! i totally forgot to submit photos
:(	1467979094	@twitterhandle3 OHH! OMG. LMAO. I'm crying right now, LOL! KUTNERRRR was the best!
:)	1836399555	@twitterhandle4
:)	1879932456	@twitterhandle5 ????????????????
:)	1833894264	Just another SARS is coming...

We present selected examples of potentially mislabeled in dataset P. The curators of Sentiment140 scrubbed the tweets in P of emoticons. A ':(' corresponds to a 0 label, ':)' corresponds to 1 label.

In retrospect, using emoticons as a proxy to label sentiment is an imperfect method; one that the creators of Sentiment140 acknowledged as a key difficulty of creating their dataset Go et al. (2009). Table 7 showcases typical examples of such statements. Rows one and four consist of tweets only consisting of a social media handle after the relevant emoticon is removed. After pre-processing, the handle is changed to the token USERNAME, making these tweets a possible source of bias in the training data as inputs consisting of the exact same sequence of tokens would have opposite labels.

Row two includes an example of mixed sentiment. One aspect of the tweet is positive ("Congrats!!"), yet one is negative "i totally forgot to submit photos", nevertheless a human annotator may label this tweet as positive considering it starts by mentioning another user's handle and congratulating them. Row three is arguably an example of a false-negative, and we believe that this sentence expresses positive sentiment. To us, row six shows a false-positive, referencing the spread of a disease. In its original context, with a smiling emoticon, the tweet in row six would have been an example of sarcasm. Row five is also a potential false-positive as a long sequence of question marks is traditionally associated with disbelief or anger. Given that M was hand-labeled, we can assume that false-negatives and false-positives as shown in Table 7 would not be as prevalent in the data.

DOI 10.48550/arXiv.2310.13836

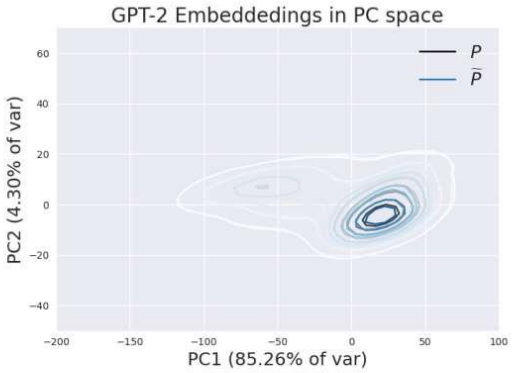


Figure 14: KDE of 20k points of P_{YN} and 29k points of P used to generate P_{YN}.

6 AI Filtration does not Remedy Distributional Shifts

Much of the work so far can be seen as an unsurprising consequence that P and M do not share the same distributional patterns under the lens of GPT-2. In fact, other models are able to identify a similar shift. Figure 15 suggests a conjectural correlation between the size of a model and its ability to differentiate distinct distributions; a small model like Bert-base cannot distinguish P and M, whereas larger models like Falcon-7b can do so.

In this section we ask whether LLMs are able to remedy some of the shift between P and M. Specifically, we use GPT-3.5 as a filter on P to create a new relabeled dataset $\widetilde{P}$. For each tweet in P, we prompt GPT-3.5 with the following in order to get a new list of positive tweets. A similar prompt provides a filtered set of negative tweets.

```
Does the following tweet, which is delimited by triple backticks,
express "positive" sentiment?

Reply with a single word and nothing else.

If the sentiment is "positive", reply "Yes".  Otherwise, reply "No".

Tweet:  ```[TWEET]```
```

The filtered dataset P_{YN} is attained by aggregating a balanced set of positive and negative tweets as understood by GPT-3.5. We do not include potentially ambiguous tweets — those tweets labeled both positive and negative and those tweets labeled neither positive nor negative. As seen in 14, there is effectively no change in the distributions between P and P_{YN}. As far as hypothesizing why this could be the case, we provide the following potential explanations:

- GPT-3.5 ingested the data comprising Sentiment140 along with its labels in its training set. In this case, it has arguably memorized P so that when we try to filter
- GPT-3.5 is randomly guessing...

Downstream, the effects of training on P_{YN} versus on P are also minimal; in fact, TABLE actually indicates that this naïve filtration using GPT-3.5 may harm consequential inference on M.

7 Filtration by PCA

Returning to figure 13a, we study whether the KDE granted by PCA can be used to filter irrelevant data from P for the purpose of studying M. Under the hypothesis that M is a contaminant of P, we perform a search to subsample P in hopes to find a collection of tweets which are in-distribution

DOI 10.48550/arXiv.2310.13836

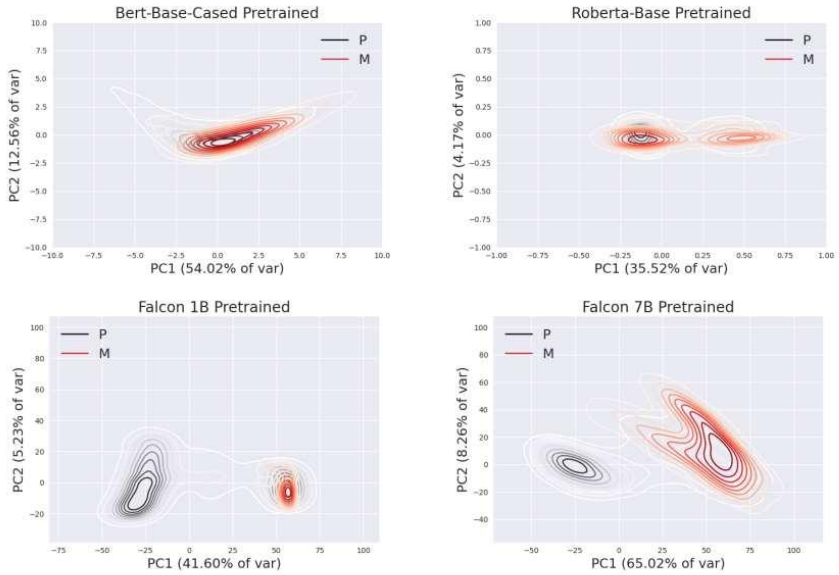


Figure 15: Kernel Density Estimates (KDEs) plotting the values of PCs 1 and 2 of the pre-trained embedded kernels for four different architectures for data corresponding to P and M.

with M. We do admit, however, that PCA does not guarantee that when two datasets follow the same distribution but rather can identify when distributions differ from each other.

8 Conclusions and Future Work

Much of the analyses in this case study can be generalized both to other models and datasets. We use four other pre-trained language models (Bert-base Devlin et al. (2019), RoBERTa-base Liu et al. (2019), Falcon 1B, and Falcon 7B Almazrouei et al. (2023)) to quantify the distributional shift between P and M. Figure 15 shows that, while Bert-base cannot differentiate the shift, both RoBERTa-base and Falcon 1B can do so with limited success. Falcon 7B is able to completely differentiate the shift like GPT-2. This would suggest that the ability to differentiate the shift correlates with the number of parameters of the model, but this inference ignores other factors such as the training corpus for each language model. Our analysis cannot disentangle these confounding effects. Beyond analyzing other sentiment datasets, one natural way to extend this work is to generate sentiment labels by one LLM (e.g. GPT-4) and determine if we find a similar shift using a different LLM (Falcon 7B).

Our work has shown the unintended consequences when models are trained and subsequently evaluated on data from different distributions. This potentially leads the community to flawed inference on the performance of these models. These consequences can be challenging to ascertain in the absence of informative baselines for comparison. While our use of a linear probe baseline was specific to a classification task, other tools may be better suited to uncover the existing capabilities of pre-trained foundation models. The standard practice of either comparing with the naïve binomial/-multinomial baseline models or benchmark models from prior studies may not be enough to fully understand the efficacy of trained models.

Acknowledgements

The work of AD, MV, AE, and TC were partially supported by the Mathematics for Artificial Reasoning in Science (MARS) initiative via the Laboratory Directed Research and Development (LDRD) Program at PNNL.

4.0 DOI 10.48550/arXiv.2309.15328

Exploring Learned Representations of Neural Networks with Principal Component Analysis

Amit Harlev
Center for Applied Mathematics
Cornell University
Ithaca, NY 14853
ah843@cornell.edu

Andrew Engel
Pacific Northwest National Laboratory
Richland, WA 99354
andrew.engel@pnnl.gov

Panos Stinis
Pacific Northwest National Laboratory
Richland, WA 99354
panagiotis.stinis@pnnl.gov

Tony Chiang
Pacific Northwest National Laboratory
University of Washington
Seattle, WA 98195
tony.chiang@pnnl.gov

Abstract

Understanding feature representation for deep neural networks (DNNs) remains an open question within the general field of explainable AI. We use principal component analysis (PCA) to study the performance of a k-nearest neighbors classifier (k-NN), nearest class-centers classifier (NCC), and support vector machines on the learned layer-wise representations of a ResNet-18 trained on CIFAR-10. We show that in certain layers, as little as 20% of the intermediate feature-space variance is necessary for high-accuracy classification and that across all layers, the first $\sim$100 PCs completely determine the performance of the k-NN and NCC classifiers. We relate our findings to neural collapse and provide partial evidence for the related phenomenon of intermediate neural collapse. Our preliminary work provides three distinct yet interpretable surrogate models for feature representation with an affine linear model the best performing. We also show that leveraging several surrogate models affords us a clever method to estimate where neural collapse may initially occur within the DNN.

1 Introduction

In the past several years, DNNs have become a common tool in many scientific fields and real-world applications. As their use becomes more widespread, it is more important now than ever to better our understanding of these models. One way this can be accomplished is by studying their learned representations. This topic has been explored by many papers in recent years, including methods such as linear probing (Alain & Bengio (2018); Cohen et al. (2018); Raghu et al. (2017); Montavon et al. (2011)), studying the dimensionality of the manifold underlying the activations (Ansuini et al. (2019); Recanatesi et al. (2019); Zhang et al. (2017)), and studying the geometry of the learned representations (Papyan et al. (2020)).

In this paper, we return to a classical tool for data analysis, *principal component analysis*, to help us better understand the learned representations present in DNNs. While several papers have used PCA to study learned representations (e.g. Montavon et al. (2011); Raghu et al. (2017)), we are the first to study in depth the performance of multiple surrogate models using varying number of PCs across an entire CNN. We train a k-nearest neighbors classifier (k-NN), a nearest class-center classifier (NCC), and a support vector machine (SVM) on each residual block's activations after projecting down to the

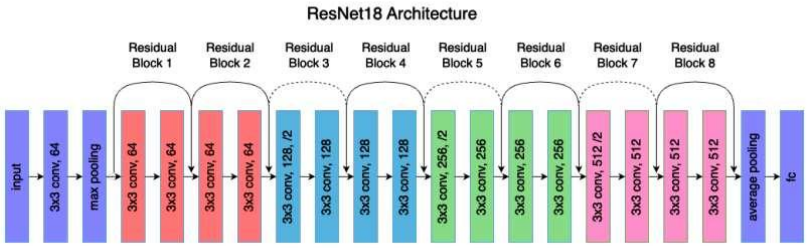


Figure 16: Diagram showing ResNet-18 architecture with residual blocks labeled.

first d principal components (PCs) and make qualitative observations based on the results. Studying a pretrained ResNet-18 on the CIFAR10 dataset, we observed that:

1. The SVM matches or outperforms the k-NN and NCC across the network.
2. The best possible performance of k-NN and NCC models on intermediate layer activations are completely determined by the first $\sim$100 PCs. In fact, the k-NN model seems to overfit as additional PCs are used.
3. The low-variance PCs of intermediate layers contain meaningful information that improves SVM performance.
4. In the latter half of the network, the PCs necessary for 90% of the classification accuracy account for only 20%-40% of the variance.

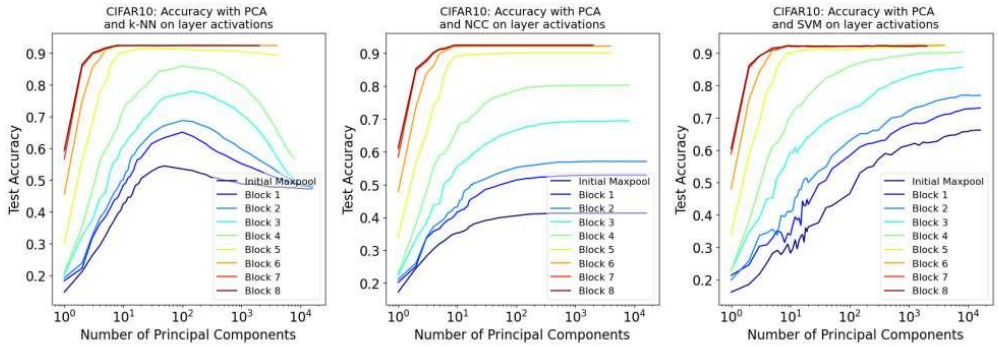
2 Related work

Probing intermediate layers. The idea behind classifier probes is that we can learn more about the behavior of intermediate layers, and thus neural networks in general, by studying the suitability of the intermediate representations for the desired task. The term "probe" was introduced by Alain & Bengio (2018), who observed that the measurements of linear probes monotonically and gradually increased on trained networks the deeper they were in the network. Cohen et al. (2018) observed that k-NN, SVM, and logistic regression probes all match the performance of a DNN in the last layer and that the k-NN predictions are almost identical to those of the DNN. Montavon et al. (2011) projected each layer's activations down to the first d (RBF) kernel principal components before training linear classifiers. They studied changes in performance as architecture, hyperparameters, and d were varied. While Montavon et al. (2011) studied early CNN architectures, we study behaviors of modern residual networks. Raghu et al. (2017) introduced SVCCA, a technique combining SVD and canonical correlation analysis, to study the relationships between representations coming from different layers, training epochs, and architectures. They show that "trained networks perform equally well with a number of directions just a fraction of the number of neurons with no additional training, provided they are carefully chosen with SVCCA."

Intrinsic dimension (ID) of neural network representations. Another approach to understanding the learned representations of DNNs has been to study their dimensionality across the network. Zhang et al. (2017) used tangent plane approximations to estimate the dimension of feature maps and observed that they declined quickly across the network. More recently, Ansuini et al. (2019) and Recanatesi et al. (2019) estimated IDs several orders of magnitude smaller than those of Zhang et al. (2017) using non-linear methods designed for curved manifolds. They also observed the layerwise ID profile to have a "hunchback" shape where the ID first increases and then drastically decreases. Ansuini et al. (2019) compared against "PC-ID", the number of PCs required to explain 90% of the variance in the activations. They observed that (1) layerwise PC-ID profiles were qualitatively the same in trained and untrained networks and (2) the PC-IDs were one to two orders of magnitude greater than IDs estimated with non-linear methods. Using this, they argued that the activations must lie on a highly curved manifold. While this may be the case, we show that PCA can in fact help find interesting structures in learned representations. Additionally, we show that while the underlying manifold may be highly curved, it exists in a low-dimensional subspace that can be found using PCA.

Neural collapse. First defined by Papyan et al. (2020), neural collapse is a phenomenon observed in the last layer activations of deep neural networks characterized by several properties, two of which

DOI 10.48550/arXiv.2309.15328



(a) (b) (c)

Figure 17: Performance of 10-NN (a), NCC (b), and SVM (c) after projecting activations from each residual block onto first d principal components.

are: **(NC1)** within-class variability collapses to zero and the activations collapse towards their class means and **(NC4)** the DNN classifies each activation using the NCC decision rule. Since then, there has been significant interest in investigating this phenomenon, including several papers exploring whether this phenomenon manifests in earlier layer's activations (Rangamani et al. (2023); Galanti et al. (2022); Ben-Shaul & Dekel (2022)). Both Galanti et al. (2022) and Ben-Shaul & Dekel (2022) study the performance of the NCC classifier across the layers of a neural network and observe an increase in performance the deeper the layer is in the network and the more training epochs used. Rangamani et al. (2023) shows that the within-class covariance decreases relative to the between-class covariance as you move deeper into a trained DNN.

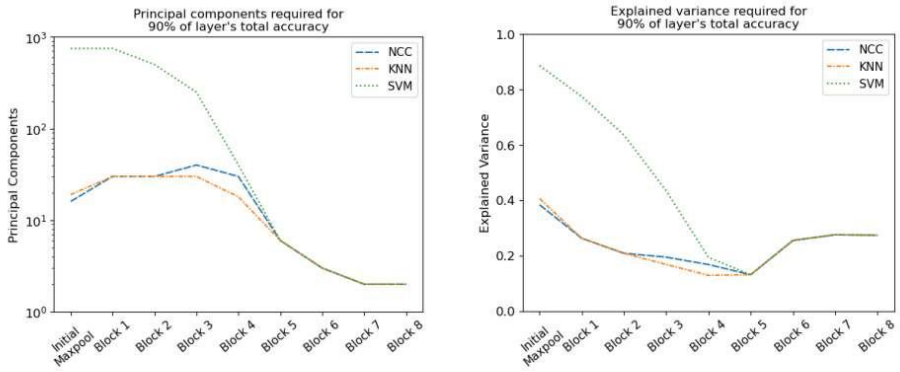
3 Experiment

We used a pre-trained (Phan (2021)) ResNet-18 (He et al. (2016)) with a test accuracy of 92.5% on the CIFAR-10 dataset (Krizhevsky et al. (2009)). For a given layer, we standardized (mean zero, std one) the activations from the training data and then used PCA to project onto the first d PCs. We trained a 10-nearest neighbors model, nearest class-center model, and soft-margin support vector machine on the resulting data and then used them to classify the test data after applying the same standardization and projection learned from the training data. This was done for each $d = 1 - 20$, 30, 40, 50, 100, 150, 200, 250, 300, 400, 500, 750, 1000, 1250, 1500, 1750, 2000 and subsequently at intervals of 1000 until reaching the size of the layer. Figure 17 shows the accuracy by number of PCs for each model. For each model and layer, we also found the minimum number of PCs needed to attain at least 90% of the best accuracy attained at that layer and by that model, as well as the variance explained by those PCs. For example, if model X's highest attained accuracy on layer Y was 96%, we found the minimum number of PCs for which model X attained $96\% * 0.9 = 86.4\%$ accuracy. This is shown in Figure 18. We considered the activations output by the initial max pooling layer and each of the eight residual blocks present in a ResNet-18— see Figure 16.

4 Results

Looking at Figure 17, we see that up until block 4, each of our three models exhibits different behaviors as we increase the number of PCs, and that from block 5 onwards, all three models exhibit qualitatively identical behavior. Up until block 4, the k-NN model's (Figure 17a) accuracy increases up to $\sim$100 PCs before decreasing significantly, a sign that it may be overfitting. On the other hand, the NCC model (Figure 17b) achieves maximum accuracy at around the same point, but then remains unchanged as more PCs are used. The SVM (Figure 17c) performs similarly to the k-NN for the first $\sim$100 PCs, but continues to improve in accuracy as the number of PCs increases. It also achieves the best performance with the original activations (i.e. before projection) across all layers. All three models see steady increases in accuracy as we move deeper into the network. On blocks 5 onwards, all three models see a sharp, almost identical spike up to the true accuracy of the DNN between one and ten PCs, followed by no change in accuracy beyond that.

DOI 10.48550/arXiv.2309.15328



(a) (b)

Figure 18: For each model: number of PCs (a) and the percentage of variance explained by those PCs (b) needed to attain 90% of maximum classification accuracy at each residual block.

In Figure 18a we see a "hunchback" profile for the NCC model (and to a lesser degree, the k-NN model) that matches the "hunchback" ID profile that Ansuini et al. (2019) observed using a non-linear dimensionality estimator. On the other hand, the SVM, the only affine-linear method we studied, exhibits a completely different profile starting very high and then monotonically decreasing. We observe that, just as in Figure 17, all three models exhibit identical profiles for blocks 5 through 8 and that, excluding block 5, they require *only* 2-3 *PCs to attain* 90% *of the accuracy of the DNN*. Figure 18b shows us that in the latter half of the network, only 20% to 40% of the variance is needed for accurate classification, and that this holds true across the entire network for the non-linear models.

5 Discussion and conclusion

While the performance of the k-NN and NCC models is determined by the first $\sim$100 PCs, the SVM's performance increases with the number of PCs up to using the whole space. When considered along with the observations of intermediate neural collapse of Rangamani et al. (2023), this could perhaps point to there being a "partially collapsed" subspace in each layer that determines the behavior of the k-NN and NCC models, while the SVM also accounts for information helpful to classification in the low variance subspaces. In particular, this means that the low-variance subspaces contain meaningful information and not just noise. Additionally, it is interesting to note that the SVM, an affine-linear model, is the most robust and best performing across all learned representations of the DNN. While all three models contribute to our intuitive understanding of how the representation is changing across the network, the SVM's accuracy suggests that applications using learned representations might benefit most from simpler models.

The behavior in blocks 5-8 can also be explained by neural collapse. That is, the network reaches a "fully collapsed state" at block 5 in which all activations are approximately equal to their class means, so all three classifiers perform equally well on very few PCs. Note that had we only trained one surrogate model, it would not be clear between which layers the network was "fully collapsing". However, with three models, Figure 17 and Figure 18 clearly show that this collapse occurs between the fourth and fifth residual blocks. Identifying this "collapsing" layer could be a useful tool for understanding mis-classified training data, as most of the information used by the DNN for classification is only present prior to that layer. The notion of intermediate neural collapse is further supported by the fact that the number of PCs needed for good classification with the SVM decreases monotonically across the network and that the variance necessary for accurate classification (by all models) decreases until block 5, which is where we see "full collapse".

Since k-NN, NCC, and PCA are all very well understood, the fact that these non-linear models display the same profile in Figure 18a as observed by Ansuini et al. (2019) provides us a more interpretable way to think about this "hunchbacked" behavior. Additionally, since the non-linear methods required only $\sim$100 PCs or less throughout the network, this implies that the curved manifold underlying the

DOI 10.48550/arXiv.2309.15328

activations most likely lives within a relatively low-dimensional subspace, which can be found using PCA.

Lastly, while it is common to select the number of PCs to keep using metrics such as accounting for 90% of variance—as seen in Ansuini et al. (2019) and Raghu et al. (2017)—Figure 18b shows that this may not be the best approach for analyzing learned representations, as the majority of the variance is not necessary for classification.

In this paper, we study learned representations of a ResNet-18 using PCA and observe multiple interesting behaviors. We hope that our work provides new intuition and inspires more experiments into the behavior and structure of learned representations, as well as demonstrates that there may still be more for us to learn about these complex models using simple techniques.

6 Acknowledgements

AH, AE, and TC were partially supported by the Mathematics for Artificial Reasoning in Science (MARS) initiative via the Laboratory Directed Research and Development (LDRD) Program at PNNL. PS was partially supported from the U.S. Department of Energy, Advanced Scientific Computing Research program, under the Scalable, Efficient and Accelerated Causal Reasoning Operators, Graphs and Spikes for Earth and Embedded Systems (SEA-CROGS) project (Project No. 80278). PNNL is a multi-program national laboratory operated for the U.S. Department of Energy (DOE) by Battelle Memorial Institute under Contract No. DE-AC05-76RL0-1830

5.0 DOI 10.48550/arXiv.2211.06506

Spectral Evolution and Invariance in Linear-width Neural Networks

Zhichao Wang
University of California San Diego
zhw036@ucsd.edu

Andrew Engel
Pacific Northwest National Laboratory
andrew.engel@pnnl.gov

Anand Sarwate
Rutgers, The State University of New Jersey
ads221@soe.rutgers.edu

Ioana Dumitriu
University of California San Diego
idumitriu@ucsd.edu

Tony Chiang
Pacific Northwest National Laboratory
University of Washington
University of Texas at El Paso
tony.chiang@pnnl.gov

Abstract

We investigate the spectral properties of linear-width feed-forward neural networks, where the sample size is asymptotically proportional to network width. Empirically, we show that the spectra of weight in this high dimensional regime are invariant when trained by gradient descent for small constant learning rates; we provide a theoretical justification for this observation and prove the invariance of the bulk spectra for both conjugate and neural tangent kernels. We demonstrate similar characteristics when training with stochastic gradient descent with small learning rates. When the learning rate is large, we exhibit the emergence of an outlier whose corresponding eigenvector is aligned with the training data structure. We also show that after adaptive gradient training, where a lower test error and feature learning emerge, both weight and kernel matrices exhibit heavy tail behavior. Simple examples are provided to explain when heavy tails can have better generalizations. We exhibit different spectral properties such as invariant bulk, spike, and heavy-tailed distribution from a two-layer neural network using different training strategies, and then correlate them to the feature learning. Analogous phenomena also appear when we train conventional neural networks with real-world data. We conclude that monitoring the evolution of the spectra during training is an essential step toward understanding the training dynamics and feature learning.

1 Introduction

Deep learning theory has made insightful connections between the behavior of neural networks (NNs) and kernel machines through asymptotic analyses of the so-called kernel regime (Neal, 1995; Williams, 1996; Lee et al., 2018; Jacot et al., 2018; Belkin et al., 2018; Arora et al., 2019; Yang, 2019). When the neural network (NN) is *infinitely wide*, the behavior of NN coincides with a kernel machine, and the training process, as well as the generalization performance of this ultra-wide NN, can be fully described. The performance of *finite-width* NNs, however, does not correspond to this theory, as NNs optimized with gradient-based methods perform better than infinitely wide networks in many circumstances (Lee et al., 2020; Hu et al., 2020; Ghorbani et al., 2020; Li et al., 2020; Daniely &

Malach, 2020; Geiger et al., 2020; Refinetti et al., 2021; Karp et al., 2021). This gap heavily relies on the task complexity, data distribution, architecture of the NN and the training strategy (Fort et al., 2020). We consider a more realistic setting, a *linear-width regime* (LWR), when the sample size n, the input feature dimension d, and the width h of the hidden layer approach infinity at comparable rates. Under the LWR, we aim to empirically study this theoretical gap in generalization and spectral properties by training various NNs with different optimization tools.

The ultra-wide NN ($h \gg n$, fixed d) stays close to the kernel machine induced by initial NN, throughout the gradient-based training processes (Zou et al., 2020; Du et al., 2018, 2019; Bartlett et al., 2021). There are two kernels commonly studied in theory: the conjugate kernel (CK) and the neural tangent kernel (NTK). CK (or the equivalent Gaussian process kernel) is the Gram matrix of the last hidden layer, which represents training only the last layer of the network (Lee et al., 2018; Louart et al., 2018; Mei & Montanari, 2022); by contrast, the NTK is the Gram matrix of the Jacobian of the NN for all trainable parameters, which governs the gradient flow of NN (Jacot et al., 2018; Du et al., 2018; Arora et al., 2019). In most theoretical results, these kernels remain fixed throughout training, which leads to a kernel gradient descent with the initial kernel (Jacot et al., 2018; Bietti & Mairal, 2019), whereas in practice the spectra of the weight matrix, CK, and NTK of the NN change while learning the features from the training data (Mahoney & Martin, 2019; Martin & Mahoney, 2021; Fan & Wang, 2020; Chen et al., 2020; Ortiz-Jiménez et al., 2021). In this paper, under the LWR, we experimentally and theoretically explore the following question:

How do the spectra of weight and kernel matrices of the NN evolve during the training process?

This question is crucial to extend our understanding beyond the kernel regime and will help us analyze the generalization of the NN in instances when it performs better than the kernel machine. For this case, the spectral properties of the trained NN could be entirely different from the initial kernel (Long, 2021; Baratin et al., 2021; Shan & Bordelon, 2021). Also, various spectral properties of weight and kernel matrices can reveal different features learned by different training procedures (Wei et al., 2022). Understanding the dynamics of the spectral properties may aid in finding better approaches to training and tuning hyper-parameters for NNs. From a theoretical perspective, random matrix theory (RMT) can be further exploited to study and elucidate the NN training under the proportional limit in high dimensions (Le Cun et al., 1991; Pennington & Bahri, 2017; Louart et al., 2018; Pennington & Worah, 2018; Hanin & Nica, 2020; Mei & Montanari, 2022).

Our main findings/contributions are as follows.

- We find a simple scenario that exhibits different spectral properties for both weight and kernel matrices through different training procedures. With the kernel regime as a benchmark, we compare how NN generalizes with different spectral evolutions of weight and kernel matrices in NN.
- The spectra of NNs trained with full batch gradient descent (GD) are globally *invariant*, indicating that the NN is still close to a kernel machine; we prove the global convergence of GD and the invariance of the limiting spectra for both weight and kernel matrices in this scenario.
- We observe a *phase transition* of the alignment and the emergence of a spike outside the bulk of the spectrum when the learning rate *exceeds* some threshold. The strong alignment of the spike with the teacher model when step sizes are large confirms that the NN is indeed learning germane features from data. This observation justifies the theoretical result of Ba et al. (2022) in an ideal two-stage training process.
- The evolution towards heavy-tailed spectra is also discovered by using adaptive methods. Our experiments rule out a *causal* relationship between the occurrence of a heavy-tailed spectrum for the weight matrices and a good generalization. This complements the work of Martin et al. (2021); Meng & Yao (2023); Yang et al. (2022) where the authors had observed a strong *correlation* between the two; while at the same time, we provide simple examples of when heavy-tailed spectra exhibit feature learning and better generalizations.

For more details on how our results fit into existing literature, please see Appendix A.

DOI 10.48550/arXiv.2211.06506

2 Notation and Preliminaries

Throughout this paper, $\|\cdot\|$ denotes the ℓ_2 norm for vectors, $\ell_2 \to \ell_2$ is the operator norm for matrices, while $\|\cdot\|_F$ is the Frobenius norm, and $\odot$ represents the Hadamard product between matrices. $o_{d,\mathbb{P}}(\cdot)$ represents little-o in probability as $d \to \infty$.

Neural Tangent Kernel Parameterization. Consider a L-layer fully connected feedforward NN at initialization without bias term: for $1 \le \ell \le L-1$,

$$\boldsymbol{h}_0 = \frac{\boldsymbol{x}}{\sqrt{d}}, \quad \boldsymbol{h}^{(\ell)} = \frac{1}{\sqrt{n_\ell}}\sigma(\boldsymbol{W}_\ell \boldsymbol{h}^{(\ell-1)}), \tag{73}$$

and $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{v}^\top \boldsymbol{h}^{(L-1)}$, where the input vector is $\boldsymbol{x} \in \mathbb{R}^d$, $\boldsymbol{W}_\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ is the weight matrix for the ℓ-th layer, and $\boldsymbol{v} := [v_1, \ldots, v_h]^\top \in \mathbb{R}^{n_{L-1}}$ is the last-layer weight. Let $n_0 = d$. Denote all trainable parameters by $\boldsymbol{\theta} := [\mathrm{vec}(\boldsymbol{W}_1), \ldots, \mathrm{vec}(\boldsymbol{W}_{L-1}), \boldsymbol{v}]^\top \in \mathbb{R}^p$ where each parameter's initial value is independently sampled from some distribution and p is the total number of parameters. Let the training dataset be $(\boldsymbol{X}, \boldsymbol{y}) := ([\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n], \boldsymbol{y}) \in \mathbb{R}^{d\times n} \times R^{1\times n}$; the output of this NN with respect to this dataset is $f_{\boldsymbol{\theta}}(\boldsymbol{X}) = [f_{\boldsymbol{\theta}}(\boldsymbol{x}_1), \ldots, f_{\boldsymbol{\theta}}(\boldsymbol{x}_n)]$. We call the above parameterization the *NTK parameterization*. The loss function for training is a mean squared error (MSE)

$$\mathcal{L}(\boldsymbol{\theta}) := \frac{1}{2n}\left\|\boldsymbol{y} - f_{\boldsymbol{\theta}}(\boldsymbol{X})\right\|^2. \tag{74}$$

We focus on the NTK parameterization and consider the kernel machine (78) induced by the initial NTK of the NN. We aim to seek the cases when the NN outperforms this kernel during the training process. For this purpose, we adopt different optimizers of training this NN to obtain different testing performances and spectral properties of trained weights and empirical kernels.

Training Processes of NNs. NNs are usually trained by gradient-based methods such as full-batch gradient descent (GD), mini-batch stochastic gradient descent (SGD), Adaptive Gradients (AdaGrad), and Adam Kingma & Ba (2014). We can represent GD by

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_t), \tag{75}$$

where η is the learning rate and $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_t)$ is the gradient of the training loss w.r.t. trainable parameters $\boldsymbol{\theta}$ at step $t \ge 0$. We will prove the global convergence of GD in some special (overparameterized) cases ensuring the convergence to a NN that interpolates the data. We will also show the hyper-parameters (e.g. learning rate η) affect the spectral properties of NNs during training.

Conjugate Kernel and Neural Tangent Kernel.[1] When $L = 2$, let $n_1 = h$ and $n_0 = d$ be the widths of the output and input layer. The CK is defined as

$$\boldsymbol{K}^{\mathrm{CK}} := \boldsymbol{X}_1^T \boldsymbol{X}_1 \in \mathbb{R}^{n\times n}, \tag{76}$$

where $\boldsymbol{X}_1 := \frac{1}{\sqrt{h}}\sigma\left(\boldsymbol{W}\boldsymbol{X}/\sqrt{d}\right)$. We can view the NN as a function of all training parameters $\boldsymbol{\theta}$ and input data $\boldsymbol{X}$. The neural tangent kernel (NTK) is related to the gradient of this neural network function with respect to $\boldsymbol{\theta}$, which is the Gram matrix of the Jacobian of the neural network function with respect to $\boldsymbol{\theta}$, $\boldsymbol{K}^{\mathrm{NTK}} := (\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{X}))^\top (\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{X}))$. Specifically, the empirical NTK of two-layer NN can be explicitly written[2] as

$$\boldsymbol{K}^{\mathrm{NTK}} = \frac{1}{d}\boldsymbol{X}^\top \boldsymbol{X} \odot \frac{1}{h}\sigma'\left(\frac{1}{\sqrt{d}}\boldsymbol{W}\boldsymbol{X}\right)^\top \mathrm{diag}(\boldsymbol{v})^2 \sigma'\left(\frac{1}{\sqrt{d}}\boldsymbol{W}\boldsymbol{X}\right) + \boldsymbol{K}^{\mathrm{CK}}. \tag{77}$$

In this paper, we are interested in comparing the spectral distributions for these three matrices (weight, CK, and NTK) at initialization and the end of training.

[1] In this work, we only consider *empirical* conjugate and neural tangent kernels of finite-width NNs.

[2] Here we train both layers, so we have two parts in the NTK expression. If we only train the first-hidden layer, we can simply remove the second CK part. In the following, we further introduce more empirical results for practical NNs in Section 5 and two-layer NNs with Gaussian dataset in Section 3. For general formula of the empirical NTK, see (Huang & Yau, 2020; Fan & Wang, 2020).

DOI 10.48550/arXiv.2211.06506

Lazy Training. Lazy training (Chizat et al., 2019) can be viewed as a linear approximation of the NN, i.e. $f_{\boldsymbol{\theta}}(\boldsymbol{x}) \approx f_{\boldsymbol{\theta}_0}(\boldsymbol{x}) + (\boldsymbol{\theta} - \boldsymbol{\theta}_0)^\top \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_0}(\boldsymbol{x})$, defined by minimum-norm interpolation $\hat{\boldsymbol{\theta}} := \arg\min \left\{ \|\boldsymbol{\theta} - \boldsymbol{\theta}_0\| : (\boldsymbol{\theta} - \boldsymbol{\theta}_0)^\top \nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_0}(\boldsymbol{X}) = \boldsymbol{y} - f_{\boldsymbol{\theta}_0}(\boldsymbol{X}) \right\}$. Then, lazy training also represents a kernel machine

$$\hat{f}(\boldsymbol{x}) = f_{\boldsymbol{\theta}_0}(\boldsymbol{x}) + (\boldsymbol{y} - f_{\boldsymbol{\theta}_0}(\boldsymbol{X}))\, \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})^{-1} \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{x}) \tag{78}$$

where $\hat{f}(\boldsymbol{x})$ is the unregularized regression prediction on test data $\boldsymbol{x} \in \mathbb{R}^d$, the kernel $\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})$ is the initial $\boldsymbol{K}^{\text{NTK}}$ on training data, and $\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{x}) = (\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_0}(\boldsymbol{X}))^\top (\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_0}(\boldsymbol{x}))$. The asymptotic performance of $\hat{f}(\boldsymbol{x})$ has been analyzed by Adlam & Pennington (2020) under the LWR. We view this regime as a *benchmark*: Chizat et al. (2019); Bartlett et al. (2021) prove that NN through gradient flow is close to lazy training if $h \gg n$; Hu et al. (2020) shows NN can go beyond lazy training under a non-proportional regime.

3 Case Study for Linear-width NNs

In this section, we investigate a two-layer NN with synthetic data. This setting is promising for future theoretical studies by virtue of RMT. We will showcase the evolution of its spectral properties over training. A two-layer NN in (73) is defined by

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) := \frac{1}{\sqrt{h}} \sum_{i=1}^{h} v_i \sigma(\boldsymbol{w}_i^\top \boldsymbol{x} / \sqrt{d}). \tag{79}$$

At initialization, we assume that the first hidden-layer $\boldsymbol{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_h]^\top \in \mathbb{R}^{h \times d}$ is composed of independent standard normal random vectors.

Assumption 3.1 (Linear-width regime (LWR))**.** Assume that $\frac{n}{d} \to \gamma_1$ and $\frac{h}{d} \to \gamma_2$ as $n \to \infty$ where the aspect ratios $\gamma_1, \gamma_2 \in (0, \infty)$ are two fixed constants.

LWR stands as a pivotal setting grounded in high-dimensional statistics Adlam & Pennington (2020); Mei & Montanari (2022). It offers valuable insights especially when addressing real-world datasets. This is in contrast to the infinite-width regime, in which we are already in the asymptotic limit for width at first. Hence, LWR is a better approximation of real-world datasets and practical neural networks compared with the infinite-width regime.

Assumption 3.2 (Activation function)**.** Suppose that the activation function $\sigma(x)$ is nonlinear and λ_σ-Lipschitz with $|\sigma'(x)|, |\sigma''(x)| \le \lambda_\sigma$ for all $x \in \mathbb{R}$. Moreover, $\mathbb{E}[\sigma(z)] = 0$ for $z \sim \mathcal{N}(0, 1)$.

Though the LWR is somewhat impractical, it is still more aligned with deployed models than the infinite-width regime ($h \gg n$, fixed d). As a kernel machine, the infinite-width NN has been studied extensively (Jacot et al., 2018; Du et al., 2018, 2019; Yang, 2019). This infinite-width limit is special, however, as NNs may generally evolve beyond the kernel regime and achieve superior performance (Fort et al., 2020; Long, 2021; Baratin et al., 2021; Shan & Bordelon, 2021).

	Optimization	Learning rate η	R^2 score	Test error	Spectra
Case 1	GD	5.0	0.63582	0.36381	Invariant Bulk
Case 2	SGD	0.1	0.60605	0.36879	Invariant Bulk
Case 3	SGD	22.0	0.76081	0.23791	Bulk+spike
Case 4	Adam	0.092	**0.78829**	**0.21071**	Heavy tail
	Lazy regime		0.68092	0.3185	

Table 8: Four models with the same architecture ($n = 2000$, $h = 1500$, $d = 1000$, and σ is normalized $\tanh$), but different choices of initial learning rates and optimizers listed in Table 8. The training label noise $\sigma_\varepsilon = 0.3$ and the teacher model is defined by (81) with σ^* a normalized *softplus* and $\tau = 0.2$. We observe that simply choosing an optimizer and learning rate can affect the shapes of the final spectra and the performance of the NN, as measured by R^2 scores and test errors.

Assumption 3.3 (Synthetic dataset and teacher model)**.** Training data is $\boldsymbol{X} := [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$, where $\boldsymbol{x}_i \overset{i.i.d.}{\sim} \mathcal{N}(\boldsymbol{0}, \mathbf{I}_d)$. The training labels $\boldsymbol{y} = [y_1, \ldots, y_n]$ are defined by $y_i = f^*(\boldsymbol{x}_i) +$

DOI 10.48550/arXiv.2211.06506

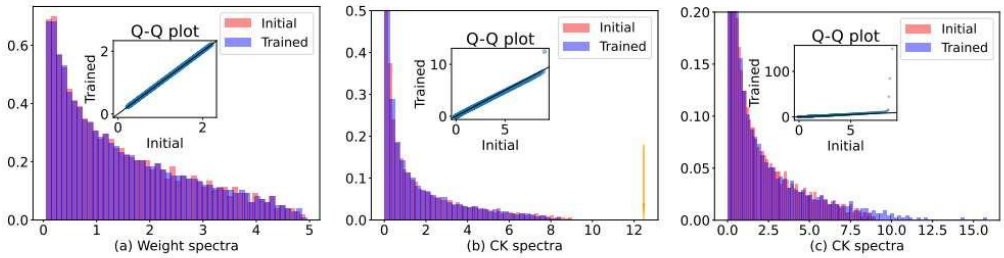


Figure 19: Different spectral behaviors in Table 8: (a) The initial and trained spectra of $\boldsymbol{W}$ in Case 1. The spectrum is invariant based on the Q-Q subplot. (b) The initial and trained spectra of $\boldsymbol{K}^{\mathrm{CK}}$ in Case 3. There is an outlier (orange arrow) in the spectrum after training. (c) The initial and trained spectra of $\boldsymbol{K}^{\mathrm{CK}}$ in Case 4. We refer to Appendix B.2 for other spectra of weight, CK, and NTK matrices in Case 1-4, where analogous phenomena hold for other matrices.

$\varepsilon_i, \quad \text{for } i \in [n]$, where $f^* : \mathbb{R}^d \to \mathbb{R}$ is the teacher model, and ε_i is centered sub-Gaussian noise with variance σ_ε^2.

One of the simplest nonlinear teacher models we can generate is the single-index model, namely $f^*(\boldsymbol{x}) = \sigma^*(\boldsymbol{x}^\top \boldsymbol{\beta})$ for a fixed vector $\boldsymbol{\beta}$ with $\|\boldsymbol{\beta}\| = 1$ and nonlinear function σ^*; the hidden feature is simply $\boldsymbol{\beta} \in \mathbb{R}^d$. In general, we can consider a multiple-index model

$$f^*(\boldsymbol{x}) = \frac{1}{k} \sum_{i=1}^{k} \sigma^*(\boldsymbol{x}^\top \boldsymbol{\beta}_i) \tag{80}$$

where $\boldsymbol{\beta}_i$ are some orthogonal unit vectors. We will specifically consider a mixture of single-index and quadratic models as our teacher model in this section:

$$f^*(\boldsymbol{x}) = \sigma^*(\boldsymbol{x}^\top \boldsymbol{\beta}) + \frac{\tau}{d} \|\boldsymbol{x}\|^2, \tag{81}$$

for some nonlinear target σ^*, signal $\boldsymbol{\beta}$ and constant τ[3]. Following the above assumptions and constructions, we show different spectral properties (Figure 19) for this two-layer NN using different training procedures (Table 8). Figure 19 exhibits three types of spectra after training: unchanged bulk distribution, bulk with one spike, and heavy tail in spectra. Putting things together, we can see close relationships between the spectra and the generalization of the NN. These different spectral properties actually reveal disparate features learned via different training strategies.

The advantage of this toy model is that we can easily extract the spectral behaviors over training and then compare them with the kernel machine. We use lazy training defined in (78) as our *benchmark* to assist us in determining whether a NN outperforms the associated kernel machine. Table 8 compares the test errors and R^2 scores for different optimization cases and the lazy training. By tuning the hyper-parameters, we can find specific situations where NN outperforms the lazy training (see also Figure 28(c) in Appendix B.2).

From Figures 19(a), 30 and 31 in Appendix B.2, one can observe the spectral distributions of the weight, CK and NTK matrices remain invariant and static during training in Cases 1&2, which indicates both cases still belong to the lazy regime. This spectral invariance impedes further feature learning during the training process. The emergence of the outlier in Figures 19(b) and 32 of Appendix B.2, however, shows the improvement over lazy training and potential feature learning via the training process, where the spectra possibly inherit the structures in teacher models (see Section 4.2). Comparing with Case 2, Case 3 of Table 8 suggests the importance of the large learning rate regime for training NNs (Li et al., 2019; Nakkiran, 2020; Long, 2021; Beugnot et al., 2022; Andriushchenko et al., 2023). As a remark, our spectral results of Case 3 are consistent with the observations in Thamm et al. (2022) through RMT hypothesis testing, where the majority of trained weight matrices remain random, and the learned feature may be contained in the largest singular value (outlier) and associated vector only. From Figures 19(c) and 34 in Appendix B.2, Case 4 further

[3]Here, the norm term of $\boldsymbol{x}$ in (81) is designed to make the teacher model more complicated to be learned. All our empirical results still hold when $\tau = 0$.

DOI 10.48550/arXiv.2211.06506

exhibits more spikes and heavy tails in the trained spectra, which thoroughly goes beyond the realm of initial kernel machine. Notably, this phenomenon is not unique to Adam since heavy tails also occur with AdaGrad in Figure 40 in Appendix B.6. Although all of these cases have the same identical initialization, different methods of optimization eventually lead to various training trajectories and evolutions of the spectra of the weight and kernel matrices. To acquire feature learning, Cases 3&4 cause weights to deviate far from initialization. In the following Section 4, we prove the invariance of the bulk distributions and provide more refined analyses of spikes and heavy tails in terms of feature learning.

4 Different Spectral Behaviors in NNs

We now further explore the spectral behaviors in different cases of Table 8 by clarifying how the spectra evolve through different training processes and how this evolution may affect the NN. Following Figure 19, we study the training processes case-by-case: invariant bulk, spikes outside the bulk, and heavy-tailed distribution. Additional experiments are exhibited in Appendix B.

4.1 Invariant Bulk Distributions

In Figure 19(a) (also Figures 30 and 31 in Appendix B.2), we observe the bulk distributions of weight and kernel matrices in Cases 1&2 remain globally unchanged (invariant) over the training process. under the LWR, this is also empirically verified by Figures 28(a)&(b) in Appendix B.2. In this section, by investigating the global convergence of GD, we prove this invariant-bulk phenomenon under certain assumptions.

For simplicity, we focus on analyzing the training process of the first-hidden layer with the second layer $\boldsymbol{v}$ fixed. Denote $f_{\boldsymbol{\theta}}(\boldsymbol{X})$ by $f_{\boldsymbol{W}}(\boldsymbol{X})$ in this case. At any time $t \in \mathbb{N}$, consider the gradient steps:

$$\boldsymbol{W}_{t+1} = \boldsymbol{W}_t - \eta \nabla_{\boldsymbol{W}} \mathcal{L}(\boldsymbol{W}_t). \tag{82}$$

Denote the CK and NTK at gradient step $t \in \mathbb{N}$ by $\boldsymbol{K}_t^{\text{CK}} := \frac{1}{h}\sigma(\boldsymbol{W}_t\boldsymbol{X})^\top \sigma(\boldsymbol{W}_t\boldsymbol{X})$, and $\boldsymbol{K}_t^{\text{NTK}} := \frac{1}{d}\boldsymbol{X}^\top\boldsymbol{X} \odot \frac{1}{h}\sigma'\left(\frac{1}{\sqrt{d}}\boldsymbol{W}_t\boldsymbol{X}\right)^\top \operatorname{diag}(\boldsymbol{v}_t)^2 \sigma'\left(\frac{1}{\sqrt{d}}\boldsymbol{W}_t\boldsymbol{X}\right)$ respectively. First, we present an elaborate description of the changes in the weight, CK, and NTK at the *early phase* of the training (after any finite t steps) as follows.

Lemma 4.1 (Early phase)**.** *Under Assumptions 3.1, 3.2 and 3.3, we further assume that $\|\boldsymbol{v}\|_\infty \le 1$ and f^* is a λ_σ-Lipschitz function. Given any fixed $t \in \mathbb{N}$ and learning rate $\eta = \Theta(1)$, after t gradient steps, the changes $\frac{1}{\sqrt{d}}\|\boldsymbol{W}_t - \boldsymbol{W}_0\|_F$, $\left\|\boldsymbol{K}_t^{CK} - \boldsymbol{K}_0^{CK}\right\|_F$, and $\left\|\boldsymbol{K}_t^{NTK} - \boldsymbol{K}_0^{NTK}\right\|$ are all less than $\frac{C}{n}$, with probability at least $1 - 4n\exp(-cn)$, for some positive constants $c, C > 0$ which only depend on step t and parameters $\eta, \gamma_1, \gamma_2, \lambda_\sigma, \sigma_\varepsilon$.*

Lemma 4.1 shows $\frac{1}{\sqrt{d}}\|\boldsymbol{W}_t - \boldsymbol{W}_0\|$, $\left\|\boldsymbol{K}_t^{\text{CK}} - \boldsymbol{K}_0^{\text{CK}}\right\|$, and $\left\|\boldsymbol{K}_t^{\text{NTK}} - \boldsymbol{K}_0^{\text{NTK}}\right\|$ are asymptotically vanishing for any fixed time t. Therefore, all the eigenvalues/eigenvectors are asymptotically unchanged at the early phase of the training (see Corollary C.3 in Appendix C). Now we aim to analyze the spectra at the end of the training process (82). In this case, although we are unable to show the invariance for each eigenvalue, we can verify the invariance of the limiting bulk distributions for $\boldsymbol{K}_t^{\text{CK}}$ and $\boldsymbol{K}_t^{\text{CK}}$ for all t.

By (Wang & Zhu, 2021, Theorem 2.9), the smallest eigenvalue of $\boldsymbol{K}_0^{\text{NTK}}$ has an asymptotic lower bound:

$$\lambda_{\min}(\boldsymbol{K}_0^{\text{NTK}}) \ge \left(a_\sigma - \sum_{k=0}^{2}\eta_k^2\right)(1 - o_{d,\mathbb{P}}(1)), \tag{83}$$

where $a_\sigma := \mathbb{E}[\sigma'(\xi)^2]$ and η_k is the k-th Hermite coefficient of σ'. Hence, we can claim there exists some constant $\alpha > 0$ only dependent on σ such that $\lambda_{\min}(\boldsymbol{K}_0^{\text{NTK}}) \ge 4\alpha^2$ with high probability. Note that α is not vanishing since σ is nonlinear. With this lower bound, we obtain the following global convergence for (82) and norm control of $\boldsymbol{W}_t$ as $n/d \to \gamma_1$ and $h/d \to \gamma_2$.

Theorem 4.2 (Global convergence)**.** *Under the same assumptions of Lemma 4.1, we further assume v_i's are independent and centered random variables in the second layer. For any*

DOI 10.48550/arXiv.2211.06506

$\eta < \min\{\frac{\alpha^2 n}{2}, \frac{n}{4\lambda_\sigma^2(1+\sqrt{\gamma_1})^2}\}$ *and all* $t \in \mathbb{N}$, *there exists some* $\gamma^* > 0$ *such that, when* $\gamma_2 \geq \gamma^*$, *the gradient steps* (82) *will satisfy*

$$\ell(\boldsymbol{W}_t) \leq \left(1 - \frac{\eta\alpha^2}{2n}\right)^t \ell(\boldsymbol{W}_0), \tag{84}$$

$$\frac{1}{4}\alpha \left\|\boldsymbol{W}_0 - \boldsymbol{W}_t\right\|_F + \ell(\boldsymbol{W}_t) \leq \ell(\boldsymbol{W}_0), \tag{85}$$

$$\sum_{t=0}^{\infty} \left\|\boldsymbol{W}_{t+1} - \boldsymbol{W}_t\right\|_F \leq \frac{4\ell(\boldsymbol{W}_0)}{\alpha}, \tag{86}$$

with high probability, as $n/d \to \gamma_1$ *and* $h/d \to \gamma_2$. *Here, training loss* $\ell(\boldsymbol{W}) := \|\boldsymbol{y} - f_{\boldsymbol{W}}(\boldsymbol{X})\|$.

We apply the techniques and results by Oymak et al. (2019); Oymak & Soltanolkotabi (2019) to obtain Theorem 4.2. Notice that, unlike Lemma 4.1, the largest learning rate we can choose is of order $\Theta(n)$. As a byproduct, the Frobenius norm in (85) implies the following corollary for the invariance of limiting *bulk* distribution.

Corollary 4.3. *Under the same assumptions of Theorem 4.2, for all* $t \in \mathbb{N}$, *with high probability, there exists some constant* $R > 0$ *such that the changes* $\frac{1}{\sqrt{d}}\left\|\boldsymbol{W}_t - \boldsymbol{W}_0\right\|_F$, $\left\|\boldsymbol{K}_t^{CK} - \boldsymbol{K}_0^{CK}\right\|_F$, *and* $\left\|\boldsymbol{K}_t^{NTK} - \boldsymbol{K}_0^{NTK}\right\|_F$ *are all less than* R *with high probability. This implies the limiting empirical spectra of* $\frac{1}{h}\boldsymbol{W}_t^\top \boldsymbol{W}_t$, $\boldsymbol{K}_t^{CK}$ *and* $\boldsymbol{K}_t^{NTK}$ *are the same as the limiting spectra of* $\frac{1}{h}\boldsymbol{W}_0^\top \boldsymbol{W}_0$, $\boldsymbol{K}_0^{CK}$ *and* $\boldsymbol{K}_0^{NTK}$ *respectively, almost surely as* $n/d \to \gamma_1$ *and* $h/d \to \gamma_2$.

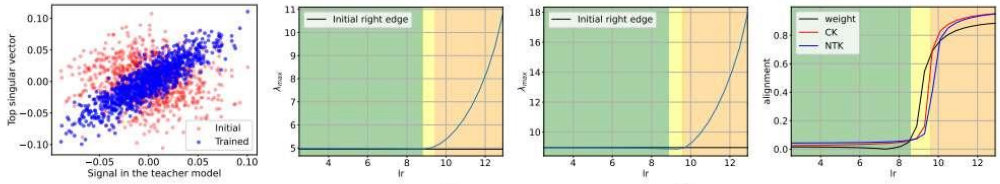
Corollary 4.3 is empirically validated by Figure 47 in Appendix C. In addition, based on Figure 49 in Appendix C, one can further extend Corollary 4.3 to the SGD training process. The total path is $O(\sqrt{h})$ in (85) and (86), which is negligible compared with the Frobenius norm of initial weight matrix (which is of order $\Theta(h)$). Thus, gradient descent iterates (82) remain close to initialization and small perturbation of NTK ensures the smallest eigenvalue (83) of NTK is always lower bounded away from zero. Theorem 4.2, however, does not require that the NTK stays unchanged all the time. Moreover, Corollary 4.3 only shows the invariance of the bulk distribution, while the emergence of outliers cannot be excluded from this result. Though we have global convergence in general, we may still move out of the kernel regime. Global convergence cannot explain when a NN in LWR outperforms the kernel regime (Figure 28(c)). Notice that (Bartlett et al., 2021, Theorem 5.4) is not directly applicable to show that a NN is still close to lazy training under the LWR. It requires deeper analysis to claim whether a NN still belongs to the kernel regime or already goes beyond in our case. As we will show in Section 4.2, this also relies on the magnitude of the learning rate for GD/SGD.

4.2 Alignments for Spiked Models

The outliers appear in the spectra of the trained weight, CK, and NTK matrices (Figure 32 in Appendix B.2) when NNs are optimized with large learning rates. The outlier is especially clear for the NTK matrix (Figure 32(b)). Heuristically, this indicates that the NN is learning the feature from the teacher model f^*. In Figure 20(a), Figures 33 and 45 in Appendix B, we empirically exhibit these phenomena for $\boldsymbol{W}$, $\boldsymbol{K}^{\text{CK}}$ and $\boldsymbol{K}^{\text{NTK}}$ respectively through different training processes.

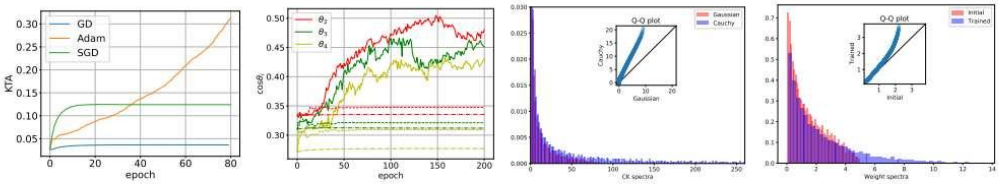
Spikes of Weight Matrices. The differences between Cases 2&3 empirically validate the benefits of training with large learning rates (Li et al., 2019; Nakkiran, 2020; Long, 2021; Beugnot et al., 2022; Andriushchenko et al., 2023). Inspired by Ba et al. (2022), we consider the alignment between the leading right singular vector $\boldsymbol{u}_1$ of $\boldsymbol{W}_t$ and the signal $\boldsymbol{\beta}$ in the teacher model defined by (81). For Case 3, a notable alignment appearing in Figure 20(a) after training suggests that $\boldsymbol{W}_t$ is capturing the feature $\boldsymbol{\beta}$ during training. Although this does not ensure NN will entirely beat the optimal kernel lower bound, this alignment reveals a non-negligible feature selection (Baratin et al., 2021) via large-stepsize training. This dynamical alignment along the task-relevant direction may further interpret the generalization of the NN. We also observe similar phenomena for the adaptive optimization in Figure 45 in Appendix B.7.

Transitions of the Spike as a Function of Learning Rate. From Case 2 to Case 3, we observe the emergence of outliers in the trained spectra when increasing the learning rate η. This indicates



(a) $\boldsymbol{\beta}$ vs. $\boldsymbol{u}_1$ of weight. (b) $\lambda_{\max}(\boldsymbol{W}_t\boldsymbol{W}_t^\top/d)$ (c) $\lambda_{\max}(\boldsymbol{K}_t^{\mathrm{CK}})$ (d) Alignments

Figure 20: (a) Alignment between teacher feature $\boldsymbol{\beta}$ and first PC of the trained/initial weights in Case 3 of Table 8. (b)-(d) Transitions of $\lambda_{\max}(\boldsymbol{W}_t\boldsymbol{W}_t^\top/d)$, $\lambda_{\max}(\boldsymbol{K}_t^{\mathrm{CK}})$ and alignments ($|\boldsymbol{\beta}^\top\boldsymbol{u}_1|/\|\boldsymbol{\beta}\|$ and $|\boldsymbol{y}^\top\boldsymbol{v}_1|/\|\boldsymbol{y}\|$ where $\boldsymbol{u}_1$ and $\boldsymbol{v}_1$ are the first singular vectors of $\boldsymbol{W}_t$ and either $\boldsymbol{K}_t^{\mathrm{CK}}$ or $\boldsymbol{K}_t^{\mathrm{NTK}}$, respectively) when increasing the learning rate η while training the NN with SGD. In the green region, the largest eigenvalues are attached to the bulk (black horizontal lines) and the alignments are weak; in the orange one, outliers become apparent and the alignments become stronger. For different η, we train the same NN with the same dataset until the training loss is less than 10^{-5}. Here, "lr" in the x-axis represents varying learning rates.



(a) Evolution of KTA. (b) Alignment evolution. (c) Heavy-tailed initial CK. (d) Bad example.

Figure 21: (a) Evolution of KTA of CK defined by (87) with respect to training labels for Cases 1, 3&4 in Table 8. We normalize the epoch scales (x-axis) for better observations. Heavy-tailed phenomena: (b) Evolutions of PC angles θ_i between feature subspace U of (80) and top 100 eigenspace of $\boldsymbol{W}_t^\top\boldsymbol{W}_t$ during training with Adam (solid line), SGD (dashed line) and GD (dash-dot). For the first PC θ_1, see Figure 35 in Appendix B.4. (c) The CK spectra at two initializations for $\boldsymbol{W}$: standard Gaussian and Cauchy distributions. (d) Weight spectra at initial and after SGD training. After training the weight reveals a heavy tail, but generalizes not as well as former examples (test loss 1.47504; R^2 score -0.48).

a transition of the emergence of the spike outside the bulk distribution. Figure 20, analogously to the well-known BBP transition by Baik, Ben Arous, and Péché in Baik et al. (2005) from the RMT community, shows there is a threshold (yellow region) for learning rate: the outliers only appear when η exceeds this threshold. We fix the same NN and dataset for all trials of training. The flat black lines in Figures 20(b) and (c) are the right edges of the limiting spectra at initialization. Figure 20(d) records the angles between $\boldsymbol{\beta}$ and the leading eigenvector of $\boldsymbol{W}_t^\top\boldsymbol{W}_t/d$, and $\boldsymbol{y}$ and the leading eigenvectors of $\boldsymbol{K}_t^{\mathrm{CK}}$ and $\boldsymbol{K}_t^{\mathrm{NTK}}$ after training for different η. Similarly with Baratin et al. (2021), when η is sufficiently large (orange region), we obtain significant alignments which suggest potential feature learning. These transitions of leading eigenvalue and eigenvector alignment have been proved for $\boldsymbol{W}_t$ by Ba et al. (2022) for a different scenario[4].

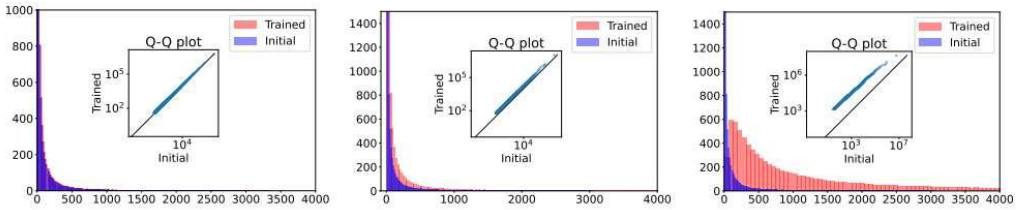
Spikes of Kernel Matrices. The alignment of the kernel matrix with the training labels $\boldsymbol{y}$ is defined by (Cristianini et al., 2001) by Kernel Target Alignment (KTA) as follows: when kernel $\boldsymbol{K}$ is either CK or NTK,

$$\mathrm{KTA} = \frac{\langle \boldsymbol{K}, \boldsymbol{y}^\top\boldsymbol{y}\rangle}{\|\boldsymbol{K}\|_F\,\|\boldsymbol{y}\|^2}. \tag{87}$$

Analogously to Baratin et al. (2021); Atanasov et al. (2022); Shan & Bordelon (2021), Figure 21(a) depicts the evolution of KTA of CK in several cases. Based on Figure 20(d), when the spike appears outside the bulk (Case 3), its corresponding (leading) eigenvector $\boldsymbol{v}_1$ of kernel matrix naturally dominates the alignment with $\boldsymbol{y}$ (Figure 33 in Appendix B.2), which is regarded as a kernel rotation during training in Ortiz-Jiménez et al. (2021). Notice that this is not the common situation in Cases 1&2 of Table 8 (and cf. Figure 29 in Appendix B.2). On the other hand, KTA measures the alignment

[4]We apply NTK parameterization for our neural networks and train both layers until convergence, while Ba et al. (2022) considers the mean-field initialization and early stage of training dynamics of GD for the first layer.

DOI 10.48550/arXiv.2211.06506



(a) GD (b) SGD (c) Adam

Figure 22: Different NTK spectra for a small-CNN model on CIFAR-2. The subplots are Q-Q plots for the comparison between initial and trained spectra. Test accuracies: (a) 79%, (b) 84%, (c) 86.4%.

between $\boldsymbol{y}$ and the full eigenbasis of the kernel. These kernel alignments improve the speed of the convergence of training dynamics but may hurt or boost the generalization of the NNs (Ortiz-Jiménez et al., 2021; Shan & Bordelon, 2021; Baratin et al., 2021). Figure 21(a) indicates that Case 4 with heavy-tailed spectra after training has a larger KTA than the other cases. In this case, the emergence of a heavy tail in the spectrum is closely related to a better generalization of the NN and more significant feature learning.

4.3 Phenomenon of Heavy-tailed Spectra

Next, we analyze the heavy-tailed spectra of weight and kernel matrices in Figure 19(c). Mahoney & Martin (2019); Martin & Mahoney (2021) found a strong correlation between the heavy-tailed spectra of trained state-of-the-art models with better generalization (Figure 25 in Appendix B.1). Heavy-tailed spectra can be viewed as an extreme of "bulk+spikes", where a fraction of the eigenvalues move out of the initial bulk. In RMT, heavy-tailed spectra generally appear when the entries of the matrix are highly correlated Martin & Mahoney (2021). This could heuristically explain heavy-tailed phenomena in the spectra since the entries of well-trained $\boldsymbol{W}_t$ should be strongly correlated. Unlike Mahoney & Martin (2019); Martin & Mahoney (2021), we focus on the heavy-tailed phenomena for both weight and kernel matrices in a simpler model (79) and provide a connection between feature learning and heavy-tailed spectra, which opens an important avenue for further theoretical analysis.

Heavy Tails and Generalization. We emphasize that heavy tails are not sufficient for good generalization, in general, Martin et al. (2021); Meng & Yao (2023). Figures 21(c)&(d) exhibit NNs with heavy-tailed weights but in the absence of good performance at initialization. In fact, it is the alignments between the features learned from the heavy-tailed part and the features in the teacher model that finally determine the generalization error of NNs.

More precisely, we provide an example of when heavy tails indicate better generalizations. Consider the multiple-index teacher model (80) with $k = 5$ feature directions $\boldsymbol{\beta}_i$, and train NNs (79) with GD, SGD, and Adam to get invariant bulk, bulk with one spike and heavy tails, respectively, after training. In Figure 21(b), we present the evolutions of the *principle angles* θ_i between feature subspace $U = \text{span}\{\boldsymbol{\beta}_i\}_{i=1}^k$ and top 100 eigenspace of $\boldsymbol{W}_t^\top \boldsymbol{W}_t$ during different training processes. This eigenspace with respect to the top 100 eigenvalues of $\boldsymbol{W}_t^\top \boldsymbol{W}_t$ corresponds to the heavy-tail part of the spectrum in $\boldsymbol{W}_t^\top \boldsymbol{W}_t$ when training NNs with Adam (solid lines in Figure 21(b)). Comparing with GD and SGD training processes, we observe strong alignments between feature space U and eigenspace w.r.t heavy tails in Adam case in Figure 21(b), which explains why Adam case (NNs with heavy-tailed spectra) generalizes better than the other two cases. For more examples, see Figures 35 and 38 in Appendix B.4. This concludes that NNs with heavy-tailed spectra can generalize better only when the teacher features from data are aligned with the heavy-tailed part of spectra. If the feature dimension in the teacher model is high (i.e. the teacher model is more complicated and intrinsically high-dimensional), then we expect to get a heavy-tailed weight spectrum of well-trained NN where the heavy-tailed part learns all the features in the teacher modes. This example explains why we can use the heavy tails to discriminate well-trained and poorly-trained large models Martin et al. (2021); Meng & Yao (2023); Yang et al. (2022).

DOI 10.48550/arXiv.2211.06506

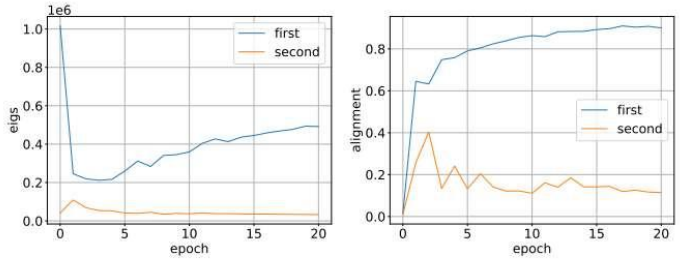
5 Discussions and Future Directions

We empirically investigated how the spectra of $\boldsymbol{W}$, $\boldsymbol{K}^{\mathrm{CK}}$, and $\boldsymbol{K}^{\mathrm{NTK}}$ evolve under the LWR for an idealized student-teacher setting. Our work implies that understanding the relationship between feature learning and training processes requires understanding the evolution of the spectra of both weight and kernel matrices. In particular, we show that different training processes affect the eigenstructure of weight and kernel matrices. Since evolution is sensitive to feature learning, we can link feature learning and different training dynamics by studying the spectra of these matrices.

While synthetic data is easier to analyze theoretically, we also investigate these spectral properties on real-world data and more complicated tasks in the following. In practice, people mainly focus on analyzing spectra of the weight matrices in fully connected layers; we choose to also focus on the spectral properties of general kernel matrices induced by the NNs, which contain abundant information (Chen et al., 2020; Long, 2021; Atanasov et al., 2022; Shan & Bordelon, 2021).

First, we show the spectra of $\boldsymbol{K}^{\mathrm{NTK}}$ before and after training for binary classification on CIFAR-2 through small CNNs in Figure 22. Similarly with Case 1, Figure 22(a) (especially in the Q-Q subplot) manifests the invariant spectral distribution of NTK through GD training while SGD exhibits a heavier tail in NTK spectrum in Figure 22(b). This phenomenon is more evident when trained by Adam in Figure 22(c) with improved accuracy. Figure 22 suggests that our observations on synthetic data in Section 3 can be extended to real-world data and on more practical architectures. We note that there is a lack of the emergence of spikes after training because spikes already exist in the initial NTK spectrum for this complicated neural architecture on real-world datasets. Figure 22(a) also indicates that the spectral invariance of NTK through training will impede the feature learning and the NN does not generalize well in this training process.

We also investigate the spectral properties on the pre-trained model, BERT from Devlin et al. (2018), with fine-tuning on Sentiment140 dataset of tweets[5] from Go et al. (2009). We fine-tune the BERT model for a binary classifier on Sentiment140 and capture the evolution of CK spectra, rather than the NTK due to the size of BERT, in Figure 23 (see also Figure 25 in Appendix B.1).



(a) Eigenvalues (b) Alignments

Figure 23: We use SGD for fine-tuning the BERT model. The training accuracy is 95.90% and the test accuracy is 84%. (a) The evolution of first and second eigenvalues of empirical CK during fine-tuning. (b) The alignments of training labels with first and second eigenvectors of CK during fine-tuning. See Figure 25 for the spectra of CK at different epochs.

A heavy-tailed CK spectrum with several spikes already exists in this pre-trained model. Unlike Figure 22 (and cases in Table 8) where the first spike of NTK becomes larger than at random initialization after training, in Figure 23(a), the leading eigenvalue first decreases and then increases. Moreover, similarly to Figure 20(d), our Figure 23(b) shows that the alignment of the first eigenvector of the CK and training labels becomes more apparent through fine-tuning with the leading eigenvalue decrease. Heuristically, this process seems to unlearn the features in the pre-trained model and, remarkably, learn new features on the new dataset in only a few epochs of fine-tuning (see Figure 25in Appendix B.1). We believe that the evolutions of the kernel matrices and some spectral metrics are crucial for understanding feature learning through fine-tuning (Wei et al., 2022). A more comprehensive exploration of the evolutionary spectral properties of "foundation models" may help shed further light on these phenomena.

[5] https://www.kaggle.com/datasets/kazanova/sentiment140

DOI 10.48550/arXiv.2211.06506

Limitations. Although LWR has garnered significant attention in recent years, e.g., Schröder et al. (2023); Li & Sompolinsky (2021); Bosch et al. (2023); Zavatone-Veth et al. (2022); Cui et al. (2023), we recognize the limitations inherent in LWR. Our LWR is more realistic compared with infinite-width neural networks and is one of the ways to approximate finite but very large neural networks with very large datasets, but there are more sophisticated regimes for NNs. We leave this for future theoretical work. The NTK parameterization is another limitation of this work. We expect to apply our spectral analysis for other parameterizations of NNs with more real-world datasets. See the discussion at the beginning of Appendix B.

Acknowledgement

Z.W., A.E., I.D., and T.C. were partially supported by the Mathematics for Artificial Reasoning in Science (MARS) initiative via the Laboratory Directed Research and Development (LDRD) Program at Pacific Northwest National Laboratory (PNNL). A.S. and T.C. were also partially supported by the Statistical Inference Generates kNowledge for Artificial Learners (SIGNAL) program at PNNL. PNNL is a multi-program national laboratory operated for the U.S. Department of Energy (DOE) by Battelle Memorial Institute under Contract No. DE-AC05-76RL0-1830. Z.W. would like to thank Denny Wu and Libin Zhu for their valuable suggestions and comments.

6.0 References

Akyürek, E., Bolukbasi, T., Liu, F., et al. 2022, in Findings of the Association for Computational Linguistics: EMNLP 2022 (Abu Dhabi, United Arab Emirates: Association for Computational Linguistics), 2429–2446. https://aclanthology.org/2022.findings-emnlp.180

Akyürek, E., Schuurmans, D., Andreas, J., Ma, T., & Zhou, D. 2023, in The Eleventh International Conference on Learning Representations. https://openreview.net/forum?id=0g0X4H8yN4I

Alvarez, M. A., Rosasco, L., & Lawrence, N. D. 2011, arXiv e-prints, arXiv:1106.6251, doi: 10.48550/arXiv.1106.6251

Atanasov, A., Bordelon, B., & Pehlevan, C. 2022, in International Conference on Learning Representations. https://openreview.net/forum?id=1NvflqAdoom

Balestriero, R., & Baraniuk, R. 2018, in International Conference on Machine Learning

Bartle, R. G., & Sherbert, D. R. 2011, Introduction to Real Analysis (4th Edition) (Wiley)

Bell, B., Geyer, M., Glickenstein, D., Fernandez, A., & Moore, J. 2023, An Exact Kernel Equivalence for Finite Classification Models. https://arxiv.org/abs/2308.00824

Bommasani, R., Hudson, D. A., Adeli, E., et al. 2021, arXiv e-prints, arXiv:2108.07258, doi: 10.48550/arXiv.2108.07258

Chen, C., Li, O., Barnett, A. J., Su, J., & Rudin, C. 2018, in Neural Information Processing Systems. https://api.semanticscholar.org/CorpusID:49482223

Chen, X., Hsieh, C.-J., & Gong, B. 2021, ArXiv, abs/2106.01548

Chen, X., Hsieh, C.-J., & Gong, B. 2022, in International Conference on Learning Representations. https://openreview.net/forum?id=LtKcMgGOeLt

Chizat, L., Oyallon, E., & Bach, F. R. 2018, in Neural Information Processing Systems

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. 2019, in Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers) (Minneapolis, Minnesota: Association for Computational Linguistics), 4171–4186. https://aclanthology.org/N19-1423

Domingos, P. 2020, arXiv e-prints, arXiv:2012.00152. https://arxiv.org/abs/2012.00152

Dziedzic, A., Rabanser, S., Yaghini, M., et al. 2022, arXiv e-prints, arXiv:2207.12545, doi: 10.48550/ arXiv.2207.12545

Fan, Z., & Wang, Z. 2020, Advances in neural information processing systems, 33, 7710

Fort, S., Dziugaite, G. K., Paul, M., et al. 2020, Advances in Neural Information Processing Systems, 33, 5850

Ghojogh, B., Ghodsi, A., Karray, F., & Crowley, M. 2021, ArXiv, abs/2106.08443

Glorot, X., Bordes, A., & Bengio, Y. 2011, in International Conference on Artificial Intelligence and Statistics. https://api.semanticscholar.org/CorpusID:2239473

Gu, T., Liu, K., Dolan-Gavitt, B., & Garg, S. 2019, IEEE Access, 7, 47230, doi: 10.1109/ACCESS. 2019.2909068

Hanawa, K., Yokoi, S., Hara, S., & Inui, K. 2021, in International Conference on Learning Representations. https://openreview.net/forum?id=9uvhpyQwzM_ 10 Under review as a conference paper at ICLR 2024

He, K., Zhang, X., Ren, S., & Sun, J. 2015a, 2015 IEEE International Conference on Computer Vision (ICCV), 1026

Hilbert, D. V. 1904. https://api.semanticscholar.org/CorpusID:121363250

Hofmann, T., Schölkopf, B., & Smola, A. 2007, Annals of Statistics, 36, 1171

Jacot, A., Gabriel, F., & Hongler, C. 2018, Advances in neural information processing systems, 31

Johnson, W. B., & Lindenstrauss, J. 1984, Contemporary Mathematics, 26

Kendall, M. G. 1938, Biometrika, 30, 81, doi: 10.1093/biomet/30.1-2.81

Koh, P. W., & Liang, P. 2017, in International conference on machine learning, PMLR, 1885–1894

Krizhevsky, A., & Hinton, G. 2009, Master's thesis, Department of Computer Science, University of Toronto

Lai, V., Chen, C., Liao, Q. V., Smith-Renner, A., & Tan, C. 2021, ArXiv, abs/2112.11471

Leavitt, M. L., & Morcos, A. 2020, arXiv e-prints, arXiv:2010.12016, doi: 10.48550/arXiv.2010.12016

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. 1998, Proceedings of the IEEE, 86, 2278, doi: 10. 1109/5.726791

Lee, J., Sohl-dickstein, J., Pennington, J., et al. 2018, in International Conference on Learning Representations. https://openreview.net/forum?id=B1EA-M-0Z

Lee, J., Xiao, L., Schoenholz, S. S., et al. 2020, Journal of Statistical Mechanics: Theory and Experiment, 2020, 124002, doi: 10.1088/1742-5468/abc62b

Lin, R., Liu, W., Liu, Z., et al. 2019, 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 6916

Long, P. M. 2021, arXiv preprint arXiv:2105.10585

Loo, N., Hasani, R., Amini, A., & Rus, D. 2022, Evolution of Neural Tangent Kernels under Benign and Adversarial Training

Loshchilov, I., & Hutter, F. 2017, arXiv e-prints, arXiv:1711.05101, doi: 10.48550/arXiv.1711.05101

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. 2019, ICLR

Mohamadi, M. A., & Sutherland, D. J. 2022, arXiv e-prints, arXiv:2206.12543, doi: 10.48550/arXiv. 2206.12543

Nair, V., & Hinton, G. E. 2010, in International Conference on Machine Learning. https://api.semanticscholar.org/CorpusID:15539264

Novak, R., Sohl-Dickstein, J., & Schoenholz, S. S. 2022, in International Conference on Machine Learning, PMLR, 17018–17044

Ortiz-Jiménez, G., Moosavi-Dezfooli, S.-M., & Frossard, P. 2021, Advances in Neural Information Processing Systems, 34, 8998

Papernot, N., & McDaniel, P. 2018, arXiv e-prints, arXiv:1803.04765. https://arxiv.org/abs/1803.04765 11 Under review as a conference paper at ICLR 2024

Papernot, N., McDaniel, P. D., & Goodfellow, I. J. 2016a, CoRR

Papernot, N., McDaniel, P. D., Goodfellow, I. J., et al. 2016b, CoRR

Papyan, V., Han, X., & Donoho, D. L. 2020, Proceedings of the National Academy of Sciences of the United States of America, 117, 24652

Park, S. M., Georgiev, K., Ilyas, A., Leclerc, G., & Madry, A. 2023, arXiv e-prints, arXiv:2303.14186. https://arxiv.org/abs/2303.14186

Paszke, A., Gross, S., Massa, F., et al. 2019, in Advances in Neural Information Processing Systems 32, ed. H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alche-Buc,

E. Fox, & R. Garnett (Curran Associates, Inc.), 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Pedregosa, F., Varoquaux, G., Gramfort, A., et al. 2011, Journal of Machine Learning Research, 12, 2825

Phan, H. 2021, huyvnphan/PyTorch_CIFAR10, v3.0.1, Zenodo, doi: 10.5281/zenodo.4431043. https: //doi.org/10.5281/zenodo.4431043

Pruthi, G., Liu, F., Kale, S., & Sundararajan, M. 2020, in Advances in Neural Information Processing Systems, ed. H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, & H. Lin, Vol. 33 (Curran Associates, Inc.), 19920–19930. https://proceedings.neurips.cc/paper_files/paper/2020/file/e6385d39ec9394f2f3a354d9d2b88eec-Paper.pdf

Radhakrishnan, A., Beaglehole, D., Pandit, P., & Belkin, M. 2022, arXiv preprint arXiv:2212.13881

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. 2018, in Proceedings of the IEEE conference on computer vision and pattern recognition, 4510–4520

Schmitz, G., Aldrich, C., & Gouws, F. 1999, IEEE Transactions on Neural Networks, 10, 1392, doi: 10.1109/72.809084

Scholkopf, B., & Smola, A. 2001, in Adaptive computation and machine learning series. https: //api.semanticscholar.org/CorpusID:52872213

Shan, S., Bhagoji, A. N., Zheng, H., & Zhao, B. Y. 2022, in 31st USENIX Security Symposium (USENIX Security 22), 3575–3592

Torralba, A., Fergus, R., & Freeman, W. T. 2008, IEEE Transactions on Pattern Analysis and Machine Intelligence, 30, 1958

Tsilivis, N., & Kempe, J. 2023, What Can the Neural Tangent Kernel Tell Us About Adversarial Robustness?

Vapnik, V. N. 2000, in Statistics for Engineering and Information Science. https://api.semanticscholar. org/CorpusID:7138354

Virtanen, P., Gommers, R., Oliphant, T. E., et al. 2020, Nature Methods, 17, 261, doi: 10.1038/ s41592-019-0686-2

Vyas, N., Bansal, Y., & Nakkiran, P. 2022, arXiv e-prints, arXiv:2206.10012. https://arxiv.org/abs/ 2206.10012

Wang, A., Singh, A., Michael, J., et al. 2018, in BlackboxNLP@EMNLP. https://api.semanticscholar. org/CorpusID:5034059

Wang, R., Chen, T., & Hero, A. 2021, arXiv e-prints, arXiv:2108.06797, doi: 10.48550/arXiv.2108. 06797 12 Under review as a conference paper at ICLR 2024

Wang, Z., Engel, A., Sarwate, A., Dumitriu, I., & Chiang, T. 2022, arXiv preprint arXiv:2211.06506

Warstadt, A., Singh, A., & Bowman, S. R. 2018, Transactions of the Association for Computational Linguistics, 7, 625

Wolf, T., Debut, L., Sanh, V., et al. 2019, arXiv e-prints, arXiv:1910.03771, doi: 10.48550/arXiv. 1910.03771

Xiao, H., Rasul, K., & Vollgraf, R. 2017, ArXiv, abs/1708.07747

Yang, F., Huang, Z., Scholtz, J., & Arendt, D. L. 2020, in Proceedings of the 25th International Conference on Intelligent User Interfaces, 189–201

Yang, G., & Hu, E. J. 2021, in Proceedings of Machine Learning Research, Vol. 139, Proceedings of the 38th International Conference on Machine Learning, ed. M. Meila & T. Zhang (PMLR), 11727–11737. https://proceedings.mlr.press/v139/yang21c.html

Yeh, C.-K., Kim, J. S., Yen, I. E. H., & Ravikumar, P. 2018, arXiv e-prints, arXiv:1811.09720, doi: 10.48550/arXiv.1811.09720

Arora, S., Du, S. S., Hu, W., et al. 2019, Advances in neural information processing systems, 32

Bartlett, P. L., Montanari, A., & Rakhlin, A. 2021, Acta numerica, 30, 87

Belkin, M. 2021, Acta Numerica, 30, 203

Boser, B. E., Guyon, I. M., & Vapnik, V. N. 1992, in Proceedings of the fifth annual workshop on Computational learning theory, 144–152

Cybenko, G. 1989, Mathematics of control, signals and systems, 2, 303

Daubechies, I., DeVore, R., Foucart, S., Hanin, B., & Petrova, G. 2022, Constructive Approximation, 55, 127

Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. 2023, QLoRA: Efficient Finetuning of Quantized LLMs. https://arxiv.org/abs/2305.14314

Engel, A., Wang, Z., Frank, N. S., et al. 2023, arXiv preprint arXiv:2305.14585

Fan, Z., & Wang, Z. 2020, Advances in neural information processing systems, 33, 7710

Go, A., Bhayani, R., & Huang, L. 2009, in Stanford CS224N Project Report. https://api.semanticscholar.org/CorpusID:18635269

He, K., Zhang, X., Ren, S., & Sun, J. 2016, in Proceedings of the IEEE conference on computervision and pattern recognition, 770–778

Hornik, K., Stinchcombe, M., & White, H. 1989, Neural networks, 2, 359

Jacot, A., Gabriel, F., & Hongler, C. 2018, Advances in neural information processing systems, 31

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., et al. 2021, Nature Reviews Physics, 3, 422

Kenton, J. D. M.-W. C., & Toutanova, L. K. 2019, in Proceedings of naacL-HLT, Vol. 1, 2

Kim, Y. 2014, arXiv preprint arXiv:1408.5882

Kovachki, N., Lanthaler, S., & Mishra, S. 2021, The Journal of Machine Learning Research, 22, 13237

Krizhevsky, A., Sutskever, I., & Hinton, G. E. 2012, Advances in neural information processing systems, 25

Lee, J., Xiao, L., Schoenholz, S., et al. 2019, Advances in neural information processing systems, 32

Liu, C., Zhu, L., & Belkin, M. 2020a, arXiv preprint arXiv:2003.00307, 7

Lu, L., Jin, P., Pang, G., Zhang, Z., & Karniadakis, G. E. 2021, Nature machine intelligence, 3, 218

Meanti, G., Carratino, L., Rosasco, L., & Rudi, A. 2020, Advances in Neural Information Processing

Systems, 33, 14410

Mikolov, T., Chen, K., Corrado, G., & Dean, J. 2013, arXiv preprint arXiv:1301.3781

Novak, R., Sohl-Dickstein, J., & Schoenholz, S. S. 2022, in International Conference on Machine

Learning, PMLR, 17018–17044

Novak, R., Xiao, L., Lee, J., et al. 2018, arXiv preprint arXiv:1810.05148

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. 2018, Improving Language Understanding by Generative Pre-Training. https://api.semanticscholar.org/CorpusID:49313245

Radford, A., Wu, J., Child, R., et al. 2019, Language Models are Unsupervised Multitask Learners.https://api.semanticscholar.org/CorpusID:160025533

Scholkopf, B., Herbrich, R., & Smola, A. J. 2001, in International conference on computational ¨

learning theory, Springer, 416–426

Tsou, A., Vargas, M., Engel, A., & Chiang, T. 2023, In Preparation

Vaswani, A., Shazeer, N., Parmar, N., et al. 2017, Advances in neural information processing systems, 30

Wolf, T., Debut, L., Sanh, V., et al. 2019, CoRR, abs/1910.03771

Yarotsky, D. 2018, in Conference on learning theory, PMLR, 639–649

Adlam, B., & Pennington, J. 2020, ArXiv, abs/2011.03321

Alain, G., & Bengio, Y. 2016, ArXiv, abs/1610.01644

Almazrouei, E., Alobeidli, H., Alshamsi, A., et al. 2023, Falcon-40B: an open large language model with state-of-the-art performance, Tech. rep., Technical report, Technology Innovation Institute

Belinkov, Y. 2021, Computational Linguistics, 48, 207

Chen, E., Lerman, K., & Ferrara, E. 2020a, JMIR Public Health Surveill, 6, e19273, doi: 10.2196/19273

Chen, M., Radford, A., Wu, J., et al. 2020b, in International Conference on Machine Learning. https://api.semanticscholar.org/CorpusID:219781060

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., & Bordes, A. 2017, ArXiv, abs/1705.02364

Conneau, A., Kruszewski, G., Lample, G., Barrault, L., & Baroni, M. 2018, in Annual Meeting of the Association for Computational Linguistics. https://api.semanticscholar.org/CorpusID:24461982

Daniely, A., Frostig, R., & Singer, Y. 2016, in NIPS. https://api.semanticscholar.org/CorpusID:217536627

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. 2019, ArXiv, abs/1810.04805

D'Andrea, A., Ferri, F., Grifoni, P., & Guzzo, T. 2015, International Journal of Computer Applications

Evci, U., Dumoulin, V., Larochelle, H., & Mozer, M. C. 2022, in International Conference on Machine Learning. https://api.semanticscholar.org/CorpusID:245837741

Fort, S., Ren, J. J., & Lakshminarayanan, B. 2021, in Neural Information Processing Systems. https://api.semanticscholar.org/CorpusID:235358891

Giryes, R., Sapiro, G., & Bronstein, A. M. 2015, IEEE Transactions on Signal Processing, 64, 3444

Go, A., Bhayani, R., & Huang, L. 2009, in Stanford CS224N Project Report. https://api.semanticscholar.org/CorpusID:18635269

Jarrett, K., Kavukcuoglu, K., Ranzato, M., & LeCun, Y. 2009, in 2009 IEEE 12th International Conference on Computer Vision, ICCV 2009, Proceedings of the IEEE International Conference on Computer Vision, 2146–2153

Kumar, A., Raghunathan, A., Jones, R., Ma, T., & Liang, P. 2022, in International Conference on Learning Representations. https://api.semanticscholar.org/CorpusID:247011290

Liu, Y., Ott, M., Goyal, N., et al. 2019, ArXiv, abs/1907.11692

Maas, A. L., Daly, R. E., Pham, P. T., et al. 2011, in Annual Meeting of the Association for Computational Linguistics. https://api.semanticscholar.org/CorpusID:1428702

Medhat, W., Hassan, A., & Korashy, H. 2014, Ain Shams Engineering Journal

Mei, S., & Montanari, A. 2019, Communications on Pure and Applied Mathematics, 75

Paszke, A., Gross, S., Massa, F., et al. 2019, in Neural Information Processing Systems. https://api.semanticscholar.org/CorpusID:202786778

Radford, A., Jozefowicz, R., & Sutskever, I. 2017, ArXiv, abs/1704.01444

Radford, A., Kim, J. W., Xu, T., et al. 2022, ArXiv, abs/2212.04356

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. 2018, Improving Language Understanding by Generative Pre-Training. https://api.semanticscholar.org/CorpusID:49313245

Radford, A., Wu, J., Child, R., et al. 2019, Language Models are Unsupervised Multitask Learners. https://api.semanticscholar.org/CorpusID:160025533

Radford, A., Kim, J. W., Hallacy, C., et al. 2021, in International Conference on Machine Learning. https://api.semanticscholar.org/CorpusID:231591445

Raffel, C., Shazeer, N. M., Roberts, A., et al. 2019, ArXiv, abs/1910.10683

Rosenfeld, E., Ravikumar, P., & Risteski, A. 2022, ArXiv, abs/2202.06856

Socher, R., Perelygin, A., Wu, J., et al. 2013, in Conference on Empirical Methods in Natural Language Processing. https://api.semanticscholar.org/CorpusID:990233

Wang, A., Pruksachatkun, Y., Nangia, N., et al. 2019, ArXiv, abs/1905.00537

Wang, A., Singh, A., Michael, J., et al. 2018, in BlackboxNLP@EMNLP. https://api.semanticscholar.org/CorpusID:5034059

Wolf, T., Debut, L., Sanh, V., et al. 2019, CoRR, abs/1910.03771

Yang, Z., Dai, Z., Yang, Y., et al. 2019, in Neural Information Processing Systems. https://api.semanticscholar.org/CorpusID:195069387

Yehudai, G., & Shamir, O. 2019, in Neural Information Processing Systems. https://api.semanticscholar.org/CorpusID:90262791

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. 2014, in NIPS. https://api.semanticscholar.org/CorpusID:362467

Zhang, K. W., & Bowman, S. R. 2018, ArXiv, abs/1809.10040

Zhang, L., Wang, S., & Liu, B. 2018, WIREs Data Mining and Knowledge Discovery

Zhou, C., Liu, P., Xu, P., et al. 2023, ArXiv, abs/2305.11206

Alain, G., & Bengio, Y. 2018, arXiv preprint arXiv:1610.01644v4

Ansuini, A., Laio, A., Macke, J. H., & Zoccolan, D. 2019, Advances in Neural Information Processing Systems, 32

Ben-Shaul, I., & Dekel, S. 2022, in Topological, Algebraic and Geometric Learning Workshops 2022, PMLR, 37–47

Cohen, G., Sapiro, G., & Giryes, R. 2018, arXiv preprint arXiv:1805.06822

Galanti, T., Galanti, L., & Ben-Shaul, I. 2022, arXiv preprint arXiv:2202.09028

He, K., Zhang, X., Ren, S., & Sun, J. 2016, in Proceedings of the IEEE conference on computer vision and pattern recognition, 770–778

Krizhevsky, A., Hinton, G., et al. 2009

Montavon, G., Braun, M. L., & Müller, K.-R. 2011, Journal of Machine Learning Research, 12

Papyan, V., Han, X., & Donoho, D. L. 2020, Proceedings of the National Academy of Sciences, 117, 24652

Phan, H. 2021, huyvnphan/PyTorch_CIFAR10, v3.0.1, Zenodo, doi: 10.5281/zenodo.4431043. https://doi.org/10.5281/zenodo.4431043

Raghu, M., Gilmer, J., Yosinski, J., & Sohl-Dickstein, J. 2017, Advances in neural information processing systems, 30

Rangamani, A., Lindegaard, M., Galanti, T., & Poggio, T. A. 2023, in International Conference on Machine Learning, PMLR, 28729–28745

Recanatesi, S., Farrell, M., Advani, M., et al. 2019, arXiv preprint arXiv:1906.00443

Zhang, M., Wu, W., Zhang, Y., et al. 2017, arXiv preprint arXiv:1711.01573

Adlam, B., & Pennington, J. 2020, in International Conference on Machine Learning, PMLR, 74–84

Allen-Zhu, Z., Li, Y., & Liang, Y. 2019, Advances in neural information processing systems, 32

Andriushchenko, M., Varre, A. V., Pillaud-Vivien, L., & Flammarion, N. 2023, in International Conference on Machine Learning, PMLR, 903–925

Arora, S., Du, S. S., Hu, W., et al. 2019, Advances in Neural Information Processing Systems, 32

Atanasov, A., Bordelon, B., & Pehlevan, C. 2022, in International Conference on Learning Representations. https://openreview.net/forum?id=1NvflqAdoom

Ba, J., Erdogdu, M. A., Suzuki, T., et al. 2022, Advances in Neural Information Processing Systems, 35, 37932

Bai, Y., & Lee, J. D. 2020, in International Conference on Learning Representations. https://openreview.net/forum?id=rkllGyBFPH

Bai, Z., & Silverstein, J. W. 2010, Spectral analysis of large dimensional random matrices, Vol. 20 (Springer)

Baik, J., Arous, G. B., & Péché, S. 2005, The Annals of Probability, 33, 1643

Baratin, A., George, T., Laurent, C., et al. 2021, in International Conference on Artificial Intelligence and Statistics, PMLR, 2269–2277

Bartlett, P. L., Montanari, A., & Rakhlin, A. 2021, Acta numerica, 30, 87

Belkin, M., Ma, S., & Mandal, S. 2018, in International Conference on Machine Learning, PMLR, 541–549

Benigni, L., & Péché, S. 2021, Electronic Journal of Probability, 26, 1

Beugnot, G., Mairal, J., & Rudi, A. 2022, in Conference on Learning Theory, PMLR, 254–282

Bietti, A., & Mairal, J. 2019, in Advances in Neural Information Processing Systems, 12873–12884

Bosch, D., Panahi, A., & Hassibi, B. 2023, arXiv preprint arXiv:2302.06210

Chatterjee, S. 2022, arXiv preprint arXiv:2203.16462

Chen, S., He, H., & Su, W. 2020, Advances in Neural Information Processing Systems, 33

Chizat, L., & Bach, F. 2018, in Advances in neural information processing systems, 3036–3046

Chizat, L., Oyallon, E., & Bach, F. 2019, Advances in Neural Information Processing Systems, 32

Cristianini, N., Shawe-Taylor, J., Elisseeff, A., & Kandola, J. 2001, Advances in neural information processing systems, 14

Cui, H., Krzakala, F., & Zdeborova, L. 2023, in Proceedings of Machine Learning Research, Vol. 202, Proceedings of the 40th International Conference on Machine Learning (PMLR), 6468–6521. https://proceedings.mlr.press/v202/cui23b.html

Damian, A., Lee, J., & Soltanolkotabi, M. 2022, in Conference on Learning Theory, PMLR, 5413–5452

Daniely, A., & Malach, E. 2020, Advances in Neural Information Processing Systems, 33, 20356

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. 2018, arXiv preprint arXiv:1810.04805

Du, S., Lee, J., Li, H., Wang, L., & Zhai, X. 2019, in International conference on machine learning, PMLR, 1675–1685

Du, S. S., Zhai, X., Poczos, B., & Singh, A. 2018, in International Conference on Learning Representations

Dyer, E., & Gur-Ari, G. 2020, in International Conference on Learning Representations. https://openreview.net/forum?id=S1gFvANKDS

Fan, Z., & Wang, Z. 2020, Advances in neural information processing systems, 33

Fort, S., Dziugaite, G. K., Paul, M., et al. 2020, Advances in Neural Information Processing Systems, 33, 5850

Geiger, M., Spigler, S., Jacot, A., & Wyart, M. 2020, Journal of Statistical Mechanics: Theory and Experiment, 2020, 113301

Gerace, F., Loureiro, B., Krzakala, F., Mézard, M., & Zdeborová, L. 2020, in International Conference on Machine Learning, PMLR, 3452–3462

Ghorbani, B., Mei, S., Misiakiewicz, T., & Montanari, A. 2020, Advances in Neural Information Processing Systems, 33, 14820

Go, A., Bhayani, R., & Huang, L. 2009, CS224N project report, Stanford, 1, 2009

Hanin, B., & Nica, M. 2020, Communications in Mathematical Physics, 376, 287

Hu, W., Xiao, L., Adlam, B., & Pennington, J. 2020, Advances in Neural Information Processing Systems, 33, 17116

Huang, J., & Yau, H.-T. 2020, in International Conference on Machine Learning, PMLR, 4542–4551

Jacot, A., Gabriel, F., & Hongler, C. 2018, Advances in neural information processing systems, 31

Jastrzebski, S., Szymczak, M., Fort, S., et al. 2020, in International Conference on Learning Representations. https://openreview.net/forum?id=r1g87C4KwB

Karp, S., Winston, E., Li, Y., & Singh, A. 2021, Advances in Neural Information Processing Systems, 34

Kingma, D. P., & Ba, J. 2014, arXiv preprint arXiv:1412.6980

Kopitkov, D., & Indelman, V. 2020, in International Conference on Artificial Neural Networks,

Springer, 168–179

Le Cun, Y., Kanter, I., & Solla, S. A. 1991, Physical Review Letters, 66, 2396

Leclerc, G., & Madry, A. 2020, arXiv preprint arXiv:2002.10376

Lee, J., Schoenholz, S., Pennington, J., et al. 2020, Advances in Neural Information Processing

Systems, 33, 15156

Lee, J., Sohl-dickstein, J., Pennington, J., et al. 2018, in International Conference on Learning Representations. https://openreview.net/forum?id=B1EA-M-0Z

Lewkowycz, A., Bahri, Y., Dyer, E., Sohl-Dickstein, J., & Gur-Ari, G. 2020, arXiv preprint arXiv:2003.02218

Li, Q., & Sompolinsky, H. 2021, Physical Review X, 11, 031059

Li, Y., Ma, T., & Zhang, H. R. 2020, in Conference on learning theory, PMLR, 2613–2682

Li, Y., Wei, C., & Ma, T. 2019, in Advances in Neural Information Processing Systems, 11674–11685

Liao, Z., Couillet, R., & Mahoney, M. W. 2020, Advances in Neural Information Processing Systems, 33, 13939

Liu, C., Zhu, L., & Belkin, M. 2022, Applied and Computational Harmonic Analysis

Long, P. M. 2021, arXiv preprint arXiv:2105.10585

Loo, N., Hasani, R., Amini, A., & Rus, D. 2022, Advances in Neural Information Processing Systems, 35, 11642

Louart, C., Liao, Z., & Couillet, R. 2018, The Annals of Applied Probability, 28, 1190

Mahoney, M., & Martin, C. 2019, in International Conference on Machine Learning, PMLR, 4284–4293

Martin, C. H., & Mahoney, M. W. 2021, Journal of Machine Learning Research, 22, 1

Martin, C. H., Peng, T. S., & Mahoney, M. W. 2021, Nature Communications, 12, 1

Mei, S., & Montanari, A. 2022, Communications on Pure and Applied Mathematics, 75, 667

Mei, S., Montanari, A., & Nguyen, P.-M. 2018, Proceedings of the National Academy of Sciences, 115, E7665

Meng, X., & Yao, J. 2023, Journal of Machine Learning Research, 24, 1

Nakkiran, P. 2020, OPT2020 Workshop

Neal, R. M. 1995, Bayesian learning for neural networks, Vol. 118 (Springer Science & Business Media)

Nguyen, Q. 2021, in International Conference on Machine Learning, PMLR, 8056–8062

Ortiz-Jiménez, G., Modas, A., Moosavi, S.-M., & Frossard, P. 2020, Advances in Neural Information Processing Systems, 33, 17896

Ortiz-Jiménez, G., Moosavi-Dezfooli, S.-M., & Frossard, P. 2021, Advances in Neural Information Processing Systems, 34

Oymak, S., Fabian, Z., Li, M., & Soltanolkotabi, M. 2019, arXiv preprint arXiv:1906.05392

Oymak, S., & Soltanolkotabi, M. 2019, in International Conference on Machine Learning, PMLR, 4951–4960

Oymak, S., & Soltanolkotabi, M. 2020, IEEE Journal on Selected Areas in Information Theory, 1, 84

Pennington, J., & Bahri, Y. 2017, in International Conference on Machine Learning, PMLR, 2798–2806

Pennington, J., & Worah, P. 2017, Advances in neural information processing systems, 30

Polaczyk, B., & Cyranka, J. 2022, Transactions on Machine Learning Research

Refinetti, M., Goldt, S., Krzakala, F., & Zdeborová, L. 2021, in International Conference on Machine Learning, PMLR, 8936–8947

Schröder, D., Cui, H., Dmitriev, D., & Loureiro, B. 2023, in Proceedings of Machine Learning Research, Vol. 202, Proceedings of the 40th International Conference on Machine Learning (PMLR), 30285–30320. https://proceedings.mlr.press/v202/schroder23a.html

Shan, H., & Bordelon, B. 2021, arXiv preprint arXiv:2105.14301

Thamm, M., Staats, M., & Rosenow, B. 2022, Physical Review E, 106, 054124

Vershynin, R. 2018, High-dimensional probability: An introduction with applications in data science, Vol. 47 (Cambridge university press)

Wang, Z., & Zhu, Y. 2021, arXiv preprint arXiv:2109.09304

Wei, A., Hu, W., & Steinhardt, J. 2022, in Proceedings of Machine Learning Research, Vol. 162, Proceedings of the 39th International Conference on Machine Learning (PMLR), 23549–23588

Williams, C. 1996, Advances in neural information processing systems, 9

Yang, G. 2019, Advances in Neural Information Processing Systems, 32

Yang, G., & Hu, E. J. 2021, in International Conference on Machine Learning, PMLR, 11727–11737

Yang, Y., Theisen, R., Hodgkinson, L., et al. 2022, arXiv preprint arXiv:2202.02842

Zavatone-Veth, J. A., Tong, W. L., & Pehlevan, C. 2022, Physical Review E, 105, 064118

Zou, D., Cao, Y., Zhou, D., & Gu, Q. 2020, Machine learning, 109, 467

Pacific Northwest National Laboratory

902 Battelle Boulevard
P.O. Box 999
Richland, WA 99354

1-888-375-PNNL (7665)

www.pnnl.gov