NFT-K: NON-FUNGIBLE TANGENT KERNELS

Sina Alemohammad, Hossein Babaei, CJ Barberan, Naiming Liu,
Lorenzo Luzi, Blake Mason, Richard G. Baraniuk

Rice University

ABSTRACT

Deep neural networks have become essential for numerous applications due to their strong empirical performance such as vision, RL, and classification. Unfortunately, these networks are quite difficult to interpret, and this limits their applicability in settings where interpretability is important for safety, such as medical imaging. One type of deep neural network is neural tangent kernel that is similar to a kernel machine that provides some aspect of interpretability. To further contribute interpretability with respect to classification and the layers, we develop a new network as a combination of multiple neural tangent kernels, one to model each layer of the deep neural network individually as opposed to past work which attempts to represent the entire network via a single neural tangent kernel. We demonstrate the interpretability of this model on two datasets, showing that the multiple kernels model elucidates the interplay between the layers and predictions.

Index Terms— Deep neural network, neural tangent kernel, interpretability, multi-layer perception

1. INTRODUCTION

Deep neural networks (DNNs) have become the main parametric models used to solve machine learning problems as they provide state-of-the-art empirical performance, e.g., see [1–6]. DNNs can be used for a host of applications, including autonomous driving [7], medical imaging [8], medical records [9], and cyber security [10]. These applications all benefit from DNNs but are also high risk situations.

Despite the extraordinary performance of DNNs, it is still not clear how to best interpret their predictions and inferences. We direct the reader to this summary of recent efforts on interpretations [11]. Indeed there are some aspects within DNN interpretability that need more understanding or are poorly explained by prior work. A core challenge is to understand which of the layers are responsible for the which aspects of the decision-making process for each class.

Towards a deeper understanding of the functionality of neural networks, recent work has demonstrated that some DNNs correspond to kernel machines [12]. Kernel machines have been studied extensively and can provide extra interpretability depending on the structure of the kernel. Similar to their kernel counterparts, DNN architectures typically use a large number of parameters which can be orders of magnitude more than the training samples. Overparameterized DNNs generalize unseen data very well despite over-fitting training data, which is a property of interpolating methods such as kernel machines [13, 14]. The exact relationship between DNNs and kernel methods have been revealed by analysing the *infinite-width* DNNs, when the number of units (neurons) in the hidden layers goes to infinity. It has been shown that infinite-width DNNs trained with gradient descent converges to a kernel ridge regression predictor with respect to a kernel known as the *neural tangent kernel* (NTK) [15]. In recent years NTKs have been developed for a variety of architectures [16–23].

Though they live in potentially infinite dimensional spaces, kernel predictors can be reduced to linear classifiers in the eigenspace of the grammian of the input data. This is the famed "kernel trick" in machine learning and allows improved computational efficiency and interpretability of results. NTKs likewise use a gram matrix, it is easier to interpret the relationship between the input and the output of the DNN. However, it is not straightforward to use NTK to interpret which *layers* the network uses to classify a certain class via the gram matrix. The relationship between class and layer is not well understood but can be useful for identifying when misclassifications occur or if depth should be increased or decreased in a neural network. We begin to understand the relationship between classes and layers by modelling it using multiple NTKs, one for each layer.

1.1. Contributions

In order to capture the relationship between layers and classes, we combine NTK with inspiration from another network [24]. Our architecture admits multiple-kernel learning in the infinite-width limit, where each kernel corresponds to the NTK of each layer. We use the layers of multi-layer perceptrons (MLPs) and call our model the non-fungible tangent

ICASSP 2022

kernel MLP (NFT-K MLP) for our proposed architecture. This stems from our model learning to select layers of a DNN which are crucial for classification of certain classes (Figure 1). As the layers selected by the model are the most important, they are not fungible with the layers that were not selected, and this reduces the overall complexity of the model while ensuring strong empirical performance.

To validate our model, we conduct experiments on the MNIST dataset and the large movie review dataset to provide interpretable results into how our network provides insight into which layers are important for classification. We see that NFT-K MLPs focus on similar layers for classes which visually appear similar, even without the use of convolutional layers. Moreover, we can see which layers are the most important for classification; some layers are not at all important and are dropped from the model. Overall, NFT-K MLPs are able to model the relationship between classes and layers and provide insightful information.

2. BACKGROUND

Given an input $\boldsymbol{x} \in \mathbb{R}^m$, an MLP with L hidden layers and n neurons performs the following recursive relations

$$\boldsymbol{g}^{(1)}(\boldsymbol{x}) = \mathbf{W}^{(1)}\boldsymbol{x} + \mathbf{b}^{(1)}, \tag{1}$$

$$\boldsymbol{h}^{(\ell)}(\boldsymbol{x}) = \phi(\boldsymbol{g}^{(\ell)}(\boldsymbol{x})), \tag{2}$$

$$\boldsymbol{g}^{(\ell)}(\boldsymbol{x}) = \mathbf{W}^{(\ell)}\boldsymbol{h}^{(\ell-1)}(\boldsymbol{x}) + \mathbf{b}^{(\ell)}, \quad 2 \leq \ell \leq L, \tag{3}$$

where $\mathbf{W}^{(1)} \in \mathbb{R}^{m \times n}$, $\mathbf{W}^{(\ell)} \in \mathbb{R}^{n \times n}$, and $\mathbf{b}^{(\ell)} \in \mathbb{R}^n$. $\phi(\cdot) :$ $\mathbb{R} \to \mathbb{R}$ represents the activation function that acts coordinate wise on a vector.

The output of an MLP is obtained by a linear transformation of the last layer

$$f_\theta^{(L)}(\boldsymbol{x}) = \mathbf{V}^{(L)}\boldsymbol{h}^{(L)}(\boldsymbol{x}) \in \mathbb{R}^d, \tag{4}$$

where θ is the set of all parameters in the networks. Let $\mathcal{X}$ be the set of training inputs. Let and $\mathcal{Y} \in \mathbb{R}^{|\mathcal{X}|d}$ be the concatenation of all labels. Let $l(\widehat{y}, y) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$ be the MSE loss function, and $\mathcal{L}(\theta) = \frac{1}{|\mathcal{X}|} \sum_{(\boldsymbol{x},\boldsymbol{y}) \in \mathcal{X} \times \mathcal{Y}} l(f_\theta(\boldsymbol{x}), \boldsymbol{y})$ be the the empirical loss. Let θ_0 be the networks parameters at initialization. An MLP trained with infinitesimal step size for Gaussian distributed θ_0 using the following loss

$$\min_\theta \left(\mathcal{L}(\theta) + \lambda \|\theta - \theta_0\|_2^2 \right) \tag{5}$$

is equivalent to a kernel ridge regression predictor with ridge hyperparameter λ and a deterministic kernel known as its neural tangent kernel (NTK). The NTK is defined as

$$\Theta(\boldsymbol{x}, \boldsymbol{x}') = \Theta^{(L)}(\boldsymbol{x}, \boldsymbol{x}')\boldsymbol{I}_d = \nabla_\theta f_\theta^{(L)}(\boldsymbol{x})^\top \nabla_\theta f_\theta^{(L)}(\boldsymbol{x}') \tag{6}$$

where $\Theta^{(L)}(\boldsymbol{x}, \boldsymbol{x}') \in \mathbb{R}$ is the NTK of a MLP with depth L that depends on the network's activation function, depth and initialization hyperparameters [15]. We define $\Theta^{(L)}(\boldsymbol{x}, \mathcal{X})$ as the vector in $\mathbb{R}^{|\mathcal{X}|}$ whose entries are $\Theta^{(L)}(\boldsymbol{x}, \boldsymbol{x}')$ for $\boldsymbol{x}' \in \mathcal{X}$; similarly, $\Theta^{(L)}(\mathcal{X}, \mathcal{X}') \in \mathbb{R}^{|\mathcal{X}| \times |\mathcal{X}'|}$.
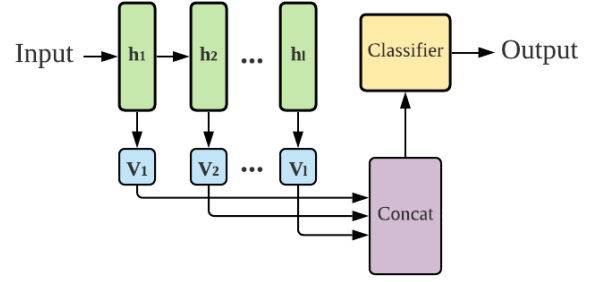

Fig. 1: Diagram of the NFT-K MLP network. Each inner product of the kernel with the layer is sent as the input to the classifier. The weights of the classifier can interpret which layers are aiding in the decision-making process.

3. NON-FUNGIBLE TANGENT KERNELS

In this section, we propose our NFT-K architecture. Consider the recursive relations from Equations (1), (2), and (3) that form the hidden states $\boldsymbol{h}^{(\ell)}(\boldsymbol{x})$. Instead of obtaining the output by a linear transformation from the last layer as shown in Equation (4), we obtain the output $f_\theta(\boldsymbol{x})$:

$$f_\theta^{(\ell)}(\boldsymbol{x}) = \mathbf{V}^{(\ell)}\boldsymbol{h}^{(\ell)}(\boldsymbol{x}), \tag{7}$$

$$f_\theta(\boldsymbol{x}) = \sum_{\ell=1}^L \omega_\ell \odot f_\theta^{(\ell)}(\boldsymbol{x}) \quad \omega_\ell \in \mathbb{R}^d, \tag{8}$$

where $\odot$ is the entry-wise product. Note that $\mathbf{V}^{(\ell)}$ for $\ell = 1, \ldots, L-1$ are not present in the MLP architecture. Figure 1 displays a network diagram for visualizing of this architecture.

We are interested in the following optimization problem

$$\min_\Omega \min_\theta \left(\mathcal{L}(\theta, \Omega) + \lambda \|\theta - \theta_0\|_2^2 \right) \quad s.t \quad \Omega \in \mathcal{S} \tag{9}$$

where

$$\theta = \text{Vect}[\{\mathbf{W}^{(\ell)}, \mathbf{V}^{(\ell)}, \mathbf{b}^{(\ell)}\}_{\ell=1}^L], \tag{10}$$

$$\Omega = [\omega_\ell, \ldots, \omega_L] \in \mathbb{R}^{d \times L}, \tag{11}$$

$$\mathcal{S} = \left\{ \Omega \in \mathbb{R}^{d \times L} | \Omega > 0 \right\}. \tag{12}$$

To solve (9), we use alternating minimization by first solving over θ for a fixed Ω and then solving over Ω for a fixed θ. More precisely, we perform gradient descent on θ for fixed Ω:

$$\theta_{t+1} = \theta_t - \eta_\theta \nabla_{\theta_t} \mathcal{L}(\theta_t, \Omega) - \eta_\theta \lambda(\theta_t - \theta_0) \tag{13}$$

until convergence. In practice, we use stochastic gradient descent with mini-batches. The following theorem characterizes the output of the network after each update performed in Equation 13.

Table 1: Test accuracy among different datasets showing that MLPs and NFT-K MLP perform similarly. In the MNIST classification task, the 10 layer NFT-K MLP outperforms all other networks. In the large movie review classification problem, MLPs and NFT-K MLP perform within 0.42% of one another.

Network	Dataset	Test Acc
10 layer MLP	MNIST	96.89%
10 layer NFT-K MLP	MNIST	**97.25**%
20 layer MLP	MNIST	97.22%
20 layer NFT-K MLP	MNIST	95.23%
5 layer MLP	Large Movie Review	**86.73**%
5 layer NFT-K MLP	Large Movie Review	86.4%
10 layer MLP	Large Movie Review	86.31%
10 layer NFT-K	Large Movie Review	86.7%

Theorem 1 *After convergence of gradient descent in (13) over θ for a fixed Ω with infinitesimal step size $\eta_\theta \to 0$, the output of the neural network will be*

$$f(\boldsymbol{x}) = \Theta_\Omega(\boldsymbol{x}, \mathcal{X}) \left(\Theta_\Omega(\mathcal{X}, \mathcal{X}) + \lambda \boldsymbol{I}_{d|\mathcal{X}|}\right) \mathcal{Y}, \quad (14)$$

where

$$\Theta_\Omega(\boldsymbol{x}, \boldsymbol{x}') = \sum_{\ell=1}^{L} \Theta^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}')\mathrm{diag}(\omega_\ell) \quad (15)$$

and $\Theta^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}')$ is the NTK of an MLP with depth ℓ as defined in Equations 1 to 6.

Proof of Theorem 1. For a fixed Ω, the output of the network is just a linear combination of the output at each layers. Since we are using gradient descent with infinitesimal step size the out put will be kernel ridge regression with respect to the NTK of network and the NTK can be easily obtained from the results of [23]. $\square$

In particular this result highlights that in the infinite width limit, solving over θ has a closed form solution that reduces to kernel ridge regression. The corresponding kernel is a linear combination of the NTKs of each layer of the network, as shown in Equation 15.

In the second step of our alternating minimization procedure, optimizing over $\Omega \in \mathcal{S}$ is a very well known problem known as multiple kernel learning [25]. Hence, optimizing the network loss in 9 using alternating minimization results in multiple-kernel learning in the infinite-width limit and the correspondingly learned ω_ℓ informs us of each layer's importance in the given classification task.

To solve over Ω, we use gradient projection:

$$\Omega_{t+1} = \mathcal{P}_\mathcal{S}\left(\Omega_t - \eta_\Omega \nabla_{\Omega_t}\mathcal{L}(\theta, \Omega)\right). \quad (16)$$

4. RESULTS

4.1. Datasets

The datasets used in the experiments are MNIST [26] and Large Movie Review [27]. MNIST is composed of the handwritten digits from 0 to 9 and the task is to classify the 10 digits from the image. Large Movie Review dataset consists of movie reviews from IMDB and the task is to determine if the review's sentiment is positive or negative. For data preprocessing of this dataset, we use the encoder from the smallest GPT-2 model [28] to extract word token embeddings. Then for each movie review, we normalize the embeddings based on the its length, and the resulting input embedding has a dimension of 768.

4.2. Hyperparameter Selection

For MNIST in the NFT-K scenario we use 2 Stochastic Gradient Descent (SGD) [29] optimizers where one optimizer is for all the network parameters except the Ω's while the second optimizer is training the Ω. we set $\eta_\theta = 0.02$, while $\eta_\Omega = 0.05$, both with a weight decay of 2×10^{-6}. The number of neurons per layer are 256. The depths of the MLPs are 10 and 20. The mini-batch size is 50. The initialization strategy is Xavier normal initializer [30]. For Ω, they are projected to be non-negative.

For Large Movie Review, we use one SGD optimizer for all the network parameters. The learning rates is 0.1. The number of neurons per layer are 128. The depths of the MLPs are 5 and 10. The mini-batch size is 25. The experiment also uses Xavier normal initializer. For Ω, they are projected to be non-negative.

4.3. MNIST Experiment

We compare an MLP network with an NFT-K MLP network with two different depths, as shown in Table 1. We show that the 10 layer NFT-K MLP outperforms all other networks that we trained on. Most of these networks performed similarly, even though the architectures are quite different. Therefore, in this experiment, NFT-K MLPs provide a way to increase interpretability without a cost to performance.

Our experiments show that, although classification was on par with the baseline, only one layer contributed to the classification of digit 2 in this experiment. We see from the top right panel of Figure 2 that the row for digit 2 has only a single non-zero entry. The non-zero entry is located at the first layer of the network, meaning that the proceeding layers do not contribute to the classification of 2 whatsoever. In these plots, we calculate $\Omega = \begin{bmatrix} \omega_1 & \dots & \omega_L \end{bmatrix}$ and normalize its rows to get Ω_n because we want to compare how each class is being affected by each layer. We see that the sparsity of Ω_n is a useful property for determining which layers are important for classifying certain classes.
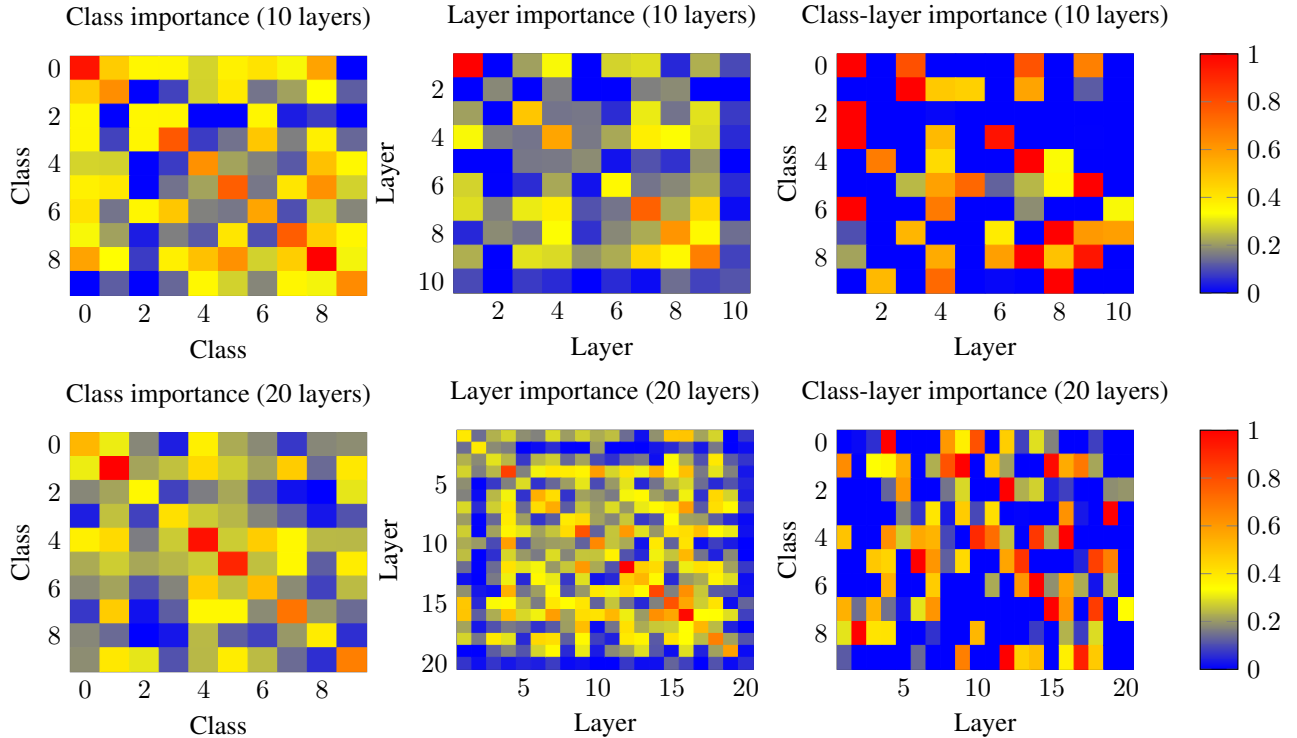
Fig. 2: We present heatmaps which can be used to interpret the class importance (ω), layer importance, and class-layer importance. The heatmaps in the **top row** are constructed from a trained NFT-K MLP with 10 layers, while the heatmaps in the **bottom row** are constructed from a NFT-K MLP with 20 layers. We first define $\Omega = \begin{bmatrix} \omega_1 & \cdots & \omega_L \end{bmatrix}$ and then normalize it by row to get Ω_n. In this figure, the first column is $\Omega_n^\top \Omega_n$, the second column is $\Omega_n \Omega_n^\top$, and the third column is Ω_n. For the 10 layer case, 53.0% of the elements of Ω are zero. For the 20 layer case, 45.5% of the elements of Ω_n are zero.

We can also see from the top right panel of Figure 2 that the the rows for the digit 4 and 9 are similar. There's only one significant difference: layer 7 is critical for distinguishing between the two digits as it only appears when classifying the 4 digits. Moreover, we see that both layer 7 and layer 1 are crucial for classification (center top panel of Figure 2). It is obvious why layer 1 is important: it is the only layer that can classify the 2 digits and it plays a role in classification of 6 out of the 10 digits. Thus, we can use these class-layer, layer, and class importance heatmaps to interpret how our layers are classifying inputs.

4.4. Large Movie Review Experiment

We compare the MLP network with a NFT-K MLP network with two different depths shown in Table 1. From Table 1, the NFT-K MLP networks perform even more closely to the MLP networks, varying by at most 0.42%. Thus we can use NFT-K MLPs to perform similarly to MLPs while gaining interpretability.

The sparsity of Ω_n in this experiment is close to that of MNIST: 50% and 45% of the elements were zero for the 5 layer and 10 layer, respectively. What is even more interesting

is that the weights for two layers where completely zeroed out and not used in the NFT-K MLPs. This sparsity of the weights and of the columns of Ω shows how one can achieve almost the same performance without using a significant part of the network, similar to dropout. [1]

5. DISCUSSION

Inspired from the NTK theory, we designed a new network that allows connections from all the layers. From the interpretable weights from the NFT-K network classifier we were able to quantify which layers contributed based on Figure 2. With these weights, they provide insight to see if it is the early layers contributing the most or the later layers contributing the most. There are many avenues to pursue for future work such as investigating how increasing the depth or the number of classes will affect the prioritization of layers for the classifier. Another question that arises is whether we can apply other sparsity constraints to investigate if more sparse solutions exist with fewer layers contributing to the classification.

---

[1]Our code can be found at https://gitlab.com/cjbarb7/icassp2022.

6. REFERENCES

[1] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," *arXiv preprint arXiv:1809.11096*, 2018.

[2] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8110–8119, 2020.

[3] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[6] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[7] A. Uçar, Y. Demir, and C. Güzeliş, "Object recognition and detection with deep learning for autonomous driving applications," *Simulation*, vol. 93, no. 9, pp. 759–769, 2017.

[8] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[9] B. Shickel, P. J. Tighe, A. Bihorac, and P. Rashidi, "Deep EHR: a survey of recent advances in deep learning techniques for electronic health record (EHR) analysis," *IEEE journal of biomedical and health informatics*, vol. 22, no. 5, pp. 1589–1604, 2017.

[10] P. Dixit and S. Silakari, "Deep learning algorithms for cybersecurity applications: A technological and status review," *Computer Science Review*, vol. 39, p. 100317, 2021.

[11] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning," in *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pp. 80–89, IEEE, 2018.

[12] J. H. Friedman, *The elements of statistical learning: Data mining, inference, and prediction*. springer open, 2017.

[13] M. Belkin, D. Hsu, S. Ma, and S. Mandal, "Reconciling modern machine-learning practice and the classical bias-variance trade-off," *Proceedings of the National Academy of Sciences*, vol. 116, no. 32, pp. 15849–15854, 2019.

[14] M. Belkin, S. Ma, and S. Mandal, "To understand deep learning we need to understand kernel learning," in *International Conference on Machine Learning*, pp. 541–549, PMLR, 2018.

[15] A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," in *Advances in Neural Information Processing Systems*, pp. 8571–8580, 2018.

[16] S. Arora, S. S. Du, W. Hu, Z. Li, R. R. Salakhutdinov, and R. Wang, "On exact computation with an infinitely wide neural net," in *Advances in Neural Information Processing Systems*, pp. 8141–8150, 2019.

[17] R. Novak, L. Xiao, Y. Bahri, J. Lee, G. Yang, D. A. Abolafia, J. Pennington, and J. Sohl-dickstein, "Bayesian deep convolutional networks with many channels are gaussian processes," in *International Conference on Learning Representations*, 2019.

[18] S. S. Du, K. Hou, R. R. Salakhutdinov, B. Poczos, R. Wang, and K. Xu, "Graph neural tangent kernel: Fusing graph neural networks with graph kernels," in *Advances in Neural Information Processing Systems*, pp. 5724–5734, 2019.

[19] K. Huang, Y. Wang, M. Tao, and T. Zhao, "Why do deep residual networks generalize better than deep feedforward networks?—a neural tangent kernel perspective," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[20] J. Hron, Y. Bahri, J. Sohl-Dickstein, and R. Novak, "Infinite attention: NNGP and NTK for deep attention networks," in *International Conference on Machine Learning*, pp. 4376–4386, PMLR, 2020.

[21] S. Alemohammad, Z. Wang, R. Balestriero, and R. Baraniuk, "The recurrent neural tangent kernel," in *International Conference on Learning Representations*, 2021.

[22] S. Alemohammad, R. Balestriero, Z. Wang, and R. Baraniuk, "Scalable neural tangent kernel of recurrent architectures," *arXiv preprint arXiv:2012.04859*, 2020.

[23] G. Yang, "Tensor programs II: Neural tangent kernel for any architecture," *arXiv preprint arXiv:2006.14548*, 2020.

[24] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125, 2017.

[25] L. Zhang, D. Romero, and G. B. Giannakis, "Fast convergent algorithms for multi-kernel regression," in *2016 IEEE Statistical Signal Processing Workshop (SSP)*, pp. 1–4, IEEE, 2016.

[26] L. Deng, "The MNIST database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[27] A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pp. 142–150, 2011.

[28] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[29] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng, "On optimization methods for deep learning," in *ICML*, 2011.

[30] S. K. Kumar, "On weight initialization in deep neural networks," *arXiv preprint arXiv:1704.08863*, 2017.