Exact Learning of Permutations for Nonzero Binary Inputs with Logarithmic Training Size and Quadratic Ensemble Complexity

George Giapitzakis GGIAPITZ@UWATERLOO.CA
University of Waterloo, Waterloo, ON, Canada

Artur Back de Luca ABACKDEL@UWATERLOO.CA
University of Waterloo, Waterloo, ON, Canada

Kimon Fountoulakis KIMON.FOUNTOULAKIS@UWATERLOO.CA
University of Waterloo, Waterloo, ON, Canada

Abstract

The ability of an architecture to realize permutations is quite fundamental. For example, Large Language Models need to be able to correctly copy (and perhaps rearrange) parts of the input prompt into the output. Classical universal approximation theorems guarantee the existence of parameter configurations that solve this task but offer no insights into whether gradient-based algorithms can find them. In this paper, we address this gap by focusing on two-layer fully connected feed-forward neural networks and the task of learning permutations on nonzero binary inputs. We show that in the infinite width Neural Tangent Kernel (NTK) regime, an ensemble of such networks independently trained with gradient descent on only the k standard basis vectors out of $2^k - 1$ possible inputs successfully learns *any* fixed permutation of length k with arbitrarily high probability over the weights initialization. By analyzing the exact training dynamics, we prove that the network's output converges to a Gaussian process whose mean captures the ground truth permutation via sign-based features. We then demonstrate how averaging these runs (an "ensemble" method) and applying a simple rounding step yields an arbitrarily accurate prediction on any possible input unseen during training. Notably, the number of models needed to achieve exact learning with high probability (which we refer to as *ensemble complexity*) exhibits a linearithmic dependence on the input size k for a single test input and a quadratic dependence when considering all test inputs simultaneously.

1. Introduction

Neural networks have long been known as universal function approximators: a two-layer feed-forward neural network with sufficient width can approximate any continuous function on a compact domain [10, 14]. However, these classical theorems only guarantee the *existence* of a suitable parameter configuration; they do not explain whether a standard training procedure (e.g., gradient-based training) can efficiently discover such parameters. While universal approximation highlights the expressive power of neural networks, it does not directly address how trained networks generalize beyond the specific data they observe.

In contrast, the literature on domain generalization provides general learning guarantees from a Probably Approximately Correct (PAC) learning perspective. Given training and testing distributions, one can bound the expected test loss of *any* hypothesis that minimizes the empirical training risk. This bound typically depends on the discrepancy between the training and testing distributions, the training sample size, and the complexity of the hypothesis class [4, 18]. However, such bounds do not

© G. Giapitzakis, A.B.d. Luca & K. Fountoulakis.

account for training dynamics, making them quite pessimistic, especially when a single hypothesis that fails to perform well on the testing data is a minimizer of the training loss, even if a standard training procedure would never find it.

Our work complements these approaches in a more specialized setting. We focus on exact learning using the *training dynamics* of a simple two-layer network in the *infinite-width* Neural Tangent Kernel (NTK) regime. We show that an ensemble of such models can learn exactly, with arbitrarily high probability over the weights initialization, a global transformation, in our case, a *permutation* function, from a logarithmically small training set.

Contributions. We use the NTK framework to analyze the problem of learning permutations, a task many models need to perform as a subroutine, yet it is unknown if it can be performed exactly with high probability. Note that we work with a fixed, deterministic training set, unlike the usual setting, which assumes data are drawn from an underlying distribution. Likewise, our learning guarantee holds with high probability over the randomness of weight initialization for any test input. We list our contributions below:

- We show that when the inputs are nonzero, normalized, and binary,[1] a two-layer fully connected feed-forward network in the infinite width limit trained on the *standard basis* using gradient descent (or gradient flow) converges to a Gaussian Process whose mean captures the ground truth permutation via sign-based features. More precisely, each entry in the predicted mean is non-positive exactly when (and only when) the corresponding ground-truth value is zero.
- We show that this can be achieved with logarithmic training size by picking the training set to consist of the k standard basis vectors (out of the $2^k - 1$ vectors in the domain), where k is the input length.
- We can achieve zero error on any input unseen during training, with high probability, by averaging the output of $\mathcal{O}(k \log k)$ independently trained models and applying a post-processing rounding step. To guarantee zero error *simultaneously* for all inputs, the same procedure requires averaging the output of $\mathcal{O}(k^2)$ independently trained models.

2. Literature Review

The NTK framework [16] is a popular framework designed to provide insights into the continuous-time gradient descent (gradient flow) dynamics of fully connected feed-forward neural networks with widths tending to infinity. The initial results have been extended to discrete gradient descent [17] as well as generalized to more architecture families like RNNs and Transformers [26, 27]. While the general NTK theory has been extensively studied to provide insights on the in-distribution generalization behavior of different architectures [5, 6, 8, 9, 28], very few works use the framework to study the out-of-distribution performance of these architectures, even on specific tasks. Notably, the framework has been used to attempt to explain the "double descent" phenomenon [19] observed in deep architectures [1, 23]. It has also been used to study the behavior of Residual Networks [30] and Physics-Informed Neural Networks [24]. To the best of our knowledge, the closest work to ours is [7], where the authors use results from the general NTK theory to prove that Transformers can

1. Our results can be extended to different vocabularies by considering, for example, one-hot encodings of the alphabet symbols. The key assumption is that all symbols are individually "seen" by the model during training.

provably generalize out-of-distribution on template matching tasks. Furthermore, they show that feed-forward neural networks fail to generalize to unseen symbols for any template task, including copying, a special case of a permutation learning setting. In our paper, we show that feed-forward neural networks can indeed learn to copy without error with high probability. At first, this might seem to contradict [7]. However, the key difference between our work and the work in [7] lies in the problem setting. The setting in [7] poses no restrictions on the input, whereas we require binary-encoded input. Therefore, the model has access to all symbols during training. In particular, we specifically choose the training set, i.e., the standard basis, so that each symbol appears in every position. The restrictions above are what enable exact permutation learning.

Lastly, various studies highlight the expressive power of neural networks through simulation results. For instance, Siegelmann and Sontag [22] demonstrates the Turing completeness of recurrent neural networks (RNNs), while Hertrich and Skutella [13] introduces specific RNN constructions capable of solving the shortest paths problem and approximating solutions to the knapsack problem. Moreover, other simulation studies on Transformers have established their Turing completeness [20, 25] and showcased constructive solutions for linear algebra, graph-related problems [2, 11, 29], and parallel computation [3]. However, the results in these works are existential, and none consider the dynamics of the training procedure.

3. Notation and Preliminaries

Throughout the text, we use boldface to denote vectors. We reserve the notation $\boldsymbol{e_1}, \boldsymbol{e_2}, \dots$ for the standard basis vectors. We use $\mathbf{1}$ to denote the vector of all ones. For a vector $\boldsymbol{x} \in \mathbb{R}^n$ we denote by $\|\boldsymbol{x}\| := \sqrt{\sum_{i=1}^n \boldsymbol{x}_i^2}$ the Euclidean norm of $\boldsymbol{x}$. We denote the $n \times n$ identity matrix by I_n. For $n \in \mathbb{N} := \{1, 2, \dots\}$ let $H_n = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{x}_i = 0 \text{ or } \boldsymbol{x}_i = 1 \; \forall\, i = 1, 2, \dots, n\}$ denote the n-dimensional Hamming cube (the set of all n-dimensional binary vectors). We use the notation $[n]$ to refer to the set $\{1, 2, \dots, n\}$.

Model and NTK Results. Here we provide a brief overview of the necessary theory used to derive our results. We refer the reader to [12] for a comprehensive treatment of the NTK theory. We work with two-layer, ReLU-activated fully connected feed-forward neural networks with the same input and output dimension and no bias. Concretely, the architecture is defined as the function $F : \mathbb{R}^k \to \mathbb{R}^k$ with $F(\boldsymbol{x}) = W^2 \operatorname{ReLU}(W^1 \boldsymbol{x})$ where $W^1 \in \mathbb{R}^{n_h \times k}$, $W_2 \in \mathbb{R}^{n_h \times k}$, and $n_h \in \mathbb{N}$ is the hidden dimension. The weights are initialized according to the NTK parametrization as $W^1_{ij} = \frac{\sigma_\omega}{\sqrt{k}} \omega^1_{ij}$ and $W^2_{ij} = \frac{\sigma_\omega}{\sqrt{n_h}} \omega^2_{ij}$ where ω^1_{ij} and ω^2_{ij} are trainable parameters initialized i.i.d. from a standard Gaussian distribution. When $n_h \to \infty$, the empirical NTK kernel given by $\nabla_{\{W^1, W^2\}} F(\boldsymbol{x})^\top \nabla_{\{W^1, W^2\}} F(\boldsymbol{x'})$ converges to the deterministic limit:

$$\Theta(\boldsymbol{x}, \boldsymbol{x'}) = \left(\frac{\boldsymbol{x}^\top \boldsymbol{x'}}{2\pi k} (\pi - \theta) + \frac{\|\boldsymbol{x}\| \cdot \|\boldsymbol{x'}\|}{2\pi k} \left((\pi - \theta) \cos\theta + \sin\theta \right) \right) I_k \in \mathbb{R}^{k \times k} \tag{1}$$

and the NNGP kernel is given by

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x'}) = \left(\frac{\|\boldsymbol{x}\| \cdot \|\boldsymbol{x'}\|}{2\pi k} \left((\pi - \theta) \cos\theta + \sin\theta \right) \right) I_k \in \mathbb{R}^{k \times k} \tag{2}$$

where $\theta = \arccos\left(\boldsymbol{x}^\top \boldsymbol{x'} / \|\boldsymbol{x}\| \cdot \|\boldsymbol{x'}\| \right)$. For a set of vectors $\mathcal{X}$, we will use the notation $\Theta(\mathcal{X}, \cdot)$, $\Theta(\cdot, \mathcal{X})$ and $\Theta(\mathcal{X}, \mathcal{X})$ to refer to the limit NTK calculated when the the set $\{F(\boldsymbol{x}) : \boldsymbol{x} \in \mathcal{X}\}$ is vectorized

(the outputs are stacked vertically), and similarly for the NNGP kernel. Our learnability results rely on the following theorem by [17], adapted to our architecture:

Theorem 1 (Theorem 2.2 from Lee et al. 17) *Let $\mathcal{X}$ and $\mathcal{Y}$ be the training dataset (training inputs and ground truth labels, respectively). Assume that $\Theta := \Theta(\mathcal{X}, \mathcal{X})$ is positive definite. Suppose the network is trained with gradient descent (with a small enough step size) or gradient flow to minimize the empirical MSE loss.*[2] *Then, for every $\hat{\boldsymbol{x}} \in \mathbb{R}^k$ with $\|\hat{\boldsymbol{x}}\| \leq 1$, as $n_h \to \infty$, the output at training time t, $F_t(\hat{\boldsymbol{x}})$, converges in distribution to a Gaussian with mean and variance given by*[3]

$$\mu(\hat{\boldsymbol{x}}) = \Theta(\hat{\boldsymbol{x}}, \mathcal{X})\Theta^{-1}\mathcal{Y} \tag{3}$$

$$\Sigma(\hat{\boldsymbol{x}}) = \mathcal{K}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}) + \Theta(\hat{\boldsymbol{x}}, \mathcal{X})\Theta^{-1}\mathcal{K}(\mathcal{X}, \mathcal{X})\Theta^{-1}\Theta(\mathcal{X}, \hat{\boldsymbol{x}}) - (\Theta(\hat{\boldsymbol{x}}, \mathcal{X})\Theta^{-1}\mathcal{K}(\mathcal{X}, \hat{\boldsymbol{x}}) + h.c) \tag{4}$$

where $\mathcal{Y}$ in Equation (3) denotes the vectorization of all vectors $\boldsymbol{y} \in \mathcal{Y}$.

4. Summary of results

In this section, we first formalize the task and then proceed with a summary of our key results. Suppose $k \in \mathbb{N}$ and let $\mathbb{K} = \{\boldsymbol{x}/\|\boldsymbol{x}\| : \boldsymbol{x} \in H_k \setminus \{\boldsymbol{0}\}\}$.[4] Let $P^* \in \{0,1\}^{k\times k}$ be any fixed permutation matrix. We are interested in the ability of shallow neural networks to learn permutations exactly when trained on the set of standard basis vectors for the task of learning P^*. Concretely, the training dataset consists of training inputs $\mathcal{X} = \{\boldsymbol{e_1}, \ldots, \boldsymbol{e_k}\}$ and their corresponding ground-truth labels $\mathcal{Y} = \{P^*\boldsymbol{e_1}, \ldots, P^*\boldsymbol{e_k}\}$.

4.1. Main results

Here we present and discuss our two main results: Theorem 2 and Theorem 3. The former is proved in Appendix B and the latter in Appendix C.

Theorem 2 (The mean, at the limit, carries useful information) *For any test point $\hat{\boldsymbol{x}} \in \mathbb{K}$ with $n_{\hat{\boldsymbol{x}}}$ non-zero entries, the mean of the limiting distribution of Theorem 1 when the training dataset consists of the standard basis vectors carries sign-based information about the ground-truth permutation. Namely, there exist $\mu_0 \equiv \mu_0(n_{\hat{\boldsymbol{x}}}) \leq 0$ and $\mu_1 \equiv \mu_1(n_{\hat{\boldsymbol{x}}}) > 0$ that depend on $n_{\hat{\boldsymbol{x}}}$, such that for each $i \in [k]$, $\mu(\hat{\boldsymbol{x}})_i = \mu_0$ if $(P^*\hat{\boldsymbol{x}})_i = 0$ and $\mu(\hat{\boldsymbol{x}})_i = \mu_1$ otherwise (in which case $(P^*\hat{\boldsymbol{x}})_i = 1/\sqrt{n_{\hat{\boldsymbol{x}}}}$).*

Theorem 2 establishes that the NTK regime generates a distribution of models whose mean encapsulates the information required to solve the permutation problem. Thus, by averaging the network's output over enough independent runs on the same input $\hat{\boldsymbol{x}}$, we can approximate the actual mean $\mu(\hat{\boldsymbol{x}})$. This gives rise to a natural inference strategy: repeat the training procedure enough times, average the outputs of the network on input $\hat{\boldsymbol{x}}$ and then round each coordinate to 0 or $1/\sqrt{n_{\hat{\boldsymbol{x}}}}$ depending on whether the result of the averaging is positive or not. This averaging procedure identifies the target permutation P^* using a logarithmic training sample size (only k out of the $2^k - 1$ total possible inputs are needed) provided enough independent trials are averaged. We define *ensemble complexity* as the number of independent trials needed to obtain any desired level of post-rounding. By analyzing the

2. The empirical MSE loss is defined as $\mathcal{L}(\mathcal{D}) = (2|\mathcal{D}|)^{-1}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}} \|f_t(\boldsymbol{x}) - \boldsymbol{y}\|^2$.
3. Here "$h.c.$" is an abbreviation for the Hermitian conjugate.
4. We chose to exclude the zero vector because it unnecessarily complicates the calculations.

asymptotic orders of the mean and variance of the limiting distribution, we can prove the following theorem:

Theorem 3 (Ensemble complexity of permutation learning) *Let $\delta \in (0,1)$ and let $\hat{\boldsymbol{x}} \in \mathbb{K}$ be a test input. Under the assumptions of Theorem 2, when averaging over $\mathcal{O}(k \log(k/\delta))$ independent trials and rounding the average using the aforementioned rounding strategy, $\hat{\boldsymbol{x}}$ is correctly permuted (post-rounding) according to P^* with probability $1-\delta$.*

Theorem 3 gives an upper on the ensemble complexity for a *single* input. An application of the union bound shows that the conclusion of Theorem 3 holds simultaneously for all test inputs when averaging over $\mathcal{O}(k^2)$ models. Combining the last remark with Theorem 2 shows that two-layer feed-forward neural networks can learn *exactly with high probability* any fixed permutation on binary inputs of size k with logarithmic training size and quadratic ensemble complexity. In the next section, we provide experiments that demonstrate our theoretical results.

5. Experiments

We present two experiments that empirically demonstrate our theoretical results. We train a two-layer fully connected feed-forward network with a hidden layer of width 50,000 only using standard basis vectors.[5] Our goal is to learn some random permutation on normalized binary inputs of length k. For testing, we evaluate the performance on 1000 random vectors with $n_{\hat{\boldsymbol{x}}} = 2$ nonzero entries. To match the assumptions of our theoretical analysis, we initialize the weights exactly as given in Section 3.

1. Number of models to achieve 90% accuracy. Our first experiment examines how many independently trained models need to be averaged to achieve a testing accuracy of 90% as a function of the input length k.

2. Accuracy vs ensemble size. Our second experiment fixes $k \in \{5, 10, 15, 20, 25, 30\}$ and examines how the post-rounding accuracy varies as the ensemble size increases.

In Figure 1, the left plot presents the results of Experiment 1 compared to the theoretical ensemble complexity bound (expanded in Equation (20)) for the appropriate choice of δ to guarantee 90% accuracy for all test inputs simultaneously. We observe that the empirical ensemble size remains below the theoretical bound, and both curves follow a similar pattern as a function of k. For Experiment 2, the results on the right plot of Figure 1 illustrate the convergence to perfect accuracy as a function of the ensemble size for different input sizes k. As shown, larger input sizes require more models to achieve perfect accuracy, but with a sufficient number of models, all instances eventually reach perfect accuracy.

6. Conclusion and Future Work

We have demonstrated that, within the NTK framework, an ensemble of $\mathcal{O}(k^2)$ simple two-layer fully connected feed-forward neural networks can learn any fixed permutation on binary inputs of size k with arbitrarily high probability over the weights initialization from a training set of logarithmic size, namely, the standard basis vectors. In terms of future work, it would be interesting to extend

5. Our source code can be found at https://github.com/opallab/permlearn.

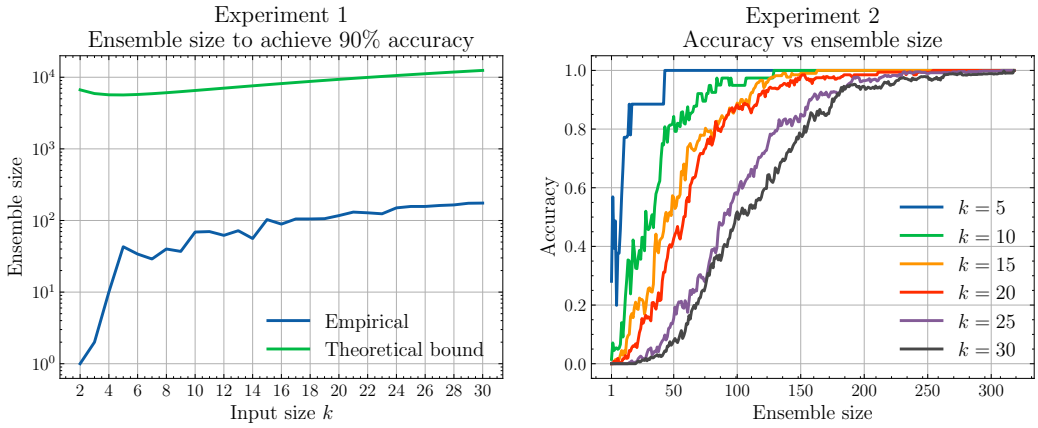

Figure 1: Experiment 1 (left) presents the required number of models (ensemble size) to achieve 90% accuracy as a function of input size k, compared to the theoretical bound from Equation (20) with the appropriate δ. Experiment 2 (right) presents accuracy as a function of ensemble size for different input sizes k.

our approach to a broader class of functions beyond just permutations and to a wider range of architectures like CNNs, RNNs, or Transformers. Finally, an exciting next step is developing a more general PAC-like framework for exact learning guarantees, specifically under the NTK regime. This would utilize the infinite-width limit and standard gradient-based training dynamics to give conditions under which neural architectures can learn global transformations with provable exactness.

Acknowledgements

K. Fountoulakis would like to acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC). Cette recherche a été financée par le Conseil de recherches en sciences naturelles et en génie du Canada (CRSNG), [RGPIN-2019-04067, DGECR-2019-00147].

G. Giapitzakis would like to acknowledge the support of the Onassis Foundation - Scholarship ID: F ZU 020-1/2024-2025.

References

[1] Ben Adlam and Jeffrey Pennington. The neural tangent kernel in high dimensions: Triple descent and a multi-scale theory of generalization. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 74–84. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/adlam20a.html.

[2] Artur Back De Luca and Kimon Fountoulakis. Simulation of graph algorithms with looped transformers. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 2319–2363. PMLR, 2024. URL https://dl.acm.org/doi/10.5555/3692070.3692162.

[3] Artur Back de Luca, George Giapitzakis, Shenghao Yang, Petar Veličković, and Kimon Fountoulakis. Positional attention: Expressivity and learnability of algorithmic computation, 2025. URL https://arxiv.org/abs/2410.01686.

[4] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006. URL https://proceedings.neurips.cc/paper_files/paper/2006/file/b1b0432ceafb0ce714426e9114852ac7-Paper.pdf.

[5] Alberto Bietti. Approximation and learning with deep convolutional models: a kernel perspective. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=lrocYB-0ST2.

[6] Alberto Bietti and Francis Bach. Deep equals shallow for reLU networks in kernel regimes. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=aDjoksTpXOP.

[7] Enric Boix-Adserà, Omid Saremi, Emmanuel Abbe, Samy Bengio, Etai Littwin, and Joshua M. Susskind. When can transformers reason with abstract symbols? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=STUGfUz8ob.

[8] Francesco Cagnetta, Alessandro Favero, and Matthieu Wyart. What can be learnt with wide convolutional neural networks? In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 3347–3379. PMLR, 2023. URL https://proceedings.mlr.press/v202/cagnetta23a.html.

[9] Abdulkadir Canatar, Blake Bordelon, and Cengiz Pehlevan. Spectral bias and task-model alignment explain generalization in kernel regression and infinitely wide neural networks. *Nature Communications*, 12(1):2914, 2021. doi: 10.1038/s41467-021-23103-1. URL https://doi.org/10.1038/s41467-021-23103-1.

[10] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989. doi: 10.1007/BF02551274. URL https://doi.org/10.1007/BF02551274.

[11] Angeliki Giannou, Shashank Rajput, Jy-Yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pages 11398–11442, 2023. URL https://dl.acm.org/doi/10.5555/3618408.3618866.

[12] Eugene Golikov, Eduard Pokonechnyy, and Vladimir Korviakov. Neural tangent kernel: A survey, 2022. URL https://arxiv.org/abs/2208.13614.

[13] Christoph Hertrich and Martin Skutella. Provably good solutions to the knapsack problem via neural networks of bounded size. *INFORMS journal on computing*, 35(5):1079–1097, 2023. URL https://pubsonline.informs.org/doi/10.1287/ijoc.2021.0225.

[14] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(89)90020-8. URL https://www.sciencedirect.com/science/article/pii/0893608089900208.

[15] Wolfram Research, Inc. Mathematica, Version 14.1. URL https://www.wolfram.com/mathematica. Champaign, IL, 2024.

[16] Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/5a4be1fa34e62bb8a6ec6b91d2462f5a-Paper.pdf.

[17] Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/0d1a9651497a38d8b1c3871c84528bd4-Paper.pdf.

[18] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. In *Proceedings of The 22nd Annual Conference on Learning Theory (COLT 2009)*, Montréal, Canada, 2009. URL http://www.cs.nyu.edu/~mohri/postscript/nadap.pdf.

[19] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=B1g5sA4twr.

[20] Jorge Pérez, Pablo Barceló, and Javier Marinkovic. Attention is turing-complete. *Journal of Machine Learning Research*, 22(75):1–35, 2021. URL https://dl.acm.org/doi/pdf/10.5555/3546258.3546333.

[21] Jack Sherman and Winifred J. Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 21(1):124–127, 1950. ISSN 00034851. URL http://www.jstor.org/stable/2236561.

[22] Hava Siegelmann and Eduardo Sontag. On the computational power of neural nets. *Journal of Computer and System Sciences*, 50:132–150, 1995. URL https://www.sciencedirect.com/science/article/pii/S0022000085710136.

[23] Sidak Pal Singh, Aurelien Lucchi, Thomas Hofmann, and Bernhard Schölkopf. Phenomenology of double descent in finite-width neural networks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=lTqGXfn9Tv.

[24] Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2021.110768. URL https://www.sciencedirect.com/science/article/pii/S002199912100663X.

[25] Colin Wei, Yining Chen, and Tengyu Ma. Statistically meaningful approximation: a case study on approximating turing machines with transformers. *Advances in Neural Information Processing Systems*, 35:12071–12083, 2022. URL https://dl.acm.org/doi/10.5555/3600270.3601147.

[26] Greg Yang. Tensor programs ii: Neural tangent kernel for any architecture, 2020. URL https://arxiv.org/abs/2006.14548.

[27] Greg Yang and Etai Littwin. Tensor programs iib: Architectural universality of neural tangent kernel training dynamics, 2021. URL https://arxiv.org/abs/2105.03703.

[28] Greg Yang and Hadi Salman. A fine-grained spectral perspective on neural networks, 2020. URL https://openreview.net/forum?id=HJlU-AVtvS.

[29] Liu Yang, Kangwook Lee, Robert D Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=HHbRxoDTxE.

[30] Li Yuqing, Luo Tao, and Kwan Yip, Nung. Towards an understanding of residual networks using neural tangent hierarchy (nth). *CSIAM Transactions on Applied Mathematics*, 3(4): 692–760, 2022. ISSN 2708-0579. doi: https://doi.org/10.4208/csiam-am.SO-2021-0053. URL https://global-sci.com/article/82332/towards-an-understanding-of-residual-networks-using-neural-tangent-hierarchy-nth.

Appendix A. Useful Results

In this section, we state and prove some results from linear algebra and probability theory that were used to derive our main results. For two matrices $A_1 \in \mathbb{R}^{n_1 \times m_1}$, $A_2 \in \mathbb{R}^{n_2 \times m_2}$, we denote by $A_1 \otimes A_2 \in \mathbb{R}^{n_1 n_2 \times m_1 m_2}$ their Kronecker product. We remark that $A_1 \otimes A_2$ is positive definite whenever A_1 and A_2 are positive definite. The following two results are used to derive the required NTK kernels:

Theorem 4 (Sherman and Morrison 21) *Suppose $A \in \mathbb{R}^{n \times n}$ is an invertible matrix and $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$. Then $A + \boldsymbol{u}\boldsymbol{v}^\top$ is invertible if and only if $1 + \boldsymbol{v}^\top A^{-1}\boldsymbol{u} \neq 0$. In this case,*

$$(A + \boldsymbol{u}\boldsymbol{v}^\top)^{-1} = A^{-1} - \frac{A^{-1}\boldsymbol{u}\boldsymbol{v}^\top A^{-1}}{1 + \boldsymbol{v}^\top A^{-1}\boldsymbol{u}}$$

Lemma 5 *Suppose $A \in \mathbb{R}^{n \times n}$, $\boldsymbol{a} = [a_1, \ldots, a_n]^\top \in \mathbb{R}^n$ and $\boldsymbol{y} \in \mathbb{R}^{n^2}$ is given by*

$$\boldsymbol{y} = \begin{bmatrix} A\boldsymbol{e_1} \\ A\boldsymbol{e_2} \\ \vdots \\ A\boldsymbol{e_n} \end{bmatrix}$$

Then $(\boldsymbol{a}^\top \otimes I_n)\boldsymbol{y} = A\boldsymbol{a}$.

Proof Both the right-hand side and left-hand side are vectors of size n. We shall show that for each $i = 1, 2, \ldots, n$, the i-th entry of the left-hand side is equal to the i-th entry of the right-hand side. The i-th entry of $A\boldsymbol{a}$ is simply given by

$$(A\boldsymbol{a})_i = \sum_{j=1}^{n} A_{ij} a_j$$

Now let $C = \boldsymbol{a}^\top \otimes I_n \in \mathbb{R}^{n \times n^2}$. We can easily observe that C has the following block structure:

$$C = \begin{bmatrix} a_1 I_n & a_2 I_n & \ldots & a_n I_n \end{bmatrix}$$

The i-th row of C is therefore given by

$$C_i = \begin{bmatrix} a_1 \boldsymbol{e_i}^\top & a_2 \boldsymbol{e_i}^\top & \ldots & a_n \boldsymbol{e_i}^\top \end{bmatrix} \in \mathbb{R}^{1 \times n^2}$$

Thus, the i-th entry of the left-hand side is given by

$$C_i \boldsymbol{y} = \sum_{j=1}^{n} a_j \boldsymbol{e_i}^\top A \boldsymbol{e_j} = \sum_{j=1}^{n} A_{ij} a_j = (A\boldsymbol{a})_i$$

since $\boldsymbol{e_i}^\top A \boldsymbol{e_j} = A_{ij}$. This concludes the proof.

■

The last result is a concentration bound for sums of Gaussian random variables, used to derive our ensemble complexity bounds:

Lemma 6 *Let $X_1, X_2, \ldots, X_n$ be independent Gaussian random variables with mean μ and variance σ^2 and let $\overline{X}_n = \sum_{i=1}^n X_i$. Then for all $t > 0$, we have:*

$$\mathbb{P}\left\{|\overline{X}_n - \mu| \geq t\right\} \leq 2\exp\left(-\frac{nt^2}{2\sigma^2}\right)$$

Proof We will use the Chernoff technique. Let $\lambda > 0$. We have $\overline{X}_n - \mu \sim \mathcal{N}\left(0, \frac{\sigma^2}{n}\right)$ and so by symmetry and Markov's inequality we get:

$$\mathbb{P}\left\{|\overline{X}_n - \mu| \geq t\right\} = 2\mathbb{P}\left\{\overline{X}_n - \mu \geq t\right\} = 2\mathbb{P}\left\{e^{\lambda(\overline{X}_n - \mu)} \geq e^{\lambda t}\right\} \leq e^{-\lambda t} \cdot \mathbb{E}\left[e^{\lambda(\overline{X}_n - \mu)}\right] \quad (5)$$

The expectation on the right-hand side is equal to the moment-generating function of a normal distribution with mean 0 and variance σ^2/n and so it is equal to $\exp\left(\frac{\sigma^2\lambda^2}{2n}\right)$. Now let

$$\phi(\lambda) = \exp\left(\frac{\sigma^2\lambda^2}{2n} - \lambda t\right)$$

be the right-hand side of Equation (5). Minimizing $\phi(\lambda)$ with respect to λ we find that the minimum occurs at $\lambda^* = \frac{nt}{\sigma^2}$ and plugging this back into Equation (5) gives the required bound. ■

Appendix B. Proof of Theorem 2

In this section, we provide the proof of Theorem 2 by analytically calculating the mean and variance as given by Theorem 1. Let $\hat{\boldsymbol{x}} \in \mathbb{K}$ be a test point. According to Theorem 1, the output of the network at training time t, $F_t(\hat{\boldsymbol{x}})$, in the infinite width limit converges to a multivariate Gaussian with mean and variance given by Equation (3) and Equation (4), respectively, provided that the matrix $\Theta(\mathcal{X}, \mathcal{X})$ is positive definite. To calculate $\mu(\hat{\boldsymbol{x}})$, we first need to derive $\Theta(\hat{\boldsymbol{x}}, \mathcal{X})$ and $\Theta(\mathcal{X}, \mathcal{X})$ according to Equation (1), which we refer to as the test and train NTK kernels, respectively. We divide the computation into sections and also provide an illustrative numerical example.

B.1. Train NTK

In this section, we derive the train NTK kernel $\Theta := \Theta(\mathcal{X}, \mathcal{X})$. Recall that the train inputs are simply the standard basis vectors $\boldsymbol{e_1}, \ldots, \boldsymbol{e_n}$ and so we have

$$\Theta := \Theta(\mathcal{X}, \mathcal{X}) = \left((d-c)I_k + c\mathbf{1}\mathbf{1}^\top\right) \otimes I_k \in \mathbb{R}^{k^2 \times k^2} \quad (6)$$

where

$$d = \frac{1}{k} \quad \text{and} \quad c = \frac{1}{2\pi k} \quad (7)$$

We can observe that since $d > c > 0$, Θ is positive definite. Indeed, since $\mathbf{1}\mathbf{1}^\top$ (the matrix of all ones) is positive semidefinite, $(d-c)I_k + c\mathbf{1}\mathbf{1}^\top$ is (strictly) positive definite and so the Kronecker

product with I_k is also positive definite (see Section 3). In that case, a direct application of Theorem 4 shows that

$$\Theta^{-1} = \left(\frac{1}{d-c} I_k - \frac{c}{(d-c)(d-c+ck)} \mathbf{1}\mathbf{1}^\top \right) \otimes I_k \tag{8}$$

Notice that the fact that the training set is an orthonormal set of vectors (the standard basis) the train NTK has a convenient structure, namely each block $\Theta(\boldsymbol{e_i}, \boldsymbol{e_j})$ is diagonal. This fact is captured in the Kronecker product structure of Equation (6).

B.2. Test NTK

In this section, we calculate the test NTK $\Theta(\hat{\boldsymbol{x}}, \mathcal{X})$ for an arbitrary test input $\hat{\boldsymbol{x}} \in \mathbb{K}$. Again, by Equation (1), we find that

$$\Theta(\hat{\boldsymbol{x}}, \mathcal{X}) = \mathbf{f}^\top \otimes I_k \in \mathbb{R}^{k \times k^2} \tag{9}$$

where for each $i = 1, 2, \ldots, k$:

$$\mathbf{f}_i = \frac{\cos \hat{\theta}_i (\pi - \hat{\theta}_i) + \sin \hat{\theta}_i}{2k\pi} + \hat{\boldsymbol{x}}_i \frac{\pi - \hat{\theta}_i}{2k\pi} \tag{10}$$

and

$$\hat{\theta}_i = \arccos(\hat{\boldsymbol{x}}_i) \tag{11}$$

Notice that since each $\hat{\boldsymbol{x}}_i$ is equal to either 0 or $1/\sqrt{n_{\hat{\boldsymbol{x}}}}$ where $n_{\hat{\boldsymbol{x}}}$ is the number of nonzero entries of $\hat{\boldsymbol{x}}$, the resulting $\hat{\theta}_i$ takes two values $\hat{\theta}^0$ and $\hat{\theta}^1$ (corresponding to $\hat{\boldsymbol{x}}_i = 0$ and $\hat{\boldsymbol{x}}_i = 1/\sqrt{n_{\hat{\boldsymbol{x}}}}$) and subsequently each $\mathbf{f}_i$ also takes two values f^0 and f^1. Substituting we get:

$$\hat{\theta}^0 = \frac{\pi}{2} \quad \text{and} \quad \hat{\theta}^1 = \arccos\left(\frac{1}{\sqrt{n_{\hat{\boldsymbol{x}}}}}\right) \tag{12}$$

The corresponding f^0 and f^1 are derived by substituting $\hat{\theta}^0$ and $\hat{\theta}^1$ from Equation (12) to Equation (10).

Numerical example. Let us consider a concrete numerical example. Take $k = 5$ and $P^* = I$ the identity permutation. Consider the test inputs vector $\hat{\boldsymbol{x}} = (1/\sqrt{2}, 1/\sqrt{2}, 0, 0, 0)^\top$. Calculating the values of f^0 and f^1 for $\hat{\boldsymbol{x}}$ we find

$$\mathrm{f}^0 \approx 0.0318 \quad \text{and} \quad \mathrm{f}^1 \approx 0.1285 \tag{13}$$

Then

$$\Theta(\hat{\boldsymbol{x}}, \boldsymbol{e_i}) \approx \begin{cases} 0.1285 & \text{if } i = 1, 2 \\ 0.0318 & \text{if } i = 3, 4, 5 \end{cases}$$

These numbers act as a measure of similarity between the test input and each element of the training set. In our example, the test input has nonzero entries in its first two coordinates, and hence the values of $\Theta(\hat{\boldsymbol{x}}, \boldsymbol{e_1})$ and $\Theta(\hat{\boldsymbol{x}}, \boldsymbol{e_2})$ are higher, reflecting a stronger "match" of the test input $\hat{\boldsymbol{x}}$ with the training vectors $\boldsymbol{e_1}$ and $\boldsymbol{e_2}$.

B.3. Computing $\mu(\hat{\boldsymbol{x}})$

Having calculated the required NTK kernels we are now ready to calculate the mean of the output distribution $\mu(\hat{\boldsymbol{x}})$ for a test input $\hat{\boldsymbol{x}} \in \mathbb{K}$. Since Θ is positive definite we can now calculate the mean of the limiting Gaussian. Recall that we have

$$\mu(\hat{\boldsymbol{x}}) = \Theta(\hat{\boldsymbol{x}}, \mathcal{X})\Theta^{-1}\mathcal{Y}$$

where

$$\mathcal{Y} = \begin{bmatrix} P^*\boldsymbol{e_1} \\ P^*\boldsymbol{e_2} \\ \vdots \\ P^*\boldsymbol{e_k} \end{bmatrix}$$

From Equation (9) and Equation (8) we get

$$\Theta(\hat{\boldsymbol{x}}, \mathcal{X})\Theta^{-1} = \mathbf{f}^\top \left(\frac{1}{d-c} I_k - \frac{c}{(d-c)(d-c+ck)} \mathbf{1}\mathbf{1}^\top \right) \otimes I_k \tag{14}$$

and from Theorem 5 we get

$$\mu(\hat{\boldsymbol{x}}) = P^* \left(\frac{1}{d-c} I_k - \frac{c}{(d-c)(d-c+ck)} \mathbf{1}\mathbf{1}^\top \right) \mathbf{f} \tag{15}$$

There is an interesting pattern arising from Equation (15): since $\mathbf{f}_i$ only takes two values depending on whether $\hat{\boldsymbol{x}}_i$ is nonzero, $\mu(\hat{\boldsymbol{x}})$ also takes only two values. Remarkably, the multiplication by P^* in Equation (15) corresponds exactly to the ground truth permutation we want our model to learn.

Numerical example. Coming back to our numerical example of the previous section, we can compute the mean exactly by calculating $\Theta^{-1}\mathbf{f}$, which gives

$$\mu(\hat{\boldsymbol{x}}) \approx (0.5606, 0.5606, -0.0146, -0.0146, -0.0146)^\top$$

Intuitively, multiplying $\mathbf{f}$ with the inverse of the train NTK matrix reweights the similarity scores to correct for correlations between the training set.[6] Finally, multiplying by P^* (in this case the identity) permutes the reweighted similarities according to the ground truth permutation. Notice that the mean is negative for entries where the ground truth is zero and positive otherwise. Below, we show that this is not a coincidence.

Interpreting the mean. Substituting everything into Equation (15) we find that

$$\mu(\hat{\boldsymbol{x}})_i = \frac{y^2 - \sqrt{y^2-1}\,y - 2\pi(y-1) + 2y\sec^{-1}(y) - 1}{(2\pi-1)(k+2\pi-1)} \tag{16}$$

if $(P^*\hat{\boldsymbol{x}})_i = 0$ and

$$\begin{aligned} \mu(\hat{\boldsymbol{x}})_i = &-\frac{\left(\sqrt{y^2-1}-y\right)\left(y^2-k\right)}{(2\pi-1)(k+2\pi-1)y} + \frac{\pi\left(k-y^2+2\sqrt{y^2-1}-1\right)}{(2\pi-1)(k+2\pi-1)y} \\ &+ \frac{2\left(k-y^2+2\pi-1\right)\csc^{-1}(y) - \sqrt{y^2-1} + 2\pi^2}{(2\pi-1)(k+2\pi-1)y} \end{aligned} \tag{17}$$

6. This is similar to how multiplication by $(X^\top X)^{-1}$ in linear regression accounts for the correlation between observations.

if $(P^*\hat{\boldsymbol{x}})_i > 0$, where $y = \sqrt{n_{\hat{\boldsymbol{x}}}}$. Using a symbolic calculation system (Mathematica Inc. [15]) we can establish that for all possible values of y (i.e $y = \sqrt{m}$ for $m = 1, 2, \ldots, k$), $\mu(\hat{\boldsymbol{x}})_i \leq 0$ *if and only if* $(P^*\hat{\boldsymbol{x}})_i = 0$. This concludes the proof of Theorem 2. Furthermore, the expressions for $\mu_0(\hat{\boldsymbol{x}})$ and $\mu_1(\hat{\boldsymbol{x}})$ are given by Equation (16) and Equation (17), respectively.

Appendix C. Proof of Theorem 3

The conclusion of Theorem 2 suggests a simple rounding algorithm in practice: for each coordinate, if the output of the network is greater than zero round to 1, otherwise round to 0. As discussed in Section 4, this motivates a practical implementation where the network is trained enough times, its outputs are averaged over all runs, and subsequently rounded accordingly. In the following sections, we provide the proof of Theorem 3. First, we use the concentration inequality of Theorem 6 to derive an estimate of the number of independent runs needed to achieve any desired level of post-rounding accuracy in terms of k, $\mu(\hat{\boldsymbol{x}})$ and $\sigma(\hat{\boldsymbol{x}})$. Subsequently, we compute the asymptotic orders of $\mu(\hat{\boldsymbol{x}})$ and $\sigma(\hat{\boldsymbol{x}})$ to arrive at the conclusion of Theorem 3.

C.1. Rounding procedure

In this section, we analyze the practical rounding strategy in more detail. Specifically, we try to quantify how many runs are enough. Certainly, as the number of runs tends to infinity we can guarantee that an average over all of them will yield perfect classification. However, this statement is not very useful from a practical standpoint as it doesn't show the dependence on the size of the input k.

Ensuring perfect accuracy Let $\hat{\boldsymbol{x}}$ be a test input and $\hat{\boldsymbol{y}} = P^*\hat{\boldsymbol{x}}$ be the corresponding ground truth output. Suppose we take an average of N models. Let $F^j(\hat{\boldsymbol{x}})$ be the output of the j-th model. The $F^j(\hat{\boldsymbol{x}})$'s are independent samples from a multivariate normal distribution with mean $\mu(\hat{\boldsymbol{x}})$ and variance $\Sigma(\hat{\boldsymbol{x}})$, as given by Theorem 1. The i-th coordinates of the network's output, $F^j(\hat{\boldsymbol{x}})_i$, are therefore independent samples from a normal distribution with mean $\mu_i := \mu(\hat{\boldsymbol{x}})_i$ and variance $\sigma_i^2 = \boldsymbol{e_i}^\top \Sigma(\hat{\boldsymbol{x}})\boldsymbol{e} = \Sigma(\hat{\boldsymbol{x}})_{ii}$. Let $G(\hat{\boldsymbol{x}}) = \frac{1}{N}\sum_{j=1}^{N} F^j(\hat{\boldsymbol{x}})$ be the average over all N model outputs. Given that $\mu_i \leq 0$ whenever $\hat{\boldsymbol{y}}_i = 0$ and $\mu_i > 0$ whenever $\hat{\boldsymbol{y}}_i > 0$ we can see that the rounding procedure will yield a correct result for the i-th coordinate if $|G(\hat{\boldsymbol{x}})_i - \mu_i| < |\mu_i|/2$. An application of the union bound and Theorem 6 shows that

$$\mathbb{P}\left\{|G(\hat{\boldsymbol{x}})_i - \mu_i| \geq |\mu_i|/2 \text{ for all } i \in \{1, 2, \ldots, k\}\right\} \leq 2k \max_{i=1,2,\ldots,k} \exp\left(-\frac{N\mu_i^2}{8\sigma_i^2}\right) \tag{18}$$

Setting the right-hand side of Equation (18) to be at most $\delta \in (0, 1)$ and noting that $\sigma_i = \sigma^2(\hat{\boldsymbol{x}})$[7] for all $i \in [k]$ and each μ_i can take the two values $\mu_0(\hat{\boldsymbol{x}})$ and $\mu_1(\hat{\boldsymbol{x}})$ we get that our rounding procedure produces the correct output for a test input $\hat{\boldsymbol{x}} \in \mathbb{K}$ with probability at least $1 - \delta$ whenever the number of models N satisfies

$$N \geq 8\sigma^2(\hat{\boldsymbol{x}}) \cdot \max\left\{\frac{1}{\mu_0(\hat{\boldsymbol{x}})^2}, \frac{1}{\mu_1(\hat{\boldsymbol{x}})^2}\right\} \ln\left(\frac{2k}{\delta}\right) \tag{19}$$

Similarly to how sample complexity is defined, we use the term *ensemble complexity* to refer to the number of independent models we need to average to produce a correct output post-rounding.

7. The proof is in Appendix D.

Equation (19) gives an ensemble complexity bound for any specific test input $\hat{\boldsymbol{x}}$. Since the domain is finite we can turn this into a uniform bound by taking

$$N \geq 8 \max_{\hat{\boldsymbol{x}} \in \mathbb{K}} \left\{ \sigma^2(\hat{\boldsymbol{x}}) \cdot \max \left\{ \frac{1}{\mu_0(\hat{\boldsymbol{x}})^2}, \frac{1}{\mu_1(\hat{\boldsymbol{x}})^2} \right\} \right\} \ln \left(\frac{2k}{\delta} \right) \tag{20}$$

C.2. Asymptotic behavior of $\mu(\hat{\boldsymbol{x}})$ and $\sigma^2(\hat{\boldsymbol{x}})$

To conclude the proof of Theorem 3 and derive an asymptotic bound on the ensemble complexity of the permutation learning problem we need to analyze the asymptotic behavior of $\mu(\hat{\boldsymbol{x}})$ and $\sigma^2(\hat{\boldsymbol{x}})$. The asymptotic orders are given in the following technical lemma:

Lemma 7 *Let $\hat{\boldsymbol{x}} \in \mathbb{K}$ be a test input with $n_{\hat{\boldsymbol{x}}}$ nonzero entries and let $\mu_0(\hat{\boldsymbol{x}})$, $\mu_1(\hat{\boldsymbol{x}})$ and $\sigma^2(\hat{\boldsymbol{x}})$ be as above. Then, $\sigma^2(\hat{\boldsymbol{x}}) \in \mathcal{O}(1/k)$ and, depending on the relationship between $n_{\hat{\boldsymbol{x}}}$ and k we have*

$$|\mu_0(\hat{\boldsymbol{x}})| \in \begin{cases} \Theta(1/k) & \text{if } n_{\hat{\boldsymbol{x}}} \text{ is constant} \\ \Theta\left(\sqrt{n_{\hat{\boldsymbol{x}}}}/k\right) & \text{if } n_{\hat{\boldsymbol{x}}} \text{ is non-constant} \\ & \text{and sublinear in } k \\ \Theta\left(1/\sqrt{k}\right) & \text{if } n_{\hat{\boldsymbol{x}}} = ck, \text{ for some } c \in (0,1] \end{cases}$$

and

$$|\mu_1(\hat{\boldsymbol{x}})| \in \begin{cases} \Theta(1) & \text{if } n_{\hat{\boldsymbol{x}}} \text{ is constant} \\ \Theta\left(1/\sqrt{n_{\hat{\boldsymbol{x}}}}\right) & \text{if } n_{\hat{\boldsymbol{x}}} \text{ is non-constant} \\ & \text{and sublinear in } k \\ \Theta\left(1/\sqrt{k}\right) & \text{if } n_{\hat{\boldsymbol{x}}} = ck, \text{ for some } c \in (0,1) \\ \Theta(1/k) & \text{if } n_{\hat{\boldsymbol{x}}} = k \end{cases}$$

The proof of Lemma 7 is further deferred to Appendix D. In light of these asymptotic results, the uniform bound of Equation (20) behaves like $\mathcal{O}(k \log k)$. Indeed, this follows directly from the fact that the maximum of the reciprocals of the means occurs for $\mu_0(\hat{\boldsymbol{x}})$ when $n_{\hat{\boldsymbol{x}}}$ is constant which yields a bound of order $\mathcal{O}(k^2)$. This simple observation is enough to conclude the proof of Theorem 3.

Appendix D. Proof of Lemma 7

In this section, we analyze the mean and variance of the predictions under the limit NTK, expressing them as functions of the test vector $\hat{\boldsymbol{x}}$. Specifically, we characterize their dependence on the vector length k and the number of nonzero elements $n_{\hat{\boldsymbol{x}}}$. We establish that the variance of the predictions scales as $\mathcal{O}(1/k)$ while the mean coordinates exhibit two different behaviors: they scale with $\Theta(\sqrt{n_{\hat{\boldsymbol{x}}}}/k)$ when $P^*\hat{\boldsymbol{x}} = 0$ and $\Theta(1/k)$ when $P^*\hat{\boldsymbol{x}} > 0$. We begin by calculating the variance: a direct substitution on the formula for the variance $\Sigma(\hat{\boldsymbol{x}})$ of Equation (4) we find that the variance for a test input $\hat{\boldsymbol{x}} \in \mathbb{K}$ is given by

$$\Sigma(\hat{\boldsymbol{x}}) = \left(\frac{d}{2} + \mathbf{f}^\top AMA\mathbf{f} - 2\mathbf{f}^\top A\mathbf{g} \right) I_k \tag{21}$$

where

$$M = \left(\frac{d}{2} - c\right) I_k + c\mathbf{1}\mathbf{1}^\top \qquad (22) \qquad A = \frac{1}{d-c} I_k - \frac{c}{(d-c)(d-c+ck)}\mathbf{1}\mathbf{1}^\top \quad (23)$$

and for each $i = 1, 2, \ldots, k$:

$$\mathbf{g}_i = \frac{\cos\hat{\theta}_i(\pi - \hat{\theta}_i) + \sin\hat{\theta}_i}{2k\pi} \tag{24}$$

The scalars c and d, the vector $\mathbf{f}$, and the angles $\hat{\theta}_i$ are as defined in Equation (7), Equation (10) and Equation (11), respectively. This shows that the output coordinates are independent Gaussian random variables with the same covariance. Notice that whenever the test input is part of the training set, the variance is zero. This is expected and consistent with the NTK theory, which guarantees exact learning on the training set. In what follows we will use the notation $\sigma^2(\hat{\boldsymbol{x}}) := (d/2 + \mathbf{f}^\top AMA\mathbf{f} - 2\mathbf{f}^\top A\mathbf{g}) \in \mathbb{R}_{\geq 0}$. With that, we are now ready to compute the asymptotic orders of the relevant quantities.

Preliminary quantities. We begin by rewriting several useful expressions from Appendix B and adding other relevant expressions to complement them. Recall that:

$$d = \frac{1}{k} \quad \text{and} \quad c = \frac{1}{2\pi k}$$

We further introduce

$$d' = \frac{d}{2} = \frac{1}{2k} \quad \text{and} \quad c' = c = \frac{1}{2\pi k}$$

Additionally, we rewrite Equation (12) in an alternative form:

$$\hat{\theta}_i = \begin{cases} \arccos\left(\frac{1}{\sqrt{n_{\hat{\boldsymbol{x}}}}}\right) & \text{if } \hat{\boldsymbol{x}}_i \neq 0 \\ \frac{\pi}{2} & \text{otherwise} \end{cases}$$

From this, we directly obtain the cosine and sine of $\hat{\theta}_i$:

$$\cos\hat{\theta}_i = \begin{cases} \frac{1}{\sqrt{n_{\hat{\boldsymbol{x}}}}} & \text{if } \hat{\boldsymbol{x}}_i \neq 0 \\ 0 & \text{otherwise} \end{cases} \qquad \sin\hat{\theta}_i = \begin{cases} \sqrt{1 - \frac{1}{n_{\hat{\boldsymbol{x}}}}} & \text{if } \hat{\boldsymbol{x}}_i \neq 0 \\ 1 & \text{otherwise} \end{cases}$$

Using these quantities, we rewrite Equation (24) as:

$$\mathbf{g}_i = \begin{cases} \frac{1}{2\pi k\sqrt{n_{\hat{\boldsymbol{x}}}}}\left(\pi - \arccos\left(1/\sqrt{n_{\hat{\boldsymbol{x}}}}\right) + \sqrt{n_{\hat{\boldsymbol{x}}} - 1}\right) & \text{if } \hat{\boldsymbol{x}}_i \neq 0 \\ \frac{1}{2\pi k} & \text{otherwise} \end{cases}$$

and Equation (10) as

$$\mathbf{f}_i = \begin{cases} \mathbf{g}_i + \frac{\pi - \arccos\left(1/\sqrt{n_{\hat{\boldsymbol{x}}}}\right)}{2\pi k\sqrt{n_{\hat{\boldsymbol{x}}}}} & \text{if } \hat{\boldsymbol{x}}_i \neq 0, \\ \mathbf{g}_i & \text{otherwise} \end{cases}$$

D.1. Computing the order of the variance $\Sigma(\hat{\boldsymbol{x}})$

We begin by recalling the expression for the variance (substituting d' and c'):[8]

$$\Sigma(\hat{\boldsymbol{x}}) = \left(d' + \mathbf{f}^\top AMA\mathbf{f} - 2\mathbf{f}^\top A\mathbf{g}\right) I_k$$

where

$$M = \left(d' - c'\right) I_k + c'\mathbf{1}\mathbf{1}^\top \quad \text{and} \quad A = \frac{1}{d-c} I_k - \frac{c}{(d-c)(d-c+ck)}\mathbf{1}\mathbf{1}^\top$$

We aim to show that the variance is bounded by $\mathcal{O}\left(1/k\right)$. To facilitate this, we rewrite $\Sigma(\hat{\boldsymbol{x}})$ by expanding the matrix multiplication:

$$\begin{aligned}\Sigma(\hat{\boldsymbol{x}}) &= d' I_k + \mathbf{f}^\top \left(AMA\mathbf{f} - 2A(\mathbf{f} - \mathbf{z})\right) I_k \\ &= d' I_k + \mathbf{f}^\top \left(AMA\mathbf{f} - 2A\mathbf{f}\right) I_k + 2\mathbf{f}^\top A\mathbf{z} I_k\end{aligned}$$

where we define $\mathbf{z} = \mathbf{f} - \mathbf{g}$ for notational simplicity. It is straightforward to observe that the first term, $d' I_k$, is bounded by $\mathcal{O}\left(1/k\right)$. Thus, we focus our attention on the remaining two terms. For the second term, we begin by showing that $AMA - 2A$ has a maximum eigenvalue of $\mathcal{O}(1)$, and therefore:

$$\mathbf{f}^\top (AMA - 2A)\mathbf{f} \in \mathcal{O}\left(1/k\right)$$

We begin by expressing AMA in a more manageable form. A direct computation reveals:

$$AMA = uI_k - v\mathbf{1}\mathbf{1}^\top$$

where

$$u = \frac{d' - c'}{(d-c)^2} \quad \text{and} \quad v = \frac{1}{k}\left(\frac{d' - c' + kc'}{(d-c+kc)^2} - \frac{d' - c'}{(d-c)^2}\right)$$

Subtracting $2A$ from this expression gives:

$$AMA - 2A = \left(u - 2a\right) I_k + \left(v + 2b\right)\mathbf{1}\mathbf{1}^\top$$

where

$$a = \frac{d-c}{(d-c)^2} \quad \text{and} \quad b = \frac{c}{\left(d-c+ck\right)\left(d-c\right)}$$

This matrix has two unique eigenvalues: $u - 2a$ (with multiplicity $k-1$) and $u - 2a + (v + 2b)k$. We will show that the largest eigenvalue is $\mathcal{O}(1)$. To this end, we evaluate these two quantities, starting with $u - 2a$:

$$\begin{aligned}u - 2a &= \frac{d' - c' - 2(d-c)}{(d-c)^2} \\ &= -\frac{2\pi(3\pi - 1)k}{(2\pi - 1)^2}\end{aligned}$$

8. Even though in this case $d' = d/2$ and $c' = c$ we deliberately introduce the new variables to avoid carrying the $1/2$ factor.

which is negative. For the second eigenvalue, we find:

$$u - 2a + (v + 2b)k = -\frac{2\pi(3\pi - 1)k}{(2\pi - 1)^2} + \frac{2\pi k\,(\pi + k - 1)}{(2\pi + k - 1)^2} - \frac{2\pi k(\pi - 1)}{(2\pi - 1)^2} + \frac{4\pi k}{(2\pi - 1)\,(2\pi + k - 1)}$$
$$= \left(\frac{4\pi}{2\pi + k - 1} + \frac{\pi + k - 1}{(2\pi + k - 1)^2}\right) k$$

which is positive and clearly $\mathcal{O}(1)$. With this result, we can bound the multiplication by the norms of its components. Since $\lambda_{\max}(AMA - 2A) \in \mathcal{O}(1)$ and $\|\mathbf{f}\|^2 \in \mathcal{O}\left(1/k\right)$ (since each $\mathbf{f}_i \in \mathcal{O}\left(1/k\right)$), we conclude:

$$\mathbf{f}^\top (AMA - 2A)\mathbf{f} \in \mathcal{O}\left(1/k\right)$$

We now turn to the third term, $2\mathbf{f}^\top A\mathbf{z}$. Expressing $\mathbf{z}$ component-wise we have:

$$\mathbf{z}_i = \begin{cases} \frac{\pi - \arccos\left(1/\sqrt{n_{\hat{\boldsymbol{x}}}}\right)}{2\pi k\sqrt{n_{\hat{\boldsymbol{x}}}}} & \text{if } \hat{\boldsymbol{x}}_i \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

We then decompose the product $\mathbf{f}^\top A\mathbf{z}$ as:

$$\mathbf{f}^\top A\mathbf{z} = a\mathbf{f}^\top \mathbf{z} - b(\mathbf{f}^\top \mathbf{1})(\mathbf{1}^\top \mathbf{z}) \tag{25}$$

where

$$a = \frac{1}{d - c} \quad \text{and} \quad b = \frac{2\pi k}{(2\pi - 1)(2\pi + k - 1)}$$

The first term $a\mathbf{f}^\top \mathbf{z}$ can be expressed as:

$$a\mathbf{f}^\top \mathbf{z} = \frac{1}{(2\pi - 1)(2\pi k)^2}\left(2\pi - 2\arccos\left(1/\sqrt{n_{\hat{\boldsymbol{x}}}}\right) - \sqrt{n_{\hat{\boldsymbol{x}}} - 1}\right)\left(\pi - 2\arccos\left(1/\sqrt{n_{\hat{\boldsymbol{x}}}}\right)\right)$$

which is positive and $\mathcal{O}\left(1/k\right)$. For the second term $b(\mathbf{f}^\top \mathbf{1})(\mathbf{1}^\top \mathbf{z})$, we start by expressing the individual quantities $\mathbf{f}^\top \mathbf{1}$ and $\mathbf{1}^\top \mathbf{z}$:

$$\mathbf{f}^\top \mathbf{1} = \frac{\sqrt{n_{\hat{\boldsymbol{x}}}}}{\pi k}\left(\pi - \arccos\left(1/\sqrt{n_{\hat{\boldsymbol{x}}}}\right)\right) + \frac{\sqrt{n_{\hat{\boldsymbol{x}}}}\sqrt{n_{\hat{\boldsymbol{x}}} - 1}}{2\pi k} + \frac{k - n_{\hat{\boldsymbol{x}}}}{2\pi k}$$

and

$$\mathbf{1}^\top \mathbf{z} = \frac{\sqrt{n_{\hat{\boldsymbol{x}}}}}{2\pi k}\left(\pi - \arccos\left(1/\sqrt{n_{\hat{\boldsymbol{x}}}}\right)\right)$$

Therefore, the entire term can be expressed as:

$$b(\mathbf{f}^\top \mathbf{1})(\mathbf{1}^\top \mathbf{z}) = \frac{1}{2\pi k(2\pi - 1)(2\pi - 1 + k)}\Big(2n\left(\pi - \arccos\left(1/\sqrt{n_{\hat{\boldsymbol{x}}}}\right)\right)^2$$
$$+ n_{\hat{\boldsymbol{x}}}\sqrt{n_{\hat{\boldsymbol{x}}} - 1}\left(\pi - \arccos\left(1/\sqrt{n_{\hat{\boldsymbol{x}}}}\right)\right)$$
$$+ (k - n_{\hat{\boldsymbol{x}}})\sqrt{n_{\hat{\boldsymbol{x}}}}\left(\pi - \arccos\left(1/\sqrt{n_{\hat{\boldsymbol{x}}}}\right)\right)\Big)$$

which is positive and $\mathcal{O}\left(1/k\right)$, therefore, the subtraction of Equation (25) (which is equal to $\mathbf{f}^\top A\mathbf{z}$) is $\mathcal{O}\left(1/k\right)$. Combining the bounds on all three terms, we conclude that:

$$\Sigma(\hat{\boldsymbol{x}}) \in \mathcal{O}\left(1/k\right) I_k$$

completing the proof of the first part of Lemma 7.

D.2. Computing the order of the mean $\mu(\hat{\boldsymbol{x}})$

We first recall the expression for the mean:

$$\mu(\hat{\boldsymbol{x}}) = P^* \left(\frac{1}{d-c} I_k - \frac{c}{(d-c)(d-c+ck)} \mathbf{1}\mathbf{1}^\top \right) \mathbf{f}$$

which we decompose into two terms:

$$\mu(\hat{\boldsymbol{x}}) = \frac{P^* \mathbf{f}}{d-c} - \frac{c\mathbf{1}\mathbf{1}^\top \mathbf{f}}{(d-c)(d-c+ck)} \tag{26}$$

The second term simplifies since $\mathbf{1}$ is the vector of all ones, effectively absorbing the permutation P^*. For the first term in Equation (26), we obtain:

$$\frac{(P_j^*)^\top \mathbf{f}}{d-c} = \begin{cases} \frac{1}{2\pi-1} \left(\frac{\pi - \arccos\left(1/\sqrt{n_{\hat{\boldsymbol{x}}}}\right)}{\sqrt{n_{\hat{\boldsymbol{x}}}}} + \sqrt{1 - \frac{1}{\sqrt{n_{\hat{\boldsymbol{x}}}}}} \right) & \text{if } \hat{\boldsymbol{x}}_j \neq 0 \\ \frac{1}{2\pi-1} & \text{otherwise} \end{cases}$$

The second term is a constant vector with coefficient:

$$\frac{c\mathbf{1}\mathbf{1}^\top \mathbf{f}}{(d-c)(d-c+ck)} = \frac{1}{(2\pi-1)(2\pi-1+k)} \left(\frac{n_{\hat{\boldsymbol{x}}} \left(\pi - \arccos\left(1/\sqrt{n_{\hat{\boldsymbol{x}}}}\right)\right)}{2\sqrt{n_{\hat{\boldsymbol{x}}}}} + n_{\hat{\boldsymbol{x}}} \sqrt{1 - \frac{1}{n_{\hat{\boldsymbol{x}}}}} + k - n_{\hat{\boldsymbol{x}}} \right)$$

We now evaluate each case of the mean separately.
Case 1: For $\mu_0(\hat{\boldsymbol{x}})$, that is, when $P^*\hat{\boldsymbol{x}} = 0$, we have:

$$\begin{aligned} \mu_0(\hat{\boldsymbol{x}}) =& \frac{1}{2\pi-1} \left(1 - \frac{k - n_{\hat{\boldsymbol{x}}}}{2\pi+k-1} - \frac{2\pi k n_{\hat{x}}}{(2\pi+k-1)\left(\pi k \sqrt{n_{\hat{\boldsymbol{x}}}}\right)} \left(\pi - \arccos\left(1/\sqrt{n_{\hat{\boldsymbol{x}}}}\right)\right) \right) \\ & - \frac{1}{2\pi-1} \left(\frac{2\pi k n_{\hat{x}}}{(2\pi+k-1)(2\pi k)} \sqrt{1 - \frac{1}{\sqrt{n_{\hat{\boldsymbol{x}}}}}} \right) \end{aligned} \tag{27}$$

The absolute value of this expression behaves like $\Theta\left(\sqrt{n_{\hat{\boldsymbol{x}}}}/k\right)$, and setting $n_{\hat{\boldsymbol{x}}}$ to different regimes yields the bounds established in Lemma 7, namely:

$$|\mu_0(\hat{\boldsymbol{x}})| \in \begin{cases} \Theta(1/k) & \text{if } n_{\hat{\boldsymbol{x}}} \text{ is constant} \\ \Theta\left(\sqrt{n_{\hat{\boldsymbol{x}}}}/k\right) & \text{if } n_{\hat{\boldsymbol{x}}} \text{ is non-constant} \\ & \text{and sublinear in } k \\ \Theta\left(1/\sqrt{k}\right) & \text{if } n_{\hat{\boldsymbol{x}}} = ck \text{ for } c \in (0,1] \end{cases}$$

Case 2: For $\mu_1(\hat{\boldsymbol{x}})$, that is, when $P^*\hat{\boldsymbol{x}} > 0$, we have:

$$\begin{aligned} \mu_1(\hat{\boldsymbol{x}}) =& \frac{2\pi-1}{2\pi+k-1} \left(\frac{2}{(2\pi-1)\sqrt{n_{\hat{\boldsymbol{x}}}}} \left(\pi - \arccos\left(1/\sqrt{n_{\hat{\boldsymbol{x}}}}\right)\right) + \frac{1}{(2\pi-1)} \sqrt{1 - \frac{1}{\sqrt{n_{\hat{\boldsymbol{x}}}}}} \right) \\ & + \frac{k - n_{\hat{\boldsymbol{x}}}}{2\pi+k-1} \left(\frac{2}{(2\pi-1)\sqrt{n_{\hat{\boldsymbol{x}}}}} \left(\pi - \arccos\left(1/\sqrt{n_{\hat{\boldsymbol{x}}}}\right)\right) + \frac{1}{(2\pi-1)} \sqrt{1 - \frac{1}{\sqrt{n_{\hat{\boldsymbol{x}}}}}} - \frac{1}{(2\pi-1)} \right) \end{aligned} \tag{28}$$

When setting $n_{\hat{\boldsymbol{x}}} = k$, we note that the second term becomes zero and the first term becomes $\Theta\left(1/k\right)$. Alternatively, the first term in Equation (28) is $\Theta\left(1/k\right)$ and the second term is $\Theta\left(1/\sqrt{n_{\hat{\boldsymbol{x}}}}\right)$, implying $\mu_1(\hat{\boldsymbol{x}}) \in \Theta\left(1/\sqrt{n_{\hat{\boldsymbol{x}}}}\right)$. Setting $n_{\hat{\boldsymbol{x}}}$ to different regimes yields the bounds established in Lemma 7, namely:

$$|\mu_1(\hat{\boldsymbol{x}})| \in \begin{cases} \Theta\left(1/\sqrt{n_{\hat{\boldsymbol{x}}}}\right) & \text{if } n_{\hat{\boldsymbol{x}}} \text{ is non-constant} \\ & \text{and sublinear in } k, \\ \Theta\left(1/\sqrt{k}\right) & \text{if } n_{\hat{\boldsymbol{x}}} = ck \text{ for } c \in (0,1) \\ \Theta\left(1/k\right) & \text{if } n_{\hat{\boldsymbol{x}}} = k \end{cases}$$

This concludes the proof of the second part of Lemma 7