A KERNEL-BASED VIEW OF LANGUAGE MODEL FINE-TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

It has become standard to solve NLP tasks by fine-tuning pre-trained language models (LMs), especially in low-data settings. There is minimal theoretical understanding of empirical success, e.g., why fine-tuning a model with 10^8 or more parameters on a couple dozen training points does not result in overfitting. We investigate whether the Neural Tangent Kernel (NTK)—which originated as a model to study the gradient descent dynamics of infinitely wide networks with suitable random initialization—*describes* fine-tuning of pre-trained LMs. This study was inspired by the decent performance of NTK for computer vision tasks (Wei et al., 2022). We extend the NTK formalism to Adam and use Tensor Programs (Yang, 2020b) to characterize conditions under which the NTK lens may describe fine-tuning updates to pre-trained language models. Extensive experiments on 14 NLP tasks validate our theory and show that formulating the downstream task as a masked word prediction problem through prompting often induces kernel-based dynamics during fine-tuning. Finally, we use this kernel view to propose an explanation for the success of parameter-efficient subspace-based fine-tuning methods.

1 INTRODUCTION

It is now customary to solve most supervised natural language processing (NLP) tasks such as topic classification and textual entailment by fine-tuning a pre-trained language model (e.g., Devlin et al. (2019); Liu et al. (2020b); Clark et al. (2020); Raffel et al. (2020); Joshi et al. (2020)). We lack theoretical understanding of this fine-tuning paradigm. Why do we not see over-fitting when fine-tuning a very large language model using a couple dozen instances of the supervised task? Why is fine-tuning so sensitive to details such as whether or not we include a prompt (e.g., adding "It was [great/terrible]" for sentiment analysis (Schick & Schütze, 2021; Gao et al., 2021)? Why does restricting optimization to a low-rank subspace of model parameters (Hu et al., 2021; Li et al., 2018; Aghajanyan et al., 2021) still result in performance comparable to full fine-tuning? Answering such questions requires understanding how the sequence of parameter updates changes in various scenarios, e.g., the addition of a prompt, or the introduction of randomly initialized parameters. The current theory of deep learning, at first sight, seems too primitive to address such questions, especially since fine-tuning has to start from a parameter initialization inherited from pre-training.

Recently, Wei et al. (2022) suggested replacing fine-tuning with Neural Tangent Kernel (NTK), an idea invented for the study of infinite-width deep neural networks (Jacot et al., 2018; Du et al., 2019) and previously applied to solving vision tasks with infinitely wide ConvNets (Arora et al., 2019). They note that the NTK can be defined for any neural model f and any initialization θ_0 by representing an input ξ by the gradient it induces $\nabla f(\xi;\theta_0)$, which yields a kernel matrix:

$$\mathcal{K}(\xi,\xi') = \langle \nabla f(\xi;\theta_0), \nabla f(\xi';\theta_0) \rangle. \tag{1}$$

This kernel is well-defined for any parameter vector θ_0. However, for an infinite-width network initialized with θ_0 sampled from a suitably-scaled Gaussians, it can be shown that the kernel matrix is unchanged during gradient descent, which turns the classification task into a form of kernel regression with respect to this kernel (Jacot et al., 2018). In the fine-tuning setting, however, the initialization θ_0 is inherited from the pre-trained network, and not sampled from the Gaussian distribution. Nevertheless, Wei et al. (2022) found that kernel regression using this "empirical NTK"

(eNTK) defined with the inherited θ_0 performs well, achieving classification accuracy within 6% absolute of actual fine-tuning on several image recognition tasks. In other words, their work hints that mathematical understanding of the fine-tuning phenomenon (e.g., its sample efficiency) could go via the theory of kernel classifiers.

The current paper furthers an empirical and theoretical understanding of the pre-training and fine-tuning (FT) paradigm for NLP tasks. Our contributions are:

1. We formally extend the standard NTK theory developed for gradient descent to characterize kernel-based dynamics when training with Adam. See Section 3.1.
2. We use Tensor Programs formally extend infinite-width analysis to account for a pre-trained initialization and characterize conditions under which fine-tuning can exhibit kernel behavior. See Section 3.2.
3. We perform extensive experiments on 14 diverse NLP tasks and find that the inclusion of a meaningful prompt often allows FT optimization dynamics to be described by kernel-based dynamics. See Section 4.
4. We straightforwardly apply the kernel view of FT dynamics to formally analyze the success of fine-tuning methods that update in a low-rank subspace of model parameters (e.g., LoRA, Hu et al. (2021)). See Appendix C.

2 PRELIMINARIES

We consider masked language models (MLMs), such as BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2020b), which are trained to minimize the cross-entropy loss on independently predicting masked tokens. We focus on fine-tuning (FT) methods, which adapt the pre-trained model to a new input distribution $\mathcal{S}_{\text{FT}}$ using additional training on the C-way downstream classification task.

1. **Standard FT** (Devlin et al., 2019; Liu et al., 2020b): To solve a C-way downstream classification task, initialize and learn a new classifier head $\Gamma : \mathbb{R}^n \to \mathbb{R}^C$ on top of the contextual `[CLS]` embedding, denoted $h_{\texttt{[CLS]}}$. In this case, the model output $f : \mathcal{S}_{\text{FT}} \to \mathbb{R}^C$ for the eNTK construction is $f(s) = \Gamma(h_{\texttt{[CLS]}}(s))$.
2. **Prompt-based FT** (Schick & Schütze, 2021; Gao et al., 2021): Add a natural language prompt (e.g. "This is `[MASK]`.") in addition to the downstream task input, and use the pre-trained MLM to fill in the masked token. Compute the logits over task-relevant words (e.g., "great" and "terrible") using the corresponding columns of Φ, denoted $\tilde{\Phi} \in \mathbb{R}^{n \times C}$. These logits will serve as surrogates to solve the downstream task. In this case, the model output $f : \mathcal{S}_{\text{FT}} \to \mathbb{R}^C$ for the eNTK construction is $f(s) = \tilde{\Phi}^\top h_{\texttt{[MASK]}}(s)$.

To study kernel dynamics, we consider a neural network $f(\xi; \theta)$ that takes input ξ and computes a scalar output[1] using θ as the parameters. Gradient-based updates to the model parameters involve computing a loss function ℓ and $\frac{\partial \ell}{\partial \theta}$, which is decomposed as $\frac{\partial \ell}{\partial f} \frac{\partial f}{\partial \theta}$. The first term is defined as the output derivative (Definition 2.1), and the second term is used to define kernel behavior (Definition 2.2), adapting the notion of *lazy regime* (Woodworth et al., 2020)) to an arbitrary initialization.

Definition 2.1 (Output Derivative)**.** The output derivative $\chi(\xi, y, \theta)$ for a network f with parameters θ, loss function ℓ, and input ξ with label y is defined as $\chi(\xi, y, \theta) = \frac{\partial \ell(f(\xi;\theta), y)}{\partial f}$. We also define the output derivative applied at time t as $\chi_t = \chi(\xi_t, y_t, \theta_{t-1})$, where ξ_t is the input at time t with label y_t. For ease of notation, we often absorb y into ξ and write $\chi(\xi, \theta)$ and $\chi(\xi, f)$ interchangeably.

Definition 2.2 (Kernel Behavior)**.** Let θ_t be the parameters after t steps of training by a gradient-based optimization algorithm, and let ξ be an arbitrary fixed input. We say this training process of the network demonstrates *kernel behavior* if the following properties are satisfied.

1. *Linearization*: The change of the network can be well-approximated by its first order Taylor expansion, i.e., $f(\xi; \theta_t) - f(\xi; \theta_{t-1}) \approx \langle \nabla f(\xi; \theta_{t-1}), \theta_t - \theta_{t-1} \rangle$;

[1]Note that for C-way classification, f outputs a vector in $\mathbb{R}^C$. We say f exhibits kernel behavior if the Linearization and Fixed Features properties hold for every component of f. The subsequent definition of a kernel analog also generalizes to a vector output, where ν_t is a vector in $\mathbb{R}^C$ and $\mathcal{K}^{(\mathcal{A})}(\xi, \xi_t)$ is a matrix in $\mathbb{R}^{C \times C}$.

2. *Fixed Features*: The gradient at step t is approximately the same as before training, i.e., $\nabla f(\xi;\theta_t) \approx \nabla f(\xi;\theta_0)$.

∇f denotes the gradient of f w.r.t. θ. "Closeness to kernel behavior" is quantified using the difference in the quantities on the two sides of the $\approx$ symbol. We formalize the approximations in Definition D.3.

Past work (Jacot et al., 2018) has shown that if gradient-based training exhibits kernel behavior, then the function change can be expressed in terms of a fixed kernel (i.e., the kernel analog).

Definition 2.3 (Kernel Analog)**.** Suppose optimization of the parameters θ of a model f using the gradient-based update algorithm $\mathcal{A}$ exhibits kernel behavior (Definition 2.2). Then, we say that a kernel $\mathcal{K}^{(\mathcal{A})}$ is the *kernel analog* of the optimization algorithm $\mathcal{A}$ if for every $t > 0$, there exists ν_t such that for any input ξ,

$$f(\xi;\theta_t) - f(\xi;\theta_{t-1}) \approx -\nu_t \mathcal{K}^{(\mathcal{A})}(\xi, \xi_t) \tag{2}$$

where ξ_t is the training input[2] of step t, θ_t is the parameter after step t.

Jacot et al. (2018) showed that $\mathcal{K}^{(\text{SGD})}$ is the kernel analog for SGD.

Definition 2.4 (Neural Tangent Kernel $\mathcal{K}^{(\text{SGD})}$)**.** $\mathcal{K}^{(\text{SGD})}(\xi,\xi') = \langle \nabla f(\xi;\theta_0), \nabla f(\xi';\theta_0)\rangle$

3 Theory

3.1 Kernel Derivation for Adam

Computing the eNTK requires using the kernel analog (Definition 2.3) of the chosen optimization algorithm $\mathcal{A}$. Previous work has shown that in the early stages of training, full-batch (Ma et al., 2022) and mini-batch (Malladi et al., 2022) Adam updates reduce to coordinate-wise normalization on the gradient, defined below as SignGD.

Definition 3.1 (SignGD)**.** SignGD is a gradient-based optimization algorithm that updates parameters as $\theta_t = \theta_{t-1} - \eta \operatorname{sign}(\nabla \ell_t(\xi_t;\theta_{t-1}))$, where sign is applied element-wise.

We define the sign-based kernel below and prove it to be the correct kernel analog for SignGD.

Definition 3.2 (Asymmetric SignGD Kernel)**.** $\mathcal{K}^{(\text{A-SignGD})}(\xi,\xi') = \langle \nabla f(\xi;\theta_0), \operatorname{sign}(\nabla f(\xi';\theta_0)\rangle$.

Theorem 3.3 (Informal version of Theorem D.4)**.** *If a network is trained with SignGD and exhibits kernel behavior (Definition 2.2), then the training dynamics follow*

$$f(\xi;\theta_t) - f(\xi;\theta_{t-1}) \approx -\eta \operatorname{sign}(\chi_t) \mathcal{K}^{(\textit{A-SignGD})}(\xi,\xi_t),$$

where χ_t is the output derivative (Definition 2.1).

We solve the asymmetric kernel regression as suggested in He et al. (2022), but the difficulties of solving the kernel regression problem with an asymmetric kernel (Appendix A.3) motivate us to also use the symmetric SignGD kernel.

Definition 3.4 (SignGD Kernel)**.** $\mathcal{K}^{(\text{SignGD})}(\xi,\xi') = \langle \operatorname{sign}(\nabla f(\xi;\theta_0)), \operatorname{sign}(\nabla f(\xi';\theta_0))\rangle$

3.2 Prompt-Based Fine-Tuning Can Exhibit Kernel Behavior

We analyze how prompt-based FT can exhibit kernel behavior (Definition 2.2) as the network width grows large. First, we formalize how changing the architecture width impacts pre-training.

Definition 3.5 (Pre-Training Scheme)**.** A pre-training scheme $(\mathcal{X}, \mathcal{A}, \mathcal{F}^n)$ with width n contains the dataset $\mathcal{X}$, optimizer $\mathcal{A}$ and its hyperparameters, and a model architecture $\mathcal{F}^n$. Let $f^n \sim (\mathcal{X}, \mathcal{A}, \mathcal{F}^n)$ denote a model resulting from training the architecture $\mathcal{F}^n$ on the dataset $\mathcal{X}$ with optimizer $\mathcal{A}$.

The reliance of the architecture on the width is given by Tensor Programs (Yang, 2020a): for example, in a Transformer, n corresponds to the embedding dimension. Analogous to Saunshi et al. (2021), we

[2]For simplicity, we assume the batch size is 1.

Task	SST-2	SST-5	MR	CR	MPQA	Subj	TREC	AG News	MNLI	SNLI	QNLI	RTE	MRPC	QQP
Task type	—— sentiment ——				polarity	subj.	—topic clf.—		—— entailment ——				– para. detect. –	
Num. classes C	2	5	2	2	2	2	6	4	3	3	2	2	2	2
eNTK solvable	✓	✓	✓	✓	(✓)	✓		✓			(✓)	✓	✓	✓
Linearization	✓	✓	✓	✓		✓					✓	✓	✓	✓
Fixed Features	✓	✓	✓	✓	✓	✓			✓	✓	✓	✓	✓	✓
Kernel behavior	✓	✓	✓	✓		✓					(✓)	✓	✓	✓

Table 1: In our experiments, 9 out of 14 tasks induce kernel behavior during few-shot prompt-based fine-tuning. We test whether the eNTK solves the task and if the Linearization and Fixed Features properties of kernel behavior (Definition 2.2) hold. We say the eNTK solves the task if the kernel analog achieves a performance within 5% of the fine-tuning method in at least 2/4 settings (in parentheses: only in 1/4 settings; results are shown in Table 4). We say that Linearization is satisfied if the linearized model (see Appendix B.2) improves the pre-trained model by at least 50% of the amount that fine-tuning improves it (Figure 2), and we say that Fixed Features is satisfied if the average element-wise distance between the kernels before and after fine-tuning are less than 1.0 (Table 6).

reason that prompting transforms the downstream task into a fill-in-the-blank problem, and thus the downstream task can be viewed as a subcase of the pre-training task. We then assume that a wider pre-trained network will be better at filling in masked tokens and that an infinitely wide pre-trained network can solve the downstream task perfectly when using a suitable prompt.

Definition 3.6 (Natural Task in the Infinite-Width Limit)**.** A downstream task Ξ is natural with respect to a pre-training scheme $(\mathcal{X}, \mathcal{A}, \mathcal{F}^n)$ if, for any pre-trained model $f^n \sim (\mathcal{X}, \mathcal{A}, \mathcal{F}^n)$ and any downstream example $(\xi, y) \in \Xi$,

$$\lim_{n\to\infty} \chi(\xi, y, f^n) = 0. \tag{3}$$

We assume that the network can be written as a Tensor Program (Yang, 2019; 2020a;b), which is sufficiently general to allow our theory to describe many complex architectures (e.g., Transformers). The network must also be (1) *stable*: its output does not grow with width (i.e., the infinite-width limit is meaningful), and (2) *non-trivial*: its output can be updated during fine-tuning. The below theorem formalizes the intuition that if the pre-trained network is already decent at solving the downstream task, the network needs to only mildly adapt to solve the downstream task. Notably, we extend standard NTK theory to account for an arbitrary initialization and to characterize early-stage training with Adam using results from Section 3.1.

Theorem 3.7 (Informal version of Theorem D.5)**.** *Assume the downstream task Ξ is natural in the infinite-width limit with respect to a pre-training scheme $(\mathcal{X}, \mathcal{A}, \mathcal{F}^n)$, and the model $f \sim (\mathcal{X}, \mathcal{A}, \mathcal{F}^n)$ is stable, non-trivial, and can be written as a Tensor Program. Then prompt-based FT of f will exhibit the Linearization and Fixed Features properties of kernel behavior (Definition 2.2).*

4 Experiments

We compute the eNTK as described in Section 2 for different optimization algorithms and FT settings. eNTK performance being comparable to FT performance is a necessary but not sufficient condition for FT to exhibit kernel behavior (Definition 2.2), so we also directly measure if the Linearization and Fixed Features properties hold (Appendix B.2). Our experiments are on 14 NLP tasks in the few-shot setting with manual prompt templates from Gao et al. (2021). We summarize our results in Table 1. We find that the eNTK can solve 11 out of 14 tasks comparably to prompt-based fine-tuning, out of which 9 induce kernel behavior during fine-tuning. For tasks that the eNTK cannot solve, we conjecture that the prompt is not well-designed for the task (in the sense of Definition 3.6), forcing the pre-trained model to adapt more during FT. Our results show that FT optimization dynamics depend on the downstream task and the inclusion of a meaningful prompt. Additional implementation details, results, and discussion are in Appendices A and B.

REFERENCES

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 7319–7328, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.568. URL https://aclanthology.org/2021.acl-long.568.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/dbc4d84bfcfe2284ba11beffb853a8c4-Paper.pdf.

Roy Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. The second PASCAL recognising textual entailment challenge. 2006. URL https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.60.8552&rep=rep1&type=pdf.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. The fifth PASCAL recognizing textual entailment challenge. In *TAC*, 2009. URL https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.232.1231&rep=rep1&type=pdf.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 632–642, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1075. URL https://aclanthology.org/D15-1075.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1xMH1BtvB.

Ido Dagan, Oren Glickman, and Bernardo Magnini. The PASCAL recognising textual entailment challenge. In *the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*, 2005. URL https://kdd.cs.ksu.edu/Courses/Fall-2008/CIS798/Handouts/06-dagan05pascal.pdf.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 4171–4186, 2019.

William B. Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *the Third International Workshop on Paraphrasing (IWP2005)*, 2005. URL https://aclanthology.org/I05-5002.pdf.

Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1675–1685. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/du19c.html.

Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. In *Association for Computational Linguistics (ACL)*, pp. 3816–3830, 2021.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. The third PASCAL recognizing textual entailment challenge. In *the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, 2007. URL https://aclanthology.org/W07-1401.pdf.

Horace He and Richard Zou. functorch: Jax-like composable function transforms for pytorch. https://github.com/pytorch/functorch, 2021.

Mingzhen He, Fan He, Lei Shi, Xiaolin Huang, and Johan A. K. Suykens. Learning with asymmetric kernels: Least squares and feature interpretation, 2022. URL https://arxiv.org/abs/2202.01397.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004.

Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/5a4be1fa34e62bb8a6ec6b91d2462f5a-Paper.pdf.

William B Johnson. Extensions of lipschitz mappings into a hilbert space. *Contemp. Math.*, 26: 189–206, 1984.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association of Computational Linguistics (TACL)*, 2020.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=ryup8-WCW.

Zhiyuan Li, Srinadh Bhojanapalli, Manzil Zaheer, Sashank Reddi, and Sanjiv Kumar. Robust training of neural networks using scale invariant architectures. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 12656–12684. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/li22b.html.

Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han. Understanding the difficulty of training transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 5747–5763, Online, November 2020a. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.463. URL https://aclanthology.org/2020.emnlp-main.463.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Ro{bert}a: A robustly optimized {bert} pretraining approach, 2020b. URL https://openreview.net/forum?id=SyxS0T4tvS.

Robert Logan IV, Ivana Balazevic, Eric Wallace, Fabio Petroni, Sameer Singh, and Sebastian Riedel. Cutting down on prompts and parameters: Simple few-shot learning with language models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 2824–2835, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.222. URL https://aclanthology.org/2022.findings-acl.222.

Chao Ma, Lei Wu, and Weinan E. A qualitative study of the dynamic behavior for adaptive gradient algorithms. In *Proceedings of the 2nd Mathematical and Scientific Machine Learning Conference*, volume 145 of *Proceedings of Machine Learning Research*, pp. 671–692. PMLR, 16–19 Aug 2022. URL https://proceedings.mlr.press/v145/ma22a.html.

Sadhika Malladi, Kaifeng Lyu, Abhishek Panigrahi, and Sanjeev Arora. On the sdes and scaling rules for adaptive gradient algorithms, 2022. URL https://arxiv.org/abs/2205.10287.

Roman Novak, Jascha Sohl-Dickstein, and Samuel S Schoenholz. Fast finite width neural tangent kernel. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 17018–17044. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/novak22a.html.

Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Association for Computational Linguistics (ACL)*, 2004.

Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Association for Computational Linguistics (ACL)*, 2005.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2016. URL https://aclanthology.org/D16-1264/.

Nikunj Saunshi, Sadhika Malladi, and Sanjeev Arora. A mathematical exploration of why language models help solve downstream tasks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=vVjIW3sEc1s.

Timo Schick and Hinrich Schütze. Exploiting cloze-questions for few-shot text classification and natural language inference. In *European Chapter of the Association for Computational Linguistics (EACL)*, pp. 255–269, 2021.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2013. URL https://aclanthology.org/D13-1170.pdf.

Ellen M Voorhees and Dawn M Tice. Building a question answering test collection. In *the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, 2000.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://openreview.net/forum?id=rJ4km2R5t7.

Alexander Wei, Wei Hu, and Jacob Steinhardt. More than a toy: Random matrix models predict how real-world neural representations generalize. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 23549–23588. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/wei22a.html.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3), 2005.

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2018. URL https://aclanthology.org/N18-1101.pdf.

Blake Woodworth, Suriya Gunasekar, Jason D. Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. Kernel and rich regimes in overparametrized models. In *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pp. 3635–3673. PMLR, 09–12 Jul 2020. URL https://proceedings.mlr.press/v125/woodworth20a.html.

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

Greg Yang. Tensor programs ii: Neural tangent kernel for any architecture. *arXiv preprint arXiv:2006.14548*, 2020a.

Greg Yang. Tensor programs iii: Neural matrix laws. *arXiv preprint arXiv:2009.10685*, 2020b.

Greg Yang and Edward J Hu. Tensor programs iv: Feature learning in infinite-width neural networks. In *International Conference on Machine Learning*, pp. 11727–11737. PMLR, 2021.

Greg Yang and Etai Littwin. Tensor programs iib: Architectural universality of neural tangent kernel training dynamics. In *International Conference on Machine Learning*, pp. 11762–11772. PMLR, 2021.

Greg Yang, Edward J Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer. *arXiv preprint arXiv:2203.03466*, 2022.

Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank Reddi, Sanjiv Kumar, and Suvrit Sra. Why are adaptive methods good for attention models? In *Advances in Neural Information Processing Systems*, volume 33, pp. 15383–15393. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/b05b57f6add810d3b7490866d74c0053-Paper.pdf.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/250cf8b51c773f3f8dc8b4be867a9a02-Paper.pdf.

A Experimental Details

A.1 Datasets and Prompts

Dataset	C	#Train	#Test	Type	Prompt	
SST-2	2	67,349	872	sentiment	$<S_1>$ It was [MASK] .	{great, terrible}
SST-5	5	8,544	1,000	sentiment	$<S_1>$ It was [MASK] .	{great, good, okay, bad, terrible}
MR	2	8,662	1,000	sentiment	$<S_1>$ It was [MASK] .	{great, terrible}
CR	2	3,175	500	sentiment	$<S_1>$ It was [MASK] .	{great, terrible}
MPQA	2	8,606	1,000	opinion polarity	$<S_1>$ It was [MASK] .	{great, terrible}
Subj	2	8,000	1,000	subjectivity	$<S_1>$ This is [MASK] .	{subjective, objective}
TREC	6	5,452	500	question cls.	[MASK] : $<S_1>$	{Description, Expression, Entity, Human, Location, Number}
AG News	4	120,000	7,600	news topic	$<S_1>$ This article is about [MASK] news.	{world, sports, business, tech}
MNLI	3	392,702	1,000	NLI	$<S_1>$? [MASK] , $<S_2>$	{Yes, Maybe, No}
SNLI	3	549,367	1,000	NLI	$<S_1>$? [MASK] , $<S_2>$	{Yes, Maybe, No}
QNLI	2	104,743	1,000	NLI	$<S_1>$? [MASK] , $<S_2>$	{Yes, No}
RTE	2	2,490	277	NLI	$<S_1>$? [MASK] , $<S_2>$	{Yes, No}
MRPC	2	3,668	408	paraphrase	$<S_1>$ [MASK] , $<S_2>$	{Yes, No}
QQP	2	363,846	1,000	paraphrase	$<S_1>$ [MASK] , $<S_2>$	{Yes, No}

Table 2: The statistics and prompts of the datasets we used in our experiments. The choices of prompts are adapted from Gao et al. (2021) and include a template and a set of label words that can fill in the [MASK] token. $<S_1>$ and $<S_2>$ refer to the first and the second (if any) input sentence.

Table 2 shows the set of downstream tasks, which are adapted from Gao et al. (2021). We consider 8 single sentence classification datasets (SST-2 (Socher et al., 2013), SST-5 (Socher et al., 2013), MR (Pang & Lee, 2005), CR (Hu & Liu, 2004), MPQA (Wiebe et al., 2005), Subj (Pang & Lee, 2004), TREC (Voorhees & Tice, 2000), and AG News (Zhang et al., 2015)), and 6 sentence pair datasets (MNLI (Williams et al., 2018), SNLI (Bowman et al., 2015), QNLI (Rajpurkar et al., 2016), RTE (Dagan et al., 2005; Bar Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009), MRPC (Dolan & Brockett, 2005) and QQP[3]. Our datasets represent 6/8 datasets of the GLUE benchmark Wang et al. (2019) (SST-2, MNLI, QNLI, RTE, MRPC, QQP).

In contrast to Gao et al. (2021), we add AG News as an additional multi-label classification task, and make two modifications to the test sets. First, we split CR into 500 test examples and 3,175 training examples to ensure enough training examples for our 512-shot experiments and secondly, we limit the test sizes to 1,000 examples to speed up kernel evaluations.

To generate k-shot few-shot datasets, the original training data is used to randomly sample k examples per label for training and another, separate k examples per label for the validation set. Unless otherwise

[3] https://www.quora.com/q/quoradata/

stated, we usually run experiments over 5 seeds of few-shot data sets. We directly use the 'manual' prompt templates and label words proposed by Gao et al. (2021), which are reproduced in Table 2. We do include any demonstrations in our prompts.

A.2 Computing the Kernel

We use functorch (He & Zou, 2021) to compute the eNTK for RoBERTa-base (125M parameters), using a mix of backward-mode auto-differentiation for computing the jacobians and forward-mode auto-differentiation for computing jacobian-vector products (Novak et al., 2022) . Note that $\mathcal{K}^{(\text{SignGD})}$ cannot be computed via jacobian-vector products and requires substantially more memory and run-time in practice.

A.3 Solving the Kernel

In the standard NTK setting, the initial output of the model $f(\cdot;\theta_0)$ contains no information about solving the task, because θ_0 is a random initiaization. However, in the prompted FT setting, we expect the pre-trained model to be able to solve the downstream task well even before any fine-tuning occurs (see Table 8). So, we add the pre-trained model's output to the output from the kernel. Furthermore, we run a grid search over scaling the labels in order to take advantage of any pre-existing knowledge the model has about the downstream task. In particular, the kernel regression is based on the ℓ_2 distance to the ground truth one-hot vector, but the pre-trained model outputs the logits which will be used for cross-entropy loss. Scaling the one-hot vector by f_0 helps align its scaling with the logits. Our hyperparameter grid for f_0 can be found in Table 3, where ∞ corresponds to not using the pre-trained model logits when solving the kernel.

Solving Multi-Class Tasks There are several options for how to solve C-way classification tasks ($C > 2$). We perform the most general one, which scales with C^2. Each logit is treated as an independent output of the network, essentially scaling the size N of the original dataset by a factor of C. With CN examples, the kernel now has shape $CN \times CN$. The labels are also scaled up to treat the multi-class problem as many binary classification problems. Solving the multi-class task this way allows the kernel regression model to view relationships between different logits.

Symmetric Kernel Given a symmetric kernel $\mathcal{K} \in \mathbb{R}^{N\times N}$, we solve the kernel regression problem. In particular, we use the representer theorem to write that the empirical risk minimizer of the loss can be expressed as a linear combination of the kernel features computed on the train set.

$$h^*(\cdot) = \arg\min_{h\in\mathcal{H}_\mathcal{K}} \frac{1}{N}\sum_{i=1}^{N} \ell(h(x_i), y_i) \quad \leftrightarrow \quad h^*(\cdot) = \sum_{i=1}^{N} \alpha_i \mathcal{K}(\cdot, x_i)$$

for a given loss function ℓ. The symmetric SignGD and SGD kernels train α_i via gradient descent to minimize a regularized logistic loss on the downstream task. We search over a grid of regularization strengths chosen proportional to $\|\mathcal{K}\|_{\text{op}}$, see Table 3. For a test input x, the kernel outputs the prediction $h(x) = \sum_i \alpha_i \mathcal{K}(x, x_i)$.

Asymmetric Kernel We write how to solve the kernel regression problem with an asymmetric kernel, developed in He et al. (2022), here. Consider the augmented linear system:

$$\begin{bmatrix} I/\gamma & H \\ H^\top & I/\gamma \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

where $H_{ij} = y_i \phi_s(x_i)^\top \phi_t(x_j) y_j$ with ϕ_s and ϕ_t as the two different feature maps and y_i as the label for the ith example. In our setting, ϕ_s is the gradient of the datapoint, and ϕ_t is the sign of the gradient. Define ω^* and ν^* as

$$\omega^* = \sum_i \beta_i^* y_i \phi_t(x_i)$$

$$\nu^* = \sum_i \alpha_i^* y_i \phi_s(x_i)$$

Solving this system yields two discriminant functions:

$$f_s(x) = K(x, X)(\beta^* \odot Y)$$
$$f_t(x) = K(X, x)(\alpha^* \odot Y)$$

where $K(x_i, x_j) = \langle \phi_s(x_i), \phi_t(x_j) \rangle$.

We can thus create one discriminant function as $cf_s(x) + (1-c)f_t(x)$ where $c \in [0, 1]$ is some hyperparameter. When $\phi_s = \phi_t$, we see that $f_s = f_t$ and we reduce to the standard kernel problem (though with repeated equations). Note that per He et al. (2022), this system is only meaningful in terms of stationary points when training α and β using the least squares loss.

We now leverage some specific knowledge about the NTK setting. In particular, we know that we should only use f_s as the predictor in order to correctly represent a new test input in the kernel analog for SignGD.

Experiment	Hyperparameters	Values
SGD FT	Batch size	$\{2, 4, 8\}$ $\times$
	Learning rate	$\{1e{-}4, 5e{-}4, 1e{-}3, 5e{-}3, 1e{-}2\}$
SGD-LoRA FT	Batch size	$\{4, 16\}$ $\times$
	Learning rate	$\{1e{-}4, 1e{-}3, 1e{-}2\}$ $\times$
	$(r_{LoRA}, \alpha_{LoRA})$	$\{(8, 16)\}$
Adam FT	Batch size	$\{2, 4, 8\}$ $\times$
	Learning rate	$\{1e{-}5, 2e{-}5, 5e{-}5\}$
Adam-LoRA FT	Batch size	$\{4, 16\}$ $\times$
	Learning rate	$\{1e{-}5, 4e{-}5, 4e{-}4\}$
	$(r_{LoRA}, \alpha_{LoRA})$	$\{(8, 16)\}$
$\mathcal{K}^{(\text{SGD})}$, $\mathcal{K}^{(\text{SignGD})}$	Kernel regularization	$\{0, 0.001, 0.01, 0.1, 1\}$ $\times$
	f_0 scaling	$\{10, 100, 1000, 10000, \infty\}$
$\mathcal{K}^{(\text{A-SignGD})}$	Kernel regularization	$\{0, 0.001, 0.01, 0.1, 1\}$ $\times$
	f_0 scaling	$\{10, 100, 1000, 10000, \infty\}$ $\times$
	Kernel γ	$\{0.01, 0.1, 1, 10\}$ $\times$
	Kernel c	$\{1\}$

Table 3: The hyperparameter grids used in our experiments.

Hyperparameters and Implementation We follow Gao et al. (2021) in using the few-shot validation set to search over hyperparameters and finding the best hyperparameter per few-shot dataset. We use value ranges given by Gao et al. (2021) and Hu et al. (2021), and search over a wider range of values for SGD. Table 3 shows the hyperparameter grids for fine-tuning and the kernel method.

Gao et al. (2021) train for 1000 steps in the 16-shot setting, and validate the performance every 100 steps to take the best checkpoints. As we consider varying values of k, we use the formula of training for $32kC$ steps and validating every $4kC$ steps, where C is the number of classes in the dataset. This gives a comparable number of training and validation steps for binary tasks in the 16-shot setting.

B Discussion of Experiments

B.1 Experimental Insights

Prompting is critical for eNTK to match FT performance. We measure the eNTK performance in the standard and prompt-based FT settings across SST-2, MR, CR, QNLI, QQP and RTE. (Figure 1). In the standard FT setting, $\mathcal{K}^{(\text{SGD})}$ and SGD-FT demonstrate a gap of up to 16% absolute on tasks that exhibit only a 3% gap in the prompt-based setting (ablations in Appendix B.6). These results agree with our theoretical analysis that tasks must use a meaningful prompt in order to induce kernel behavior (Definition 3.6).

SGD performs comparably to Adam in prompt-based FT. Table 4 shows that Adam and SGD perform within 4% absolute of each other when using a prompt, suggesting that known difficulties in optimizing transformers with SGD (Li et al., 2022; Zhang et al., 2020; Liu et al., 2020a) do not play

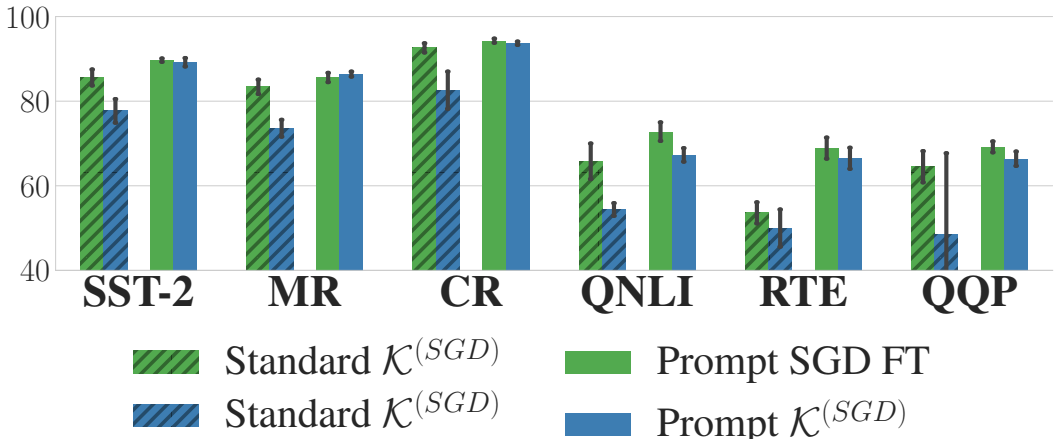

Figure 1: Comparing SGD-FT and $\mathcal{K}^{(\text{SGD})}$ performance in the standard and the prompt-based settings (Section 2) suggests that kernel behavior (Definition 2.2) can only arise when using a prompt. In standard FT, we initialize the new classification head (i.e., Γ) using the linear probing solution. Performance is measured by accuracy, or by F1 in QQP, and is averaged over 5 random seeds for $k = 64$.

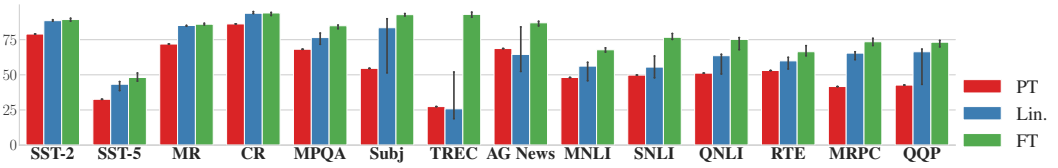

Figure 2: Accuracies of zero-shot pre-trained model (PT), linearized model (Lin., see Definition 2.2) and fine-tuned model (FT). Tasks that induce the Linearization property of kernel behavior (Definition 2.2) will show that Lin. performance recovers a substantial amount of the Adam FT performance. We plot the median and range of the test accuracies across 5 seeds and data splits for $k = 64$.

a substantial role during prompt-based FT. Indeed, we expect that the benefit of Adam over SGD is reduced when the task is simple enough to induce kernel behavior.

Prompt-based eNTK matches FT in most tasks. We compare SGD-FT to $\mathcal{K}^{(\text{SGD})}$ and Adam-FT to $\mathcal{K}^{(\text{A-SignGD})}$ in Table 4. We observe that for 10 out of 14 tasks, the kernel analog can achieve accuracy within 10% of the corresponding FT performance for $k = 16$ and $k = 64$.

B.2 Measuring Kernel Behavior

The eNTK can often solve the task comparably to fine-tuning (Table 4), suggesting that these tasks may induce kernel behavior (Definition 2.2). We take additional measurements to provide further empirical evidence that FT can be *described* by kernel behavior.

The Linearization property holds for all tasks the eNTK can solve. If FT exhibits kernel behavior (Definition 2.2), then the function after FT should be close to the first order Taylor expansion around the pre-trained model:

$$f(\xi; \theta_{\text{FT}}) \approx f(\xi; \theta_{\text{PT}}) + \langle \nabla f(\xi; \theta_{\text{PT}}), \theta_{\text{FT}} - \theta_{\text{PT}} \rangle$$

where θ_{PT} is the model parameters after pre-training, θ_{FT} is the model parameters after fine-tuning on the downstream task, and ξ is sampled from the test set. Figure 2 summarizes how this linearized model performs in comparison to the pre-trained and fine-tuned models.

Pre-trained models perform significantly better than random on many single-sentence downstream tasks (e.g., SST-2, MR, and CR) but close to random on most sentence-pair tasks (e.g., QNLI, RTE,

k-shot	Method	SST-2	SST-5	MR	CR	MPQA	Subj	TREC	AG News
16	SGD-FT	**89.0**$_{(1.5)}$	**44.6**$_{(1.4)}$	83.2$_{(2.4)}$	**93.3**$_{(0.2)}$	**83.3**$_{(1.3)}$	88.5$_{(2.6)}$	**80.3**$_{(7.2)}$	**84.2**$_{(1.1)}$
	$\mathcal{K}^{(\text{SGD})}$	88.3$_{(0.3)}$	43.6$_{(2.2)}$	**84.7**$_{(1.5)}$	93.2$_{(0.9)}$	76.4$_{(2.7)}$	**88.6**$_{(1.3)}$	56.0$_{(9.2)}$	82.1$_{(2.0)}$
	Adam-FT	**88.3**$_{(1.2)}$	**45.4**$_{(2.6)}$	81.3$_{(6.1)}$	93.0$_{(1.6)}$	**82.8**$_{(2.2)}$	87.4$_{(2.1)}$	**79.6**$_{(6.1)}$	**84.0**$_{(1.6)}$
	$\mathcal{K}^{(\text{SignGD})}$	**88.3**$_{(0.5)}$	42.2$_{(3.9)}$	84.3$_{(1.5)}$	**93.7**$_{(0.5)}$	76.7$_{(3.3)}$	**89.2**$_{(2.0)}$	58.1$_{(6.5)}$	82.3$_{(1.6)}$
	$\mathcal{K}^{(\text{A-SignGD})}$	**88.3**$_{(0.4)}$	43.7$_{(1.7)}$	**84.9**$_{(1.1)}$	93.4$_{(0.5)}$	74.6$_{(3.5)}$	88.6$_{(1.8)}$	22.7$_{(2.8)}$	83.6$_{(1.0)}$
64	SGD-FT	**89.7**$_{(0.4)}$	45.8$_{(2.1)}$	85.6$_{(1.1)}$	**94.3**$_{(0.5)}$	**84.8**$_{(0.8)}$	**92.9**$_{(0.5)}$	**93.2**$_{(1.0)}$	**86.8**$_{(0.7)}$
	$\mathcal{K}^{(\text{SGD})}$	89.2$_{(1.0)}$	**46.0**$_{(1.3)}$	**86.4**$_{(0.6)}$	93.7$_{(0.4)}$	81.2$_{(0.9)}$	91.4$_{(0.7)}$	77.8$_{(2.3)}$	85.6$_{(0.7)}$
	Adam-FT	**89.3**$_{(0.7)}$	48.5$_{(2.0)}$	**86.0**$_{(0.4)}$	93.7$_{(0.8)}$	**84.6**$_{(0.9)}$	**92.7**$_{(0.6)}$	**92.6**$_{(1.3)}$	**86.8**$_{(1.1)}$
	$\mathcal{K}^{(\text{SignGD})}$	89.1$_{(0.5)}$	**49.1**$_{(1.6)}$	85.6$_{(1.0)}$	93.9$_{(0.2)}$	79.0$_{(5.8)}$	92.4$_{(0.5)}$	82.0$_{(1.4)}$	85.9$_{(0.7)}$
	$\mathcal{K}^{(\text{A-SignGD})}$	88.9$_{(0.9)}$	43.6$_{(2.2)}$	85.6$_{(1.0)}$	**94.0**$_{(0.3)}$	81.8$_{(1.1)}$	91.8$_{(1.1)}$	21.0$_{(4.3)}$	86.2$_{(0.3)}$

(a) Single-sentence tasks

k-shot	Method	MNLI	SNLI	QNLI	RTE	MRPC	QQP
16	SGD-FT	**59.2**$_{(2.7)}$	**65.7**$_{(2.7)}$	**62.1**$_{(3.1)}$	**60.0**$_{(5.5)}$	**73.9**$_{(2.7)}$	**62.1**$_{(2.3)}$
	$\mathcal{K}^{(\text{SGD})}$	53.0$_{(3.0)}$	57.8$_{(2.3)}$	60.1$_{(3.3)}$	**60.0**$_{(4.7)}$	73.4$_{(5.6)}$	58.2$_{(0.9)}$
	Adam-FT	**56.8**$_{(2.9)}$	**64.6**$_{(4.1)}$	**63.1**$_{(3.5)}$	57.6$_{(6.3)}$	**77.6**$_{(3.1)}$	**61.8**$_{(4.5)}$
	$\mathcal{K}^{(\text{SignGD})}$	53.8$_{(1.2)}$	54.9$_{(2.7)}$	59.5$_{(3.1)}$	55.4$_{(4.2)}$	75.6$_{(1.2)}$	60.7$_{(2.2)}$
	$\mathcal{K}^{(\text{A-SignGD})}$	51.9$_{(4.0)}$	54.9$_{(3.1)}$	56.0$_{(1.9)}$	**59.8**$_{(4.0)}$	75.2$_{(2.6)}$	59.4$_{(2.0)}$
64	SGD-FT	**68.7**$_{(1.7)}$	**77.3**$_{(0.9)}$	**72.8**$_{(2.2)}$	**68.9**$_{(2.5)}$	**82.8**$_{(1.2)}$	**69.2**$_{(1.3)}$
	$\mathcal{K}^{(\text{SGD})}$	60.4$_{(1.8)}$	65.5$_{(1.6)}$	67.3$_{(1.6)}$	66.5$_{(2.5)}$	79.2$_{(2.5)}$	66.4$_{(1.7)}$
	Adam-FT	**67.9**$_{(1.0)}$	**76.9**$_{(1.4)}$	**74.2**$_{(3.2)}$	**67.3**$_{(2.7)}$	**80.9**$_{(1.2)}$	**69.8**$_{(0.6)}$
	$\mathcal{K}^{(\text{SignGD})}$	60.8$_{(1.7)}$	64.1$_{(2.3)}$	65.4$_{(1.7)}$	63.8$_{(1.8)}$	77.4$_{(2.3)}$	63.7$_{(4.4)}$
	$\mathcal{K}^{(\text{A-SignGD})}$	58.5$_{(1.7)}$	66.8$_{(1.1)}$	66.5$_{(1.1)}$	63.8$_{(2.2)}$	77.3$_{(2.0)}$	66.1$_{(3.4)}$

(b) Sentence-pair tasks

Table 4: Prompt-based FT and prompt-based eNTK performance with different formulas on the LM-BFF test set (Gao et al., 2021). The kernel analog performs comparably to FT on many tasks but fails if the prompt is poorly designed (i.e., MPQA, TREC, SNLI, and MNLI). Performance is measure by average test accuracy over 5 k-shot splits for all tasks except MRPC and QQP, where it is F1.

MRPC, and QQP).[4] The linearized model recovers more than 50% amount of the improvement from FT for all tasks the eNTK could solve (Table 4).

The Fixed Features property holds for all tasks the eNTK can solve. We empirically test if the Fixed Features property (Definition 2.2) holds for tasks that the eNTK can solve by measuring the relative distance between $\mathcal{K}^{(\text{SGD})}$ computed before and after FT (Table 6). Tasks that the eNTK can solve exhibit low (i.e., less than 1) distances, indicating the Fixed Features property likely holds.

Entailment tasks exhibit anomalous optimization characteristics. Although pre-trained models perform much better than random on MNLI and SNLI, we find that the eNTK cannot solve these tasks very well (Table 4 and Figure 2). Similarly, although the pre-trained model demonstrates near-random performance on QNLI and RTE, we find that the eNTK can solve these tasks. Moreover, although QNLI and RTE could sometimes be solved by the eNTK, the results suggest they do not induce the Linearization property of kernel behavior very strongly. Altogether, these findings suggest a deeper mystery around the fine-tuning dynamics when solving entailment tasks.

[4]Subj, MNLI, and SNLI are outliers to this trend.

B.3 TASKS WITHOUT KERNEL BEHAVIOR

TREC, MNLI, SNLI, and MPQA do not induce kernel behavior.[5] Our theoretical analysis suggests that when the prompt and label words do not format the task as a subcase of pre-training, then the task will not be natural in the infinite-width limit (Definition 3.6) and hence will not induce kernel behavior. Considering the prompt templates shown in Appendix A.1, we suggest that the TREC prompt (simply a colon) provides insufficient signal to the model to perform question type classification. For MNLI and SNLI, we observe that connecting the premise and hypothesis with the label word "Maybe" for neutral examples results in ungrammatical sentences. Furthermore, the prompt used for sentiment and polarity tasks is designed to follow a complete sentence or substantial phrase, so it is less natural when used with MPQA examples, which are often only one or two words. See Appendix B.6 for ablations.

B.4 ADDITIONAL EXPERIMENTS

Tables 7 and 8 contain the numerical results corresponding to Figures 1 and 2 respectively, and also report results for $k = 64$. Table 6 measures how well the Fixed Features property holds for different tasks. A smaller value suggests that the condition for kernel behavior (Definition 2.2) is satisfied more strongly.

B.5 SOLVABLE TASK EXPERIMENTS

We run a preliminary empirical test to verify if various tasks are solvable in the infinite-width limit (see Definition 3.6). Intuitively, the assumption states that wider models (with all other architecture and pre-training hyperparameters fixed) will solve the downstream task better in a zero-shot fashion, and in the limit, an infinitely wide model will solve the task perfectly. The cheap empirical test involves measuring the average output derivative χ of the loss w.r.t. the model output (see Definition 2.1 for a definition of χ) over the entire dataset for two models of different widths. We note that our paper uses RoBERTa-base ($n = 768$) for experiments, so a natural choice for a wider model would be RoBERTa-large ($n = 1024$). However, RoBERTa-large is also deeper than RoBERTa-base, and indeed, in general, it is difficult to find two publicly available pre-trained models with different widths and fixed depth. We nevertheless present the table of χ values for several downstream tasks measured on RoBERTa-base and RoBERTa-large in Table 5.

Model size	**SST-2**	**MR**	**CR**	**MPQA**	**Subj**	**QNLI**	**RTE**	**MRPC**	**QQP**
Base ($n = 768$)	0.32	0.32	0.26	0.38	0.43	0.48	0.48	0.56	0.49
Large ($n = 1024$)	0.32	0.25	0.25	0.40	0.46	0.48	0.47	0.52	0.52

Table 5: We measure the average output derivative (Definition 2.1) in the prompt-based FT setting for RoBERTa-base and RoBERTa-large.

B.6 ROBUSTNESS TO CHOICE OF PROMPT

We explore different choices of prompt and label words in Table 9. When using the results of the prompt and label search from Gao et al. (2021), we find that the kernel approximation matches fine-tuning well,. However, the choice of prompt does matter and $\hat{\mathcal{K}}^{(\text{SGD})}$ performs poorly with the minimal "null prompts" from Logan IV et al. (2022) on sentiment classification datasets, where the prompt is merely "$<S_1>$ `[MASK]`" and the label words remain {great, terrible}. We hypothesize this failure is because the task is no longer solvable in the infinite width limit (Definition 3.6).

[5]The eNTK can solve AG News although FT does not exhibit kernel behavior. This finding suggests that our theory holds for the prompt used with AG News, but the grid search over learning rates results in FT that does not exhibit kernel behavior. In particular, the success of the eNTK suggests the task can be solved with a very small learning rate, but the FT trajectory achieving the best performance uses a larger learning rate and thus exhibits more complex dynamics.

k-shot	SST-2	SST-5	MR	CR	MPQA	Subj	TREC	AG News
16	$0.44_{(0.14)}$	$0.51_{(0.17)}$	$0.41_{(0.13)}$	$0.43_{(0.16)}$	$0.41_{(0.08)}$	$0.43_{(0.28)}$	$2.02_{(0.85)}$	$1.87_{(1.42)}$
64	$0.41_{(0.07)}$	$0.73_{(0.07)}$	$0.45_{(0.22)}$	$0.52_{(0.08)}$	$0.42_{(0.13)}$	$0.60_{(0.17)}$	$1.59_{(0.23)}$	$1.45_{(0.53)}$

(a) Single-sentence tasks.

k-shot	MNLI	SNLI	QNLI	RTE	MRPC	QQP
16	$0.67_{(0.16)}$	$0.66_{(0.07)}$	$0.48_{(0.10)}$	$0.59_{(0.22)}$	$0.63_{(0.21)}$	$0.46_{(0.10)}$
64	$0.86_{(0.16)}$	$0.65_{(0.03)}$	$0.58_{(0.12)}$	$0.57_{(0.06)}$	$0.86_{(0.08)}$	$0.56_{(0.10)}$

(b) Sentence-pair tasks.

Table 6: Average element-wise relative distance of $\mathcal{K}^{(\mathrm{SGD})}$ computed on the pre-trained and best model fine-tuned with Adam. A smaller value indicates a higher likelihood that the Fixed Features property of kernel behavior (Definition 2.2) holds when performing fine-tuning. Distances are averaged across 5 seeds for each value of k and measured on the LM-BFF test set (Gao et al., 2021).

k-shot	Prompt	Method	SST-2	MR	CR	QNLI	RTE	QQP
16	Prompt	Adam-FT	$88.3_{(1.2)}$	$81.3_{(6.1)}$	$93.0_{(1.6)}$	$63.1_{(3.5)}$	$57.6_{(6.3)}$	$61.8_{(4.5)}$
		SGD-FT	$89.0_{(1.5)}$	$83.2_{(2.4)}$	$93.3_{(0.2)}$	$62.1_{(3.1)}$	$60.0_{(5.5)}$	$62.1_{(2.3)}$
		Kernel SGD	$88.3_{(0.3)}$	$84.7_{(1.5)}$	$93.2_{(0.9)}$	$60.1_{(3.3)}$	$60.0_{(4.7)}$	$58.2_{(0.9)}$
	Standard	Adam-FT	$78.1_{(4.2)}$	$69.0_{(6.0)}$	$83.9_{(5.2)}$	$56.7_{(3.6)}$	$51.1_{(3.8)}$	$58.5_{(5.6)}$
		SGD-FT	$77.6_{(4.3)}$	$64.8_{(5.2)}$	$86.6_{(2.6)}$	$51.6_{(1.8)}$	$50.2_{(2.1)}$	$57.0_{(4.6)}$
		Kernel SGD	$62.3_{(6.4)}$	$61.2_{(4.0)}$	$67.5_{(2.3)}$	$50.3_{(1.4)}$	$48.7_{(2.0)}$	$50.8_{(5.0)}$
64	Prompt	Adam-FT	$89.3_{(0.7)}$	$86.0_{(0.4)}$	$93.7_{(0.8)}$	$74.2_{(3.2)}$	$67.3_{(2.7)}$	$69.8_{(0.6)}$
		SGD-FT	$89.7_{(0.4)}$	$85.6_{(1.1)}$	$94.3_{(0.5)}$	$72.8_{(2.2)}$	$68.9_{(2.5)}$	$69.2_{(1.3)}$
		Kernel SGD	$89.2_{(1.0)}$	$86.4_{(0.6)}$	$93.7_{(0.4)}$	$67.3_{(1.6)}$	$66.5_{(2.5)}$	$66.4_{(1.7)}$
	Standard	Adam-FT	$86.1_{(1.2)}$	$83.9_{(1.9)}$	$92.6_{(1.0)}$	$71.5_{(4.5)}$	$53.9_{(4.2)}$	$65.0_{(3.6)}$
		SGD-FT	$85.6_{(1.9)}$	$83.4_{(1.7)}$	$92.6_{(1.1)}$	$65.8_{(4.2)}$	$53.6_{(2.5)}$	$64.5_{(3.7)}$
		Kernel SGD	$77.7_{(2.8)}$	$73.6_{(2.0)}$	$82.6_{(4.4)}$	$54.4_{(1.5)}$	$50.0_{(4.4)}$	$48.4_{(19.3)}$
512	Prompt	Adam-FT	$90.7_{(1.2)}$	$88.6_{(0.6)}$	$94.8_{(0.3)}$	$82.6_{(1.2)}$	$75.4_{(3.0)}$	$75.7_{(0.9)}$
		SGD-FT	$91.8_{(0.4)}$	$89.0_{(0.6)}$	$94.6_{(0.6)}$	$82.5_{(0.5)}$	$76.1_{(1.2)}$	$76.0_{(1.0)}$
		Kernel SGD	$90.2_{(0.4)}$	$88.2_{(0.5)}$	$94.3_{(0.2)}$	$75.1_{(0.7)}$	$71.4_{(2.0)}$	$70.7_{(1.7)}$
	Standard	Adam-FT	$91.4_{(0.7)}$	$88.4_{(0.8)}$	$94.1_{(0.6)}$	$82.2_{(0.3)}$	$72.6_{(1.9)}$	$76.1_{(0.8)}$
		SGD-FT	$91.0_{(0.4)}$	$88.7_{(0.5)}$	$95.6_{(0.4)}$	$81.6_{(1.5)}$	$72.0_{(2.2)}$	$75.8_{(0.6)}$
		Kernel SGD	$84.9_{(2.3)}$	$83.0_{(1.3)}$	$92.1_{(0.7)}$	$68.0_{(1.2)}$	$56.5_{(3.1)}$	$68.9_{(1.1)}$

Table 7: Fine-tuning performance in the standard FT setting, where the contextual embedding of the `[CLS]` token is used for classification, and the prompt-based FT setting, where a prompt is added and the embedding for the `[MASK]` token is used (see Section 2). In standard FT, we initialize the new classification head (i.e., Γ) using the linear probing solution. This table gives the figures in Figure 1, and also relates SGD fine-tuning performance to the more common fine-tuning with Adam.

k-shot	SST-2 Lin.	SST-2 FT	SST-5 Lin.	SST-5 FT	MR Lin.	MR FT	CR Lin.	CR FT
0	—— *79.0* ——		—— *32.6* ——		—— *71.9* ——		—— *86.2* ——	
16	$87.5_{(1.3)}$	$88.3_{(1.2)}$	$41.8_{(4.1)}$	$45.4_{(2.6)}$	$84.3_{(1.8)}$	$81.3_{(6.1)}$	$93.3_{(0.6)}$	$93.0_{(1.6)}$
64	$88.6_{(0.4)}$	$89.3_{(0.7)}$	$42.9_{(2.2)}$	$48.5_{(2.0)}$	$85.0_{(0.2)}$	$86.0_{(0.4)}$	$94.0_{(0.5)}$	$93.7_{(0.8)}$

k-shot	MQPA Lin.	MQPA FT	Subj Lin.	Subj FT	TREC Lin.	TREC FT	AG News Lin.	AG News FT
0	—— *68.2* ——		—— *54.6* ——		—— *27.4* ——		—— *68.7* ——	
16	$75.6_{(3.1)}$	$82.8_{(2.2)}$	$82.9_{(4.7)}$	$87.4_{(2.1)}$	$30.4_{(7.2)}$	$79.6_{(6.1)}$	$57.8_{(18.3)}$	$84.0_{(1.6)}$
64	$75.6_{(2.3)}$	$85.0_{(0.2)}$	$78.9_{(14.0)}$	$92.7_{(0.6)}$	$31.2_{(13.0)}$	$92.6_{(1.3)}$	$67.5_{(12.2)}$	$86.8_{(1.1)}$

(a) Single-sentence tasks.

k-shot	MNLI Lin.	MNLI FT	SNLI Lin.	SNLI FT	QNLI Lin.	QNLI FT
0	—— *48.1* ——		—— *49.8* ——		—— *51.2* ——	
16	$43.6_{(6.4)}$	$56.8_{(2.9)}$	$47.2_{(9.3)}$	$64.6_{(4.1)}$	$57.5_{(2.3)}$	$63.1_{(3.5)}$
64	$55.1_{(4.8)}$	$67.9_{(1.0)}$	$56.9_{(5.7)}$	$76.9_{(1.4)}$	$60.4_{(5.3)}$	$74.2_{(3.2)}$

k-shot	RTE Lin.	RTE FT	MRPC Lin.	MRPC FT	QQP Lin.	QQP FT
0	—— *53.1* ——		—— *41.7* ——		—— *42.7* ——	
16	$55.4_{(6.7)}$	$57.6_{(6.3)}$	$57.7_{(11.6)}$	$68.9_{(2.4)}$	$57.5_{(10.3)}$	$61.7_{(6.5)}$
64	$59.6_{(2.9)}$	$67.3_{(2.7)}$	$64.2_{(2.2)}$	$73.8_{(1.7)}$	$61.7_{(9.4)}$	$72.7_{(1.8)}$

(b) Sentence-pair tasks.

Table 8: Accuracies of pre-trained model (0-shot), linearized model (Lin., see Definition 2.2) and fine-tuned model (FT). Tasks that exhibit the Linearization property of kernel behavior (Definition 2.2) during fine-tuning will show that Lin. performance recovers a substantial amount of the gain in performance achieved by performing fine-tuning with Adam. Accuracies are averaged across 5 fine-tuning seeds for each value of k and measured on the test set. This table corresponds to the bar chart in Figure 2.

k-shot	Prompt + label format	Method	SST-2	MR	CR	QNLI	RTE	QQP
16	Manual (Gao et al., 2021)	Adam-FT	$88.3_{(1.2)}$	$81.3_{(6.1)}$	$93.0_{(1.6)}$	$63.1_{(3.5)}$	$57.6_{(6.3)}$	$61.8_{(4.5)}$
		SGD-FT	$89.0_{(1.5)}$	$83.2_{(2.4)}$	$93.3_{(0.2)}$	$62.1_{(3.1)}$	$60.0_{(5.5)}$	$62.1_{(2.3)}$
		$\mathcal{K}^{(\mathrm{SGD})}$	$88.3_{(0.3)}$	$84.7_{(1.5)}$	$93.2_{(0.9)}$	$60.1_{(3.3)}$	$60.0_{(4.7)}$	$58.2_{(0.9)}$
	Prompt + label search (Gao et al., 2021)	Adam-FT	$88.1_{(0.8)}$	$81.6_{(3.8)}$	$92.8_{(0.4)}$	$56.3_{(3.8)}$	$58.6_{(4.6)}$	$58.6_{(4.5)}$
		SGD-FT	$89.2_{(1.2)}$	$80.1_{(1.8)}$	$93.2_{(0.5)}$	$58.7_{(4.8)}$	$61.6_{(2.6)}$	$59.0_{(1.4)}$
		$\mathcal{K}^{(\mathrm{SGD})}$	$88.6_{(1.1)}$	$78.5_{(1.2)}$	$93.5_{(0.7)}$	$56.7_{(1.7)}$	$57.4_{(5.5)}$	$60.2_{(2.0)}$
	Null prompts (Logan IV et al., 2022)	Adam-FT	$87.6_{(0.9)}$	$82.6_{(0.6)}$	$92.8_{(0.6)}$	$59.0_{(2.9)}$	$56.4_{(4.7)}$	$57.5_{(5.2)}$
		SGD-FT	$88.1_{(0.7)}$	$82.8_{(3.6)}$	$93.4_{(0.7)}$	$59.0_{(3.4)}$	$54.1_{(1.6)}$	$57.6_{(5.5)}$
		$\mathcal{K}^{(\mathrm{SGD})}$	$78.3_{(4.3)}$	$78.7_{(1.8)}$	$91.7_{(0.8)}$	$55.8_{(2.7)}$	$55.5_{(2.3)}$	$57.4_{(1.8)}$

Table 9: We experiment with different prompt formats and label words: using the top result of an automatic prompt search performed on RoBERTa-large (Table E.1 in Gao et al. (2021)); and minimal null prompts (Table A3, Logan IV et al. (2022)), which add no additional text to the prompt. We find that our observations are robust to the choice of prompt, with the exception of the more unnatural "null prompts" on sentiment tasks (SST-2, MR, CR), which show a substantial gap between $\mathcal{K}^{(\mathrm{SGD})}$ and fine-tuning. We report F1 for QQP and accuracy otherwise, and average the metrics over 5 seeds.

C EFFICACY OF SUBSPACE-BASED FINE-TUNING METHODS

We study subspace-based fine-tuning methods, which apply updates to only a low-dimensional subspace of the high-dimensional model parameter space during fine-tuning. Although theoretical analysis of these methods seems complex, the kernel view admits a simple interpretation. We directly apply the Johnson-Lindenstrauss (JL) lemma in Johnson (1984), which guarantees inner product preservation under random projections, to suggest why LoRA (Hu et al., 2021) works. Similar analysis yields results on parameter-subspace FT methods used to study intrinsic dimension (Li et al. (2018); Aghajanyan et al. (2021), see Appendix E).

Definition C.1 ($\mathcal{A}$-LoRA FT (Hu et al., 2021))**.** Let $\mathcal{A}$ be a gradient-based optimization algorithm. For every weight matrix $W \in \mathbb{R}^{m\times n}$, choose $k \ll m$ and initialize $B \in \mathbb{R}^{m\times k}$ with i.i.d. zero-mean Gaussian values and $A \in \mathbb{R}^{k\times n}$ as 0. Set the weight to be $W + BA$. To fine-tune, fix W at its pre-trained value and train only A and B using $\mathcal{A}$.

We show that if SGD FT exhibits kernel behavior, then so does SGD-LoRA FT, and SGD-LoRA FT using a sufficiently large k does not modify the kernel or the dynamics.

Theorem C.2 (Informal version of Theorem E.5)**.** *Let $\mathcal{K}^{(SGD)}$ be the kernel analog (Definition 2.3) to SGD FT and $\mathcal{K}^{(SGD)}_{LoRA}$ be the kernel analog to SGD-LoRA FT on a downstream task Ξ with N examples. Then, with high probabililty, $\mathcal{K}^{(SGD)}_{LoRA}(i,j) \approx \mathcal{K}^{(SGD)}(i,j)$ for all $i,j \in [N]$.*

Proof sketch. Consider an individual layer in the network and a task input $\xi \in \Xi$. LoRA causes $\nabla_B f(\xi;\theta)$ to be a random projection of $\nabla_W f(\xi;\theta)$, where ∇_B denotes the gradient with respect to B, and $\nabla_A f(\xi;\theta) = 0$ since B is initialized to zero. The rest of the proof follows from applying JL to all input pairs ξ, ξ' to show the inner product (and thus, the kernel entry) is preserved. □

Remark C.3. Theorem C.2 states that the kernel analog of SGD-FT is unchanged by LoRA in both prompt-based and standard FT. However, the theorem only provides an explanation for the success of $\mathcal{A}$-LoRA FT when $\mathcal{A}$ FT exhibits kernel behavior. Therefore, as per Sections 3.2 and 4, we consider this theorem to only be meaningful when considering prompt-based SGD and prompt-based LoRA-SGD.

Table 11 verifies that prompt-based SGD FT and SGD-LoRA FT achieve similar performance on several tasks, and $\mathcal{K}^{(\text{SGD})}_{\text{LoRA}}$ achieves performance similar to $\mathcal{K}^{(\text{SGD})}$.

D KERNEL BEHAVIOR AND THE PARAMETRIZATION

Neural network training can exhibit either kernel behavior or feature learning behavior. These were described in Woodworth et al. (2020) as the lazy regime and active regime, respectively, when training from a random initialization. Kernel behavior provides a tractable tool to study the training of neural networks, but it is not believed to be a complete description of practical deep learning settings. In particular, kernel behavior implies the feature (i.e., gradient) of the neural networks remains unchanged in the overparameterized setting, which is not true in practical pre-training of large models.

Yang & Hu (2021) showed how the initialization variance, multiplier, and learning rate for each parameter can move training from the kernel behavior to the feature learning behavior. They further developed the Maximal Update Parametrization (abbreviated MUP or μP) where every parameter is updated maximally (in terms of scaling with width) while keeping the network stable. Yang et al. (2022) then extends μP to Transformers with Adam optimization, and showed empirically that for pre-training of large language models using μP, the optimal hyperparameters remain the same when increasing width. It allows more comprehensive hyperparameter searches on a smaller model and direct transfer of the resulting optimal hyperparameters to the larger model, resulting in markedly improved pre-training performance.

This section discusses two of our formal results: Theorems 3.3 and 3.7. In general, we consider the overparameterized setting in which the width of the network goes to infinity. Additionally, we assume that when initializing a weight matrix of the model, each entry of the matrix is drawn from i.i.d. Gaussian distribution. In particular, we model a pre-trained model as a non-random initialization that arose from training starting at a random initialization. We use Tensor Programs (Yang, 2020b) for our formal results.

This section is organized as follows. In Appendix D.1, we introduce the basic notation and ideas around Tensor Programs as well as the assumptions we need to make in order for an infinite-width limit to be interesting to study. Then, Appendix D.2 gives the formal proof for the kernel analog to SignGD (Theorem 3.3). In Appendix D.3, we provide a formal proof of how fine-tuning can exhibit kernel behavior (Theorem 3.7). The proof relies heavily on Tensor Programs, so we additionally provide a more accessible and intuitive sketch on linear networks in Appendix D.5. Finally, in **??**, we show that standard FT can exhibit kernel behavior when studying the infinite-width limit, though experiments in Table 7 suggest otherwise. This contradiction between theory and practice suggests that realistic finite-width networks are too far from infinite width ones when performing standard FT. Nevertheless, wider models may exhibit kernel behavior when performing standard FT.

D.1 PRELIMINARIES

Notations Let $\xi \in \mathbb{R}^{d_{in}}$ be the input of the network. Let n be the hidden dimension of the network and d_{out} be the output dimension of the network. We define the network as a function of the following form:

$$f(\xi; \{U^i\}_i, \{W^j\}_j, V) = V^\top h(\xi; \{U^i\}_i, \{W^j\}_j),$$

where ξ is the input, $U^i \in \mathbb{R}^{n \times d_{in}}$ are the input weight matrices, $W^j \in \mathbb{R}^{n \times n}$ are hidden weight matrices, $V \in \mathbb{R}^{n \times d_{out}}$ is the output weight matrix, and $h(\xi; \{U^i\}_i, \{W^j\}_j) \in \mathbb{R}^n$ is the input of last layer (readout layer). [6] We write $\mathcal{M}$ as the set of weight matrices, i.e., $\mathcal{M} = \{U^i\}_i \cup \{W^j\}_j \cup \{V\}$. For $M \in \mathcal{M}$, let $\nabla_M f(\xi)$ be the gradient of f w.r.t. M at input ξ.

To simplify the notation, we assume $d_{in} = 1$ in this section. We will note when an extension to $d_{in} > 1$ requires a non-trivial step. For any weight matrix $M \in \mathcal{M}$, let γ_M be the multiplier of M, such that M is multiplied by γ_M before performing matrix multiplication. Let η_M be the learning rate of the weight M. Let σ_M^2 be the variance of entries of M at initialization, so each entry of M is drawn $\mathcal{N}(0, \sigma_M^2)$ independently. Since our focus is the prompt-based fine-tuning, we assume no change is made to the network at the beginning of fine-tuning, and the learning rates for pre-training and fine-tuning are the same unless otherwise noted.

[6] We are able to describe transformers (without weight tying) in the definition. The bias can be regarded as input weights assuming there is a coordinate in ξ that is always 1.

Because we are considering the infinite-width limit, $f(\xi; \{U^i\}_i, \{W^j\}_j, V)$ actually represents a series of increasingly wide networks $\{f^n(\xi; \{U^{i,n}\}_i, \{W^{j,n}\}_j, V^n)\}_{n>0}$ of the same architecture, but f^n has a hidden dimension n. We use the notation f to include the model architecture, the training optimizer of the model, and $\gamma_M, \eta_M, \sigma_M$ for every weight matrix M in the model.

Let M_t be the weight matrix at time step t of training. If the network is pre-trained, we let M_{-1} be the weight matrix before pre-training, and M_0 be the parameters right after pre-training. Let $\Delta M_t = M_t - M_{t-1}$ be the change each training step induces. Let f_t be the network at step t that

$$f_t(\xi) = f(\xi; \{U_t^i\}_i, \{W_t^j\}_j, V_t).$$

Let ξ_t, y_t be the training input and target at step t, and let the loss function at step t be $\ell(f_{t-1}(\xi_t), y_t)$. For ease of notation, we often absorb y_t into ℓ and denote $\ell_t(f_{t-1}(\xi_t)) \triangleq \ell(f_{t-1}(\xi_t), y_t)$. Let $\chi_t = \ell_t'(f_{t-1}(\xi_t))$ be the derivative of the loss function, as defined in Definition 2.1. We assume ℓ_t'' (second derivative of ℓ_t) is bounded[7], which is satisfied when ℓ is mean square loss or cross entropy loss.

Big-O Notation For a series of scalar random variables $c = \{c^n\}_{n>0}$ and a function $e : \mathbb{N} \to \mathbb{R}$, we say $c = \Theta(e(n))$ if there exist A, B such that for sufficiently large n, $|c^n| \in [Ae(n), Be(n)]$ almost surely. For a series of vector random variables $x = \{x^n\}_{n>0}$, we say that x is coordinate-wise $\Theta(n^a)$, or $x = \Theta(e(n))$ if this series of scalar random variables $\{\|x^n\|_2/\sqrt{n}\}_{n>0}$ is $\Theta(e(n))$. Similarly for the notation $O(e(n))$, $\Omega(e(n))$, and $o(e(n))$. For convenience, we assume every $e(n)$ in this section is equal to n^a for some a.

Tensor Programs We refer reader to see Section 7 of Yang & Hu (2021) for detailed explanation and full definition of Tensor Programs. Here, we provide a simple overview of Tensor Programs:

Definition D.1 (Definition 7.1 of Yang & Hu (2021))**.** A Tensor Program is a sequence of $\mathbb{R}^n$-vectors and $\mathbb{R}$-scalars inductively generated via one of the following ways from an initial set $\mathcal{C}$ of random scalars, $\mathcal{V}$ of random $\mathbb{R}^n$ vectors, and a set $\mathcal{W}$ of random $\mathbb{R}^{n\times n}$ matrices.

MatMul Given $W \in \mathbb{R}^{n\times n}$ and $x \in \mathbb{R}^n$, we can generate $Wx \in \mathbb{R}^n$ or $W^\top x \in \mathbb{R}^n$.

Nonlin Given $\phi : \mathbb{R}^k \times \mathbb{R}^l \to \mathbb{R}$, previous scalar $\theta_1, \ldots, \theta_l \in \mathbb{R}$ and vector $x^1, \ldots, x^k \in \mathbb{R}^n$, we can generate a new vector

$$\phi(x^1, \ldots, x^k; \theta_1, \ldots, \theta_l) \in \mathbb{R}^n$$

where $\phi(-; \theta_1, \ldots, \theta_l)$ applies coordinate-wise to each "α-slice " $(x_\alpha^1, \ldots, x_\alpha^k)$.

Moment Given the same setup as above, we can also generate a new scalar

$$\frac{1}{n}\sum_{\alpha=1}^{n} \phi(x_\alpha^1, \ldots, x_\alpha^k; \theta_1, \ldots, \theta_l) \in \mathbb{R}.$$

Yang (2019; 2020a); Yang & Littwin (2021); Yang et al. (2022) show that Tensor Programs can express the computation, SGD/Adam optimization, and the kernel of almost any general architecture.

The key result of the Tensor Programs is that we can represent the coordinates of any vector x in the Tensor Program with a random variable Z^x, and represent any scalar θ with a deterministic scalar $\mathring{\theta}$. There is a way to define all $\mathring{\theta}$ and Z^x correspond to the Tensor Program (cf. Definition 7.3 in Yang & Hu (2021)), and the Master Theorem of the Tensor Program shows that $\theta \to \mathring{\theta}$ when $n \to \infty$ (cf. Theorem 7.4 in Yang & Hu (2021)).

Although it is in general hard to compute Z^x and $\mathring{\theta}$, it allows us to reason about the scales of vectors in the training of a network.

[7]For C-way classification, the assumption is extended to its multivariate version: each entry of Hessian of ℓ_t is bounded.

Assumptions Related to Tensor Programs. Since we are studying the infinite width limit and using Tensor Programs as our framework, there are some mild assumptions that we need in order to apply Tensor Programs and results in Yang & Hu (2021).

Assumption D.2. We assume the network f satisfies the following

a) The forward pass of f in the infinite-width limit can be written as Tensor Programs.

b) The hidden vectors have $\Theta(1)$ coordinates at initialization.

c) The hidden vectors have $O(1)$ coordinates during training.

d) For any training scheme[8] and any constant t and any input ξ, $f_t(\xi) = O(1)$.

e) There exist a training scheme and some constant t and input ξ such that $f_t(\xi) - f_0(\xi) = \Theta(1)$.

f) The activation function of f is tanh or σ-gelu for a small enough σ (so it approximates ReLU), where

$$\sigma\text{-}gelu(x) = \frac{1}{2}x\mathrm{erf}(\sigma^{-1}x) + \sigma\frac{e^{-\sigma^{-2}x^2}}{2\sqrt{\pi}} + \frac{x}{2}.$$

Furthermore, we have two assumption on SignGD:

g) SignGD is approximated as the sign function being replaced with ϵ-sign for small enough ϵ when updating parameters, where $\epsilon\text{-sign}(x) = \frac{x}{|x|+\epsilon}$ is smoothed version of sign. We assume using different ϵ when computing the sign of $\nabla_M f$, so that ϵ for $\nabla_M f$ match the maximum scale of $\nabla_M f$.

h) The ratio between the learning rate of SignGD in prompt-based fine-tuning and the learning rate of pre-training matches the maximum χ after pre-training. That is, we assume $\eta_M = \Theta(\eta_M^{\mathrm{PT}} \cdot \chi_{\max})$ where η_M^{PT} is learning rate of pre-training for SignGD, and $\chi_{\max} = \max_{(\xi,y)\in\Xi} \chi(\xi, y, f_0)$.

b), c), d) and e) in Assumption D.2 together recover the definition of nontrivial stable network in Yang & Hu (2021). b) and c) ensure that the pre-activations in the network are not too large, so that activation functions (e.g., tanh) are not trivialized to always output ± 1. b) ensures that the pre-activations in the network are not too small at initialization, so the activation function is not trivialized to its first-order Taylor expansion. d) ensures the network output is bounded. e) ensures that the network is not frozen during training (i.e., learning can occur).

f) and g) in Assumption D.2 assures all non-linear functions that appear in the Tensor Programs is pseudo-Lipschitz, which is required for the Master Theorem of Tensor Programs. g) also assures that ϵ-sign is not trivialize to 0 or sign when $\nabla_M f \neq \Theta(1)$.

h) in Assumption D.2 assures when $\chi = o(1)$, updates of SignGD in fine-tuning is not of bigger scale than SGD. It is also observed in practice that the optimal learning rate for fine-tuning is smaller than the learning rate for pre-training.

D.2 SignGD Kernel Derivation

Definition D.3 (Formal Definition of Kernel Behavior)**.** We say that this network training process demonstrates *kernel behavior* if the following properties are satisfied.

1. *Linearization*: The change of the network can be approximated by its first order Taylor expansion, i.e.,

$$\lim_{n\to\infty} \frac{f_t(\xi) - f_{t-1}(\xi)}{\chi_{\max}} = \lim_{n\to\infty} \sum_{M\in\mathcal{M}} \left\langle \nabla_M f_{t-1}(\xi), \frac{\Delta M_t}{\chi_{\max}} \right\rangle;$$

where $\chi_{\max} = \max_{(\xi,y)\in\Xi} \chi(\xi, y, f_0)$, Ξ is the training dataset.

[8]Training scheme means a sequence of training examples $\{(\xi_t, y_t\}_{t>0}$, and loss function $\ell(f_t(\xi_t), y_t)$.

2. *Fixed Features*: The gradients at step t are approximately the same as before training, i.e.,

$$\forall M \in \mathcal{M}, \lim_{n\to\infty} \frac{\|\nabla_M f_t(\xi) - \nabla_M f_0(\xi)\|_2^2}{\max_{\xi'} \|\nabla_M f_0(\xi')\|_2^2} = 0.$$

Note that we define Linearization with both LHS and RHS divided by $\chi_{\max}$ so it is meaningful for the case of $\chi = o(1)$. We do the same thing in the following theorem.

Theorem D.4 (SignGD Kernel)**.** *If SignGD training of f demonstrates kernel behavior, then under Assumption D.2,*

$$\lim_{n\to\infty} \frac{f_t(\xi) - f_{t-1}(\xi)}{\chi_{\max}} = \lim_{n\to\infty} \sum_{M\in\mathcal{M}} -\tilde{\eta}_M \left\langle \nabla_M f_0(\xi), \epsilon\text{-sign}(\nabla_M f_0(\xi_t)) \right\rangle,$$

where $\tilde{\eta}_M = \eta_M \operatorname{sign}(\chi_t)/\chi_{\max}$.

Note if $\eta_M = \eta$, the RHS of the equation above equals to

$$-\frac{\eta \operatorname{sign}(\chi_t)}{\chi_{\max}} \langle \nabla f_0(\xi), \epsilon\text{-sign}(\nabla f_0(\xi_t)) \rangle \approx -\frac{\eta \operatorname{sign}(\chi_t)}{\chi_{\max}} \mathcal{K}^{(\text{A-SignGD})}(\xi, \xi_t),$$

where the approximation comes from the difference between ϵ-sign and sign.

Proof. By the update rule of SignGD, $\frac{\Delta M_t}{\chi_{\max}} = -\tilde{\eta}_M \epsilon\text{-sign}(\nabla_M f_{t-1})$. It suffices to prove

$$\tilde{\eta}_M \left\langle \nabla_M f_t(\xi), \epsilon\text{-sign}(\nabla_M f_t(\xi_t)) \right\rangle = \tilde{\eta}_M \left\langle \nabla_M f_0(\xi), \epsilon\text{-sign}(\nabla_M f_0(\xi_t)) \right\rangle$$

when $n \to \infty$.

Since

$$\begin{aligned}
&\tilde{\eta}_M \left\langle \nabla_M f_t(\xi), \epsilon\text{-sign}(\nabla_M f_t(\xi_t)) \right\rangle - \tilde{\eta}_M \left\langle \nabla_M f_0(\xi), \epsilon\text{-sign}(\nabla_M f_0(\xi_t)) \right\rangle \\
&= \tilde{\eta}_M \left\langle \nabla_M f_t(\xi) - \nabla_M f_0(\xi), \epsilon\text{-sign}(\nabla_M f_t(\xi_t)) \right\rangle + && (4)\\
&\quad \tilde{\eta}_M \left\langle \nabla_M f_t(\xi), \epsilon\text{-sign}(\nabla_M f_t(\xi_t)) - \epsilon\text{-sign}(\nabla_M f_0(\xi_t)) \right\rangle + && (5)\\
&\quad \tilde{\eta}_M \left\langle \nabla_M f_t(\xi) - \nabla_M f_0(\xi), \epsilon\text{-sign}(\nabla_M f_t(\xi_t)) - \epsilon\text{-sign}(\nabla_M f_0(\xi_t)) \right\rangle, && (6)
\end{aligned}$$

we only need to prove Equations (4) to (6) are all 0 when $n \to \infty$.

Let $\xi^* = \arg\max_{\xi'} \|\nabla_M f_0(\xi')\|_2^2$ be the input of maximum gradient scale, then by Fixed Features, we have

$$\frac{\|\nabla_M f_t(\xi) - \nabla_M f_0(\xi)\|_2}{\|\nabla_M f_0(\xi^*)\|_2} = o(1). \tag{7}$$

Since $\epsilon\text{-sign}(x) - \epsilon\text{-sign}(y) \le |x - y|/\epsilon$,

$$\|\epsilon\text{-sign}(\nabla_M f_t(\xi)) - \epsilon\text{-sign}(\nabla_M f_0(\xi))\|_2 \le \|\nabla_M f_t(\xi) - \nabla_M f_0(\xi)\|_2/\epsilon. \tag{8}$$

Combined with $\|\nabla_M f_0(\xi^*)\|_2/\sqrt{N} = \Theta(\epsilon)$ (N is the number of entries of M, this is by g) of Assumption D.2), we have

$$\begin{aligned}
&\frac{\|\epsilon\text{-sign}(\nabla_M f_t(\xi)) - \epsilon\text{-sign}(\nabla_M f_0(\xi))\|_2}{\|\epsilon\text{-sign}(\nabla_M f_0(\xi^*))\|_2} \\
&\le \frac{\|\nabla_M f_t(\xi) - \nabla_M f_0(\xi)\|_2/\epsilon}{\|\epsilon\text{-sign}(\nabla_M f_0(\xi^*))\|_2} && \text{by eq. (8)}\\
&= \frac{\|\nabla_M f_t(\xi) - \nabla_M f_0(\xi)\|_2}{\|\nabla_M f_0(\xi^*)\|_2} \cdot \frac{\|\nabla_M f_0(\xi^*)\|_2/\sqrt{N}}{\epsilon\|\epsilon\text{-sign}(\nabla_M f_0(\xi^*))\|_2/\sqrt{N}} \\
&= \frac{\|\nabla_M f_t(\xi) - \nabla_M f_0(\xi)\|_2}{\|\nabla_M f_0(\xi^*)\|_2} \cdot \Theta(1) = o(1). && (9)
\end{aligned}$$

By d) in Assumption D.2, and consider the training scheme that sets $\xi_1 = \xi^*$ and the loss function ℓ_t so $\chi_1 = \Theta(1)$, then

$$\frac{f_1(\xi^*) - f_0(\xi^*)}{\chi_1} = -\frac{\eta_M \operatorname{sign}(\chi_1)}{\chi_1} \left\langle \nabla_M f_0(\xi^*), \epsilon\text{-sign}(\nabla_M f_0(\xi^*)) \right\rangle = O(1).$$

By h) in Assumption D.2, the scale of $\tilde{\eta}_M$ is identical across different training scheme, so we have

$$-\tilde{\eta}_M \langle \nabla_M f_0(\xi^*), \epsilon\text{-sign}(\nabla_M f_0(\xi^*)) \rangle = O(1).$$

And it is easy to see that $\tilde{\eta}_M \|\nabla_M f_0(\xi^*)\|_2 \|\epsilon\text{-sign}(\nabla_M f_0(\xi^*))\|_2$ has the same scale as $\tilde{\eta}_M \langle \nabla_M f_0(\xi^*), \epsilon\text{-sign}(\nabla_M f_0(\xi^*)) \rangle$, which is $O(1)$.

Given Equations (7) and (9), we are about to prove Equations (4) to (6) divided by $\tilde{\eta}_M \|\nabla_M f_0(\xi^*)\|_2 \|\epsilon\text{-sign}(\nabla_M f_0(\xi^*))\|_2$ are all 0 when $n \to \infty$. Provided that $\tilde{\eta}_M \|\nabla_M f_0(\xi^*)\|_2 \|\epsilon\text{-sign}(\nabla_M f_0(\xi^*))\|_2 = O(1)$, it will imply Equations (4) to (6) are all 0 when $n \to \infty$, thus conclude our whole proof.

For Equation (4),

$$\begin{aligned}
&\frac{\tilde{\eta}_M \langle \nabla_M f_t(\xi) - \nabla_M f_0(\xi), \epsilon\text{-sign}(\nabla_M f_t(\xi_t)) \rangle}{\tilde{\eta}_M \|\nabla_M f_0(\xi^*)\|_2 \|\epsilon\text{-sign}(\nabla_M f_0(\xi^*))\|_2} \\
\le &\frac{\|\nabla_M f_t(\xi) - \nabla_M f_0(\xi)\|_2 \|\epsilon\text{-sign}(\nabla_M f_t(\xi_t))\|_2}{\|\nabla_M f_0(\xi^*)\|_2 \|\epsilon\text{-sign}(\nabla_M f_0(\xi^*))\|_2} \\
= &\frac{\|\nabla_M f_t(\xi) - \nabla_M f_0(\xi)\|_2}{\|\nabla_M f_0(\xi^*)\|_2} = o(1). \qquad \text{by eq. (7)}
\end{aligned}$$

Similarly, for Equation (5),

$$\begin{aligned}
&\frac{\tilde{\eta}_M \langle \nabla_M f_t(\xi), \epsilon\text{-sign}(\nabla_M f_t(\xi_t)) - \epsilon\text{-sign}(\nabla_M f_0(\xi_t)) \rangle}{\tilde{\eta}_M \|\nabla_M f_0(\xi^*)\|_2 \|\epsilon\text{-sign}(\nabla_M f_0(\xi^*))\|_2} \\
\le &\frac{\|\epsilon\text{-sign}(\nabla_M f_t(\xi)) - \epsilon\text{-sign}(\nabla_M f_0(\xi))\|_2}{\|\epsilon\text{-sign}(\nabla_M f_0(\xi^*))\|_2} = o(1), \qquad \text{by eq. (9)}
\end{aligned}$$

and for Equation (6),

$$\begin{aligned}
&\frac{\tilde{\eta}_M \langle \nabla_M f_t(\xi) - \nabla_M f_0(\xi), \epsilon\text{-sign}(\nabla_M f_t(\xi_t)) - \epsilon\text{-sign}(\nabla_M f_0(\xi_t)) \rangle}{\tilde{\eta}_M \|\nabla_M f_0(\xi^*)\|_2 \|\epsilon\text{-sign}(\nabla_M f_0(\xi^*))\|_2} \\
\le &\frac{\|\epsilon\text{-sign}(\nabla_M f_t(\xi)) - \epsilon\text{-sign}(\nabla_M f_0(\xi))\|_2}{\|\epsilon\text{-sign}(\nabla_M f_0(\xi^*))\|_2} \cdot \frac{\|\nabla_M f_t(\xi) - \nabla_M f_0(\xi)\|_2}{\|\nabla_M f_0(\xi^*)\|_2} \\
= &\, o(1). \qquad \text{by eqs. (7) and (9)}
\end{aligned}$$

□

D.3 Prompt-based Fine-Tuning

Prompt-based fine-tuning uses the pre-trained network directly without substituting or adding any parameters. Therefore, without any additional assumptions, the behaviors of fine-tuning and pre-training are the same from the perspective of the Tensor Programs. We thus adopt the assumption that $\chi = o(1)$ before fine-tuning (Definition 3.6). Without the assumption, the fine-tuning of f will not exhibits kernel behavior if the pre-training is in feature learning regime. Intuitively, this assumption is believable because wider pre-trained networks can solve downstream tasks better. In this section, we prove that prompt-based fine-tuning exhibits kernel behavior when this assumption holds.

Theorem D.5. *If the downstream task Ξ is natural for network f, that is,*

$$\chi_{\max} \triangleq \max_{(\xi,y)\in\Xi} \chi(\xi, y, f_0) = o(1),$$

then under Assumption D.2, the fine-tuning of f exhibits kernel behavior (Definition D.3).

Below we provide a proof that is heavily based on Tensor Programs and the analysis in Yang & Hu (2021). For readers who are not familiar with Tensor Programs, we provide intuitive examples in the next few subsections, where we focus on a three-layer linear network parameterized with μP.

Proof. The high-level proof consists of two parts: 1) we prove after each step, the update of the function f is $O(\chi_t)$. Combined ℓ_t'' always bounded by some constant C, we can inductively prove

$\chi_t \le \chi(\xi_t, y_t, f_0) + C \cdot |f_{t-1}(\xi_t) - f_0(\xi_t)| = O(\chi_{\max})$ for all t. 2) Given $\chi_t = O(\chi_{\max}) = o(1)$, we show the fine-tuning exhibits kernel behavior.

We first prove the theorem under the assumption that the network is a multilayer perceptron and the optimizer is SGD, which is the same setting as Yang & Hu (2021). We will later extend this to more general cases.

Consider the following L-hidden-layer perceptron:

$$h^1(\xi) = U\xi,$$

and

$$x^l(\xi) = \phi(h^l(\xi)), \quad h^{l+1}(\xi) = W^{l+1}x^l(\xi), \text{ for } l = 1, \dots, L-1,$$

and

$$f(\xi) = Vx^L(\xi).$$

Following Yang & Hu (2021), we let the learning rate for every parameter equal to ηn^{-c}. Let $W^1 = U$ and $W^{L+1} = V$, and for $l = 1, \dots, L+1$, we parametrize W^l as $W^l = \gamma_l w^l$ for actual trainable parameter w^l, and we initialize each coordinate w^l i.i.d. from $\mathcal{N}(0, \sigma_l^2)$. The setting covers all possible parameterizations based on Lemma D.6. For convenience, we assume $\gamma_l = n^{-a_l}$ and $\sigma_l = n^{-b_l}$. Without loss of generality, we further assume that $\chi_{\max} = \Theta(n^{-d})$. Below, we will also inductively show $\chi_t = O(n^{-d})$ by showing $|f_{t+1} - f_t| = O(n^{-d})$.

By Theorem 3.3 of Yang & Hu (2021), stable network implies

$$r \triangleq \min(a_{L+1} + b_{L+1}, 2a_{L+1} + c) + c - 1 + \min_{l=1}^{L}[2a_l + \mathbb{I}(l=1)] \ge 0.$$

Also by Theorem 3.8 of Yang & Hu (2021), for nontrivial stable network (included in Assumption D.2), if $r > 0$ then there exists a kernel $\mathcal{K}$ such that

$$f_{t+1}(\xi) = f_t(\xi) - \eta\chi_t\mathcal{K}(\xi, \xi_t),$$

which is very close to our definition of kernel behavior. In fact, we will prove that they are equivalent in the fine-tuning case.

Since $\chi_t = O(n^{-d})$ for fine-tuning, it is equivalent to set the learning rate to ηn^{-c-d} and replace χ_t with $\hat{\chi}_t = n^d\chi_t = O(1)$. Formally, we are considering the following training scheme: at the pre-training stage, $r \ge 0$ (so it could demonstrate feature learning or kernel behavior); at the fine-tuning stage, c is increased to $c' \triangleq c + d > c$, thus, the corresponding r is increased to be strictly greater than 0. Therefore, it suggests kernel behavior with following caveats.

Do we handle the case of different learning rates during pre-training and fine-tuning? The answer is *effectively YES*, because the above scheme is equivalent to training from scratch with learning rate ηn^{c-d}. First of all, the scale of the update on W^l, h^l, x^l and f are all multiplied by n^{-d} when switching from the pre-training stage (ηn^{-c} learning rate) to the fine-tuning stage(ηn^{-c-d} learning rate). The scales are exactly the same as training from scratch with ηn^{-c-d} learning rate except b_{L+1} needs to be changed to $b'_{L+1} \triangleq \min(b_{L+1}, a_{L+1} + c)$. Note this change of b_{L+1} does not affect the fact that r is updated to $r' \triangleq r + d > 0$.

Does $r' > 0$ formally imply our definition of kernel behavior (Definition D.3)? The answer is *YES*. We first prove Fixed Features in Definition D.3. The gradient of matrix W^l is equal to outer product between $\nabla_{h^l} f$ (gradient w.r.t. h^l) and x^{l-1}. Let dh_t^l be the normalized gradient w.r.t. h^l at step t (so $dh_t^l = \Theta(1)$), and x_t^l be the x^l at step t ($x_t^l = \Theta(1)$ without normalization). It suffices to prove $dh_t^l - dh_0^l = O(1)$ and $x_t^l - x_0^l = o(1)$. The later was proved by Proposition H.27 of Yang & Hu (2021). To prove $dh_t^l - dh_0^l = O(1)$, we let dx_t^l be the the normalized gradient w.r.t. x^l at step t, and compute the scale of $dh_t^l - dh_{t-1}^l$ and $dx_t^l - dx_{t-1}^l$ inductively from $l = L$ to $l = 1$. We obtain that they both has the same scale of

$$n^{-\min(2a_{L+1}+c-a_{L+1}-b'_{L+1}, a_{L+1}+b_{L+1}+c'-1+\min_{m=l+1}^{L} 2a_m)} \le n^{-\min(0,r')} = 1,$$

the inequality is because $b'_{L+1} \le a_{L+1} + c$ and $r' \le a_{L+1} + b_{L+1} + c' - 1 + \min_{m=l+1}^{L} 2a_m$.

Second, we prove Linearization in Definition D.3. We need to first make a slight modification to the Tensor Program in Yang & Hu (2021), that is, changing the computation of $f_t(\xi) - f_{t-1}(\xi)$ to

$n^d(f_t(\xi) - f_{t-1}(\xi))$. By Theorem H.32 of Yang & Hu (2021) and its definition of Σ, we can show that

$$\begin{aligned}\lim_{n\to\infty} n^d(f_t(\xi) - f_{t-1}(\xi)) &= \lim_{n\to\infty} \sum_{l=1}^{L+1} \eta n^{-c} \frac{\chi_t}{n^{-d}} \langle \nabla_{W^l} f_{t-1}(\xi), \nabla_{W^l} f_{t-1}(\xi_t) \rangle \\ &= \lim_{n\to\infty} \sum_{l=1}^{L+1} \left\langle \nabla_{W^l} f_{t-1}(\xi), \frac{\Delta W_t^l}{n^{-d}} \right\rangle.\end{aligned}$$

This is exactly Linearization in Definition D.3 if we multiply $n^{-d}/\chi_{\max}$ on both side. Meanwhile, it also implies $f_t(\xi) - f_{t-1}(\xi) = O(n^{-d})$.

From SGD to SignGD. Since $\operatorname{sign}(xy) = \operatorname{sign}(x)\operatorname{sign}(y)$, the update of matrix W^l can still be written as outer product of two vectors, i.e., $\Delta W_t^l = \eta n^{-c-d} \operatorname{sign}(\chi_t) \operatorname{sign}(\nabla_{h^l} f_{t-1}) \otimes \operatorname{sign}(x_{t-1}^{l-1})$. After applying sign, the scale of vector changes. If the parametrization is the same, the scales of vectors using SignGD will be different from those using SGD. This can be easily resolved by changing learning rates for each parameter (as in Assumption D.2), so the scaling change brought by sign is corrected. Furthermore, as also mentioned in Assumption D.2, we need to approximate sign by a smoothed version ϵ-sign so the Master Theorem of Tensor Programs can still apply.

Extension to universal architectures. The theorem can apply to any network whose first forward pass can be written as Tensor Programs. Given this condition, the forward pass, backward pass, and kernel of any step can be written as Tensor Programs (Yang, 2020a;b). To analyse the scaling of the Tensor Program will need the following steps:

1. *Extension to general computation graph.* We can still inductively reason about the scale of preactivations and activations by the topological order of the computation graph; and similarly reason about the gradient by the reverse topological order.
2. *Extension to weight sharing.* We may use weights multiple times in a forward pass. The preactivations, activations and their gradients will not be affected. Only the update of a weight is now a sum of several vector outer product depending on the number of occurrence of the weight.

□

D.4 μP FOR SGD AND SIGNGD

In the following subsections, we provide more intuition for Theorem D.5. Although we consider all types of pre-trained models, we are mostly interested in models with feature learning behavior, because it is likely not true that gradients can be approximated as fixed throughout the entirety of *pre-training*. For pre-trained models with kernel behavior, it is obvious that fine-tuning with the same settings as pre-training (i.e., prompt-based FT) will also exhibit kernel behavior. Furthermore, Theorem H.17 of Yang & Hu (2021) proved that if the last layer is replaced with a freshly initialized layer (i.e., standard FT), fine-tuning from a pre-trained models with kernel behavior is the same as training on the downstream task from scratch.

Among all the pre-training schemes that exhibit feature learning behavior, μP is special because each parameter (except the last layer) can *on its own* push the model to perform feature learning. Therefore, to build an intuitive description of fine-tuning behavior, we assume that the model was pre-trained by μP. We note again that our main result *does not require* this assumption.

The formulation of μP contains three sets of hyperparameters: initial variance of M, multiplier of M and learning rate of M for $M \in \{U^i\}_i \cup \{W^j\}_j \cup \{V\}$. However, even if we restrict these three hyperparameters to be in the form of n^α, μP is not unique, because there is one degree of freedom for each weight according to the following lemma.

Lemma D.6 (Lemma J.1 of Yang et al. (2022))**.** *Consider a weight matrix M with learning rate C, initialized as $M \sim \mathcal{N}(0, B^2)$, and with a multiplier A. Then for any $\gamma > 0$, $f_t(\xi)$ stays fixed for all t and ξ if we set*

- $A \leftarrow A\gamma, B \leftarrow B/\gamma, C \leftarrow C/\gamma^2$ *if training with SGD.*
- $A \leftarrow A\gamma, B \leftarrow B/\gamma, C \leftarrow C/\gamma$ *if training with Adam.*

Note the conclusion about Adam in Lemma D.6 also extends to SignGD.

With Lemma D.6, we can always set the multiplier of any weight matrix M to be 1, which leave us only the initialization variance σ_M^2 and learning rate η_M. Furthermore, in terms of the scale at initialization and the scale of updates, μP for SGD and SignGD are entirely the same. The only difference would be learning rate. We provide details in Table 10 (recall M_{-1} is the weight M at initialization of pre-training, $\Delta M_0 = M_0 - M_{-1}$ is the overall change of weight in pre-training. We further assume $\chi_t = \Theta(n^{-d})$ for all t, thus $\eta_M n^d$ is the scale of learning rate for SignGD in pre-training).

coordinate-wise scale	$M = U^i$	$M = W^j$	$M = V$
M_{-1}	$\Theta(1)$	$\Theta(1/\sqrt{n})$	$\Theta(1/n)$
ΔM_0	$\Theta(1)$	$\Theta(1/n)$	$\Theta(1/n)$
η_M for SGD	$\Theta(n)$	$\Theta(1)$	$\Theta(1/n)$
$\eta_M \cdot n^d$ for SignGD/Adam	$\Theta(1)$	$\Theta(1/n)$	$\Theta(1/n)$

Table 10: Scales of initialization, update and learning rate for μP in pre-training.

Since we have different learning rate for different M, the kernel that we care is defined as

$$\mathcal{K}(\xi, \xi') = \sum_{M \in \mathcal{M}} \eta'_M \left\langle \nabla_W f(\xi), \phi(\nabla_W f(\xi')) \right\rangle,$$

where ϕ is identity if the algorithm is SGD, $\phi = \operatorname{sign}$ if the algorithm is SignGD, $\eta'_M = \eta_M$ for SGD, $\eta'_M = \eta_M n^d$ for SignGD. We use η'_M to keep $\mathcal{K}(\xi, \xi') = \Theta(1)$.

And we want to prove the dynamic of the network follows

$$\frac{f_t(\xi) - f_{t-1}(\xi)}{n^{-d}} \to -\tilde{\chi}_t \mathcal{K}(\xi, \xi_t) \quad \text{when } n \to \infty,$$

where $\tilde{\chi}_t = n^{-d}\chi_t$ for SGD, and $\tilde{\chi}_t = \operatorname{sign}(\chi_t)$ for SignGD. In any case, $\tilde{\chi}_t = \Theta(1)$.

D.5 Prompt-based Fine-Tuning: A Linear Example

As an intuitive example, we consider a three-layer linear network

$$f(\xi; U, W, V) = V^\top W U \xi.$$

For simplicity, we train the network with SGD, and freeze V so $\eta_V = 0$. Then we have $\nabla_U f = W^\top V \xi^\top$ and $\nabla_W f = V(U\xi)^\top$. We assume $|\langle \xi, \xi' \rangle| > 0$ for any ξ, ξ'.

In what follows, we will prove that for pre-training f cannot be written as the first-order Taylor expansion (i.e., it exhibits feature learning). Then we will prove that it is the opposite for fine-tuning. In fact, if we only look at one gradient step, the only higher order term equals to $\eta_W \eta_U \chi_t^2 \|V\|^2 \langle \xi_t, \xi \rangle f_{t-1}(\xi) = \Theta(\chi_t^2 f_{t-1}(\xi))$, where $f_{t-1}(\xi)$ is mostly $\Theta(1)$, χ_t is mostly $\Theta(1)$ in pre-training[9] and $o(1)$ in fine-tuning (by Definition 3.6).

Zero step (Pre-training) We model the pre-training of f as one step of training with $\chi_0 = \Theta(1)$. Then we have $\Delta U_0 = -\eta_U \chi_0 W_{-1}^\top V \xi_0^\top$, and $\Delta W_0 = -\eta_W \chi_0 V (U_{-1}\xi_0)^\top$. Since $W_{-1}^\top$ is independent from V, we have $W_{-1}^\top V = \Theta(1/n)$, thus $\Delta U_0 = \Theta(1)$ matching Table 10. On the other hand, it is obvious that $\Delta W_0 = \Theta(1/n)$ because $V = \Theta(1/n)$ and $U = \Theta(1)$, also matching Table 10.

[9] $f_t(\xi)$ is $\Theta(1)$ unless $t = -1$ or there are coincidental cancellations. χ_t is $\Theta(1)$ in pre-training until f memorizes the whole pre-training dataset when $n \to \infty$.

Then the function is now

$$\begin{aligned} f_0(\xi) &= V^\top (W_{-1} + \Delta W_0)(U_{-1} + \Delta U_0)\xi \\ &= V^\top (W_{-1} - \eta_W \chi_0 V (U_{-1}\xi_0)^\top)(U_{-1}\xi - \eta_U \chi_0 W_{-1}^\top V \langle \xi_0, \xi \rangle) \\ &= V^\top W_{-1} U_{-1}\xi - \eta_U \chi_0 \|W_{-1}^\top V\|_2^2 \langle \xi_0, \xi \rangle - \eta_W \chi_0 \|V\|^2 \langle U_{-1}\xi_0, U_{-1}\xi \rangle \\ &\quad + \eta_W \eta_U \chi_0^2 \|V\|^2 \langle \xi_0, \xi \rangle V^\top W_{-1} U_{-1}\xi. \end{aligned}$$

It is not difficult to see that $\eta_U \chi_0 \|W_{-1}^\top V\|_2^2 \langle \xi_0, \xi \rangle$, $\eta_W \chi_0 \|V\|^2 \langle U_{-1}\xi_0, U_{-1}\xi \rangle$, and $\eta_W \eta_U \chi_0^2 \|V\|^2 \langle \xi_0, \xi \rangle$ are all $\Theta(1)$. Unfortunately, here $V^\top W_{-1} U_{-1}\xi = f_{-1}(\xi) = o(1)$ in the infinite-width limit, but if we train one more step, it is easy to see that all four terms of f_0 is $\Theta(1)$. Therefore, pre-training with μP exhibits feature learning.

First step At the first step of fine-tuning, we have $\Delta U_1 = -\eta_U \chi_1 W_0^\top V \xi_1^\top$ and $\Delta W_1 = -\eta_W \chi_1 V (U_0 \xi_1)^\top$. The function can be written as

$$f_1(\xi) = V^\top (W_0 + \Delta W_1)(U_0 + \Delta U_1)\xi,$$

and

$$f_1(\xi) - f_0(\xi) = V^\top \Delta W_1 U_0 \xi + V^\top W_0 \Delta U_1 \xi + V^\top \Delta W_1 \Delta U_1 \xi. \tag{10}$$

Note that the sum of the first and second terms is exactly $-\chi_1 \mathcal{K}(\xi, \xi_1)$.

Plug in $\Delta W_1 = -\eta_W \chi_1 V (U_0 \xi_1)^\top$ into the first term of eq. (10),

$$V^\top \Delta W_1 U_0 \xi = -\eta_W \chi_1 V^\top V (U_0 \xi_1)^\top U_0 \xi = \Theta(\chi_1),$$

because

$$\begin{aligned} (U_0\xi_1)^\top U_0 \xi &= (U_{-1}\xi_1 + \Delta U_0 \xi_1)^\top (U_{-1}\xi + \Delta U_0 \xi) \\ &= \langle U_{-1}\xi_1, U_{-1}\xi \rangle - \eta_U \chi_0 \langle \xi_1, \xi_0 \rangle f_{-1}(\xi) - \eta_U \chi_0 \langle \xi, \xi_0 \rangle f_{-1}(\xi_1) + \|\Delta U_0\|^2 \langle \xi_1, \xi \rangle \\ &= \Theta(n). \end{aligned}$$

Plug in $\Delta U_1 = -\eta_U \chi_1 W_0^\top V \xi_1^\top$ into the second term of eq. (10), we have

$$V^\top W_0 \Delta U_1 \xi = -\eta_U \chi_1 V^\top W_0 W_0^\top V \xi_1^\top \xi = \Theta(\chi_1)$$

because

$$\begin{aligned} V^\top W_0 W_0^\top V &= \|(W_{-1} + \Delta W_0)^\top V, (W_{-1} + \Delta W_0)^\top V\|_2^2 \\ = \|W_{-1}^\top V\|_2^2 + \eta_W^2 \chi_0^2 \|V\|_2^4 \|U_{-1}\xi_0\|_2^2 - 2\eta_W \chi_0 \|V\|_2^2 f_{-1}(\xi_0) &= \Theta(1/n). \end{aligned}$$

The third term of eq. (10) equals

$$\eta_U \eta_W \chi_1^2 V^\top V (U_0 \xi_1)^\top W_0^\top V \xi_1^\top \xi = \eta_U \eta_W \chi_1^2 \|V\|^2 \langle \xi_1, \xi \rangle f_0(\xi_1) = \Theta(\chi_1^2),$$

because $f_0(\xi_1) = \Theta(1)$ unlike $f_{-1}(\xi)$ in the "zero step" analysis. Therefore, $\frac{f_1(\xi) - f_0(\xi)}{\chi_1} \to -\mathcal{K}(\xi, \xi_1)$.

Second step At the second step of fine-tuning, we have $\Delta U_2 = -\eta_U \chi_1 W_1^\top V \xi_2^\top$, and $\Delta W_2 = -\eta_W \chi_1 V (U_1 \xi_2)^\top$ and

$$f_2(\xi) - f_1(\xi) = V^\top \Delta W_2 U_1 \xi + V^\top W_1 \Delta U_2 \xi + V^\top \Delta W_2 \Delta U_2 \xi. \tag{11}$$

Assuming χ_2 and χ_1 share the same order, then when $n \to \infty$,

$$\begin{aligned} \frac{f_2(\xi) - f_1(\xi)}{\chi_2} &\to V^\top \Delta W_2 U_1 \xi / \chi_2 + V^\top W_1 \Delta U_2 \xi / \chi_2 \\ &= -\eta_W V^\top V (U_1 \xi_2)^\top U_1 \xi - \eta_U V^\top W_1 W_1^\top V \xi_2^\top \xi \\ &\to -\eta_W V^\top V (U_0 \xi_2)^\top U_0 \xi - \eta_U V^\top W_0 W_0^\top V \xi_2^\top \xi \\ &= -\mathcal{K}(\xi, \xi_2). \end{aligned}$$

tth step Same as the second step by noting ΔU_t, ΔW_t always have smaller order than ΔU_0 and ΔW_0.

D.6 LoRA FT Exhibits Kernel Behavior

Note Theorem D.5 works for any architecture, including LoRA. In order to apply the theorem to LoRA FT, we need to set the initialization and learning rate of the matrices A and B in LoRA correctly so that they satisfy Assumption D.2.

Here we provide a relatively straightforward way to accomplish this (assuming only intermediate layers use LoRA):

- Let $k = \alpha n$ where α is a small constant irrelevant to n.
- Let the initialization scale of A be $\Theta(1/\sqrt{n})$.
- Let the learning rate of A and B be $\Theta(1)$ for SGD, $\Theta(n^{-1-d})$ for SignGD / Adam.

In short words, the initialization and learning rate follows μP as in Table 10 by treating A and B as one of W^j. This setup easily generalizes to the case where U and V also use LoRA.

E Subspace-Based Fine-Tuning Methods

Experimental results related to LoRA FT are presented in Table 11. These results show that SGD-FT and SGD-LoRA FT perform similarly in the few-shot setting for many tasks, although the original experiments in Hu et al. (2021) focused on Adam. The closeness of $\mathcal{K}^{(\text{SGD})}$ and $\mathcal{K}^{(\text{SGD})}_{\text{LoRA}}$ to their respective fine-tuning methods suggests that FT and LoRA FT can be described by kernel dynamics. Moreover, we show that $\mathcal{K}^{(\text{SGD})}$ and $\mathcal{K}^{(\text{SGD})}_{\text{LoRA}}$ achieve similar performance to each other, providing empirical evidence for the claim in Theorem C.2 that LoRA preserves the kernel.

k-shot	Method	**SST-2**	**MR**	**CR**	**QNLI**	**RTE**	**QQP**
16	SGD-FT	$89.0_{(1.5)}$	$83.2_{(2.4)}$	$93.3_{(0.2)}$	$62.1_{(3.1)}$	$60.0_{(5.5)}$	$62.1_{(2.3)}$
	SGD-LoRA FT	$89.1_{(0.6)}$	$82.7_{(2.0)}$	$92.6_{(0.8)}$	$57.1_{(3.3)}$	$58.2_{(2.9)}$	$59.8_{(3.0)}$
	$\mathcal{K}^{(\text{SGD})}$	$88.3_{(0.3)}$	$84.7_{(1.5)}$	$93.2_{(0.9)}$	$60.1_{(3.3)}$	$60.0_{(4.7)}$	$58.2_{(0.9)}$
	$\mathcal{K}^{(\text{SGD})}_{\text{LoRA}}$	$88.1_{(0.4)}$	$84.9_{(1.4)}$	$93.1_{(1.0)}$	$59.4_{(3.7)}$	$56.2_{(5.8)}$	$58.2_{(3.2)}$
64	SGD-FT	$89.7_{(0.4)}$	$85.6_{(1.1)}$	$94.3_{(0.5)}$	$72.8_{(2.2)}$	$68.9_{(2.5)}$	$69.2_{(1.3)}$
	SGD-LoRA FT	$90.0_{(0.2)}$	$85.7_{(1.2)}$	$93.9_{(0.7)}$	$73.8_{(2.7)}$	$69.1_{(1.8)}$	$68.3_{(2.4)}$
	$\mathcal{K}^{(\text{SGD})}$	$89.2_{(1.0)}$	$86.4_{(0.6)}$	$93.7_{(0.4)}$	$67.3_{(1.6)}$	$66.5_{(2.5)}$	$66.4_{(1.7)}$
	$\mathcal{K}^{(\text{SGD})}_{\text{LoRA}}$	$89.2_{(0.7)}$	$85.7_{(1.5)}$	$93.6_{(0.4)}$	$66.0_{(1.6)}$	$63.5_{(3.5)}$	$63.9_{(4.5)}$

Table 11: Performance of prompt-based SGD FT and prompt-based SGD-LoRA FT, along with their kernel analogs $\mathcal{K}^{(\text{SGD})}$ and $\mathcal{K}^{(\text{SGD})}_{\text{LoRA}}$, on a subset of tasks. SGD FT and SGD-LoRA FT achieve comparable performance, and $\mathcal{K}^{(\text{SGD})}$ and $\mathcal{K}^{(\text{SGD})}_{\text{LoRA}}$ also achieve comparable performance to each other. We report F1 for QQP and accuracy otherwise, and average the metrics over 5 seeds. These experiments support Theorem C.2.

E.1 IntrinsicDimension FT

We discuss IntrinsicDimension FT (Li et al., 2018; Aghajanyan et al., 2021) here. When analyzed through the kernel, IntrinsicDimension FT and LoRA FT induce similar transformations in the optimization dynamics, but the former was originally proposed as a way to measure the difficulty of downstream tasks, and the latter was proposed as an alternative fine-tuning method.

Definition E.1 ($\mathcal{A}$-IntrinsicDimension FT (Li et al., 2018; Aghajanyan et al., 2021))**.** Let $\theta \in \mathbb{R}^M$ be the model parameters and fix a random projection $\Pi \in \mathbb{R}^{M\times k}$. Set θ to $\theta + \Pi\hat{\theta}$, where $\hat{\theta} \in \mathbb{R}^k$. To fine-tune, fix θ at its pre-trained value and only train $\hat{\theta}$.

We show a similar result for IntrinsicDimension FT as for LoRA FT: using a sufficiently large $k \geq \Theta(\log N/\epsilon^2)$ ensures that each element of the kernel is relatively unchanged.

Theorem E.2 (IntrinsicDimension FT preserves $\mathcal{K}^{(\text{SGD})}$)**.** *Let Π be a random matrix with each entry draw i.i.d from $\mathcal{N}(0, 1/k)$. Let $\mathcal{K}_{ID}^{(SGD)} \in \mathbb{R}^{N\times N}$ be the kernel analog to SGD-IntrinsicDimension FT (Definition E.1) on a downstream task Ξ. Additionally, assume $\mathcal{K}^{(SGD)}(i, j) \leq c$ for any $i, j \in [N]$. Then,*

$$\Pr\left[\exists i, j \in [N], |\mathcal{K}_{ID}^{(SGD)}(i, j) - \mathcal{K}^{(SGD)}(i, j)| \geq c\epsilon\right] \leq 4N^2 \exp(-(\epsilon^2 - \epsilon^3)k/4).$$

E.2 Proofs

A key step of the proof is to show that if $\mathcal{A}$ FT exhibits kernel behavior, then so does $\mathcal{A}$-LoRA FT. We show this step in Appendix D.6, since it invokes the Tensor Programs framework again. Now that we know FT follows kernel dynamics, we can move to showing how LoRA and IntrinsicDimension FT modify the kernel.

We restate the Johnson-Lindenstrauss lemma, which preserves inner products under random projection.

Lemma E.3 (Corollary of Johnson-Lindenstrauss, Johnson (1984))**.** *Let $u, v \in \mathbb{R}^d$ such that $\|u\|^2 \leq c$ and $\|v\|^2 \leq c$. Let $h(x) = \frac{1}{\sqrt{k}}Ax$, where $A \in \mathbb{R}^{k\times d}$ with each entry sampled i.i.d. from $\mathcal{N}(0, 1)$ or $\mathcal{U}(-1, 1)$. Then,*

$$\Pr[|u \cdot v - h(u) \cdot h(v)| \geq c\epsilon] \leq 4 \exp(-(\epsilon^2 - \epsilon^3)k/4)$$

Proof for Theorem E.2. Note $\nabla_{\hat{\theta}} f = \Pi^\top \nabla_\theta f$, and

$$\mathcal{K}_{\text{ID}}^{(\text{SGD})}(i, j) - \mathcal{K}^{(\text{SGD})}(i, j) = \langle \nabla_{\hat{\theta}} f(\xi_i; \theta), \nabla_{\hat{\theta}} f(\xi_j; \theta)\rangle - \langle \nabla_\theta f(\xi_i; \theta), \nabla_\theta f(\xi_j; \theta)\rangle.$$

The rest follows Lemma E.3 by setting $u = \nabla_\theta f(\xi_j; \theta)$, $v = \nabla_\theta f(\xi_i; \theta)$, and union bounding all i, j pairs. □

We can now look at LoRA (Hu et al., 2021) for a simple fully connected layer. The construction modifies each layer independently and only acts on fully connected layers, so this is the only part of the kernel that can change when parametrizing updates as in LoRA. For ease of notation, for any parameter or hidden vector w, we use dw to denote $\nabla_w f(\xi; \theta)$, $dw(i)$ to denote $\nabla_w f(\xi_i; \theta)$, and w_i denotes the resulting w when input is ξ_i.

Lemma E.4 (LoRA SGD Kernel)**.** *Let $h = Wx + BAx$ as defined in the paper, where $x \in \mathbb{R}^n$, $W \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{m\times k}$, and $A \in \mathbb{R}^{k\times n}$ with $k \ll n$. B is initialized to 0 and A is initialized with i.i.d. zero-mean Gaussian samples. SGD Training with LoRA (i.e., fixing W and allowing A and B to be updated) yields the kernel $\mathcal{K}_{LoRA}^{(SGD)}$, whereas full FT with SGD yields the kernel $\mathcal{K}$:*

$$\mathcal{K}_{LoRA}^{(SGD)} = dHdH^\top \odot (XA^\top AX^\top) \qquad \mathcal{K}^{(SGD)} = dHdH^\top \odot (XX^\top)$$

where $dH \in \mathbb{R}^{N\times m}$ has $dh(i)$ in the ith row and $X \in \mathbb{R}^{N\times d}$ has x_i in the ith row.

Proof. We start by noting the well-known fact that $dW = dh \otimes x$, where dh is the gradient to h and $\otimes$ is the cross product. Thus, $K = dHdH^\top \odot (XX^\top)$. In the LoRA setting, $dA = 0$ and $dB = dh \otimes Ax$. Because we are in the kernel setting, $B = 0$ and thus, $dA = 0$, throughout training. So,

$$\mathcal{K}_{\text{LoRA}}(i, j) = \langle dB(i), dB(j)\rangle = \langle dh(i), dh(j)\rangle\langle Ax_i, Ax_j\rangle.$$

Analogous reasoning yields

$$\mathcal{K}^{(\text{SGD})}(i, j) = \langle dh(i), dh(j)\rangle\langle x_i, x_j\rangle.$$

□

Theorem E.5 ($\mathcal{K}_{\text{LoRA}}^{(\text{SGD})}$ is likely not far from $\mathcal{K}^{(\text{SGD})}$)**.** *Let $\mathcal{K}_{LoRA}^{(SGD)} \in \mathbb{R}^{N\times N}$ and $\mathcal{K}^{(SGD)} \in \mathbb{R}^{N\times N}$ be defined as in Lemma E.4. Additionally, assume that $\|dh\|^2 \leq c$, $\|x\|^2 \leq c$ for any ξ in the downstream dataset. Then,*

$$\Pr\left[\exists i, j \in [N], |\mathcal{K}_{LoRA}^{(SGD)}(i, j) - \mathcal{K}^{(SGD)}(i, j)| \geq c^2\epsilon\right] \leq 4N^2 \exp(-(\epsilon^2 - \epsilon^3)k/4).$$

Proof. By Lemma E.4,

$$\begin{aligned}|\mathcal{K}_{\text{LoRA}}^{(\text{SGD})}(i,j) - \mathcal{K}^{(\text{SGD})}(i,j)| &= |\langle dh(i), dh(j)\rangle(\langle Ax_i, Ax_j\rangle - \langle x_i, x_j\rangle)| \\ &\leq c|\langle Ax_i, Ax_j\rangle - \langle x_i, x_j\rangle|.\end{aligned}$$

The rest of the proof follows from Lemma E.3 and union bound. □

Remark E.6. Theorem E.5 shows when $k \geq 20c^4 \log N/\epsilon^2$, with high probability, the difference between the two kernels is smaller than ϵ. Although Theorem E.5 focuses on a simple fully connected layer, the conclusion easily extends to the case where LoRA is applied L times in the model because LoRA components are independent of each other:

$$\Pr\left[\exists i, j \in [N], |\mathcal{K}_{\text{LoRA}}^{(\text{SGD})}(i,j) - \mathcal{K}^{(\text{SGD})}(i,j)| \geq Lc^2\epsilon\right] \leq 4N^2 \exp(-L(\epsilon^2 - \epsilon^3)k/4).$$

The requirement of k becomes $k \geq \Theta(Lc^4 \log N/\epsilon^2)$.