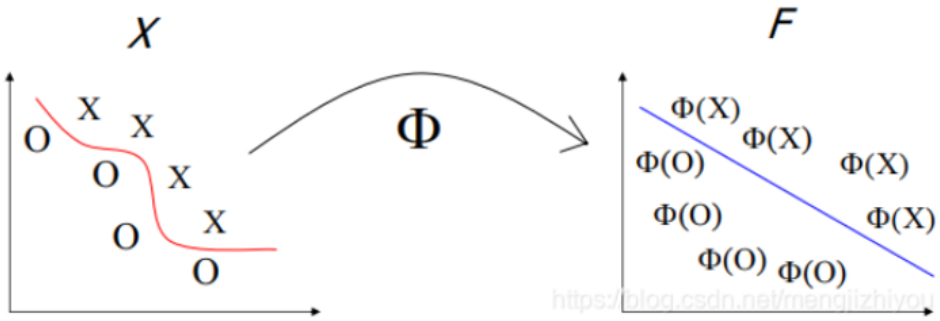
神经正切核(Neural Tangent Kernel，NTK)

核函数



核函数：给定两个样本输入x和y，返回它们在特征空间（高维）中对应向量的点积

$$K(x,y)=\langle\phi(x),\phi(y)\rangle$$

线性核：$K(x,y)=xy$

多项式核：

$$\begin{aligned}K(x,y)&=(xy+1)^2\\&=\begin{bmatrix}x^2\\\sqrt{2}x\\1\end{bmatrix}^T\cdot\begin{bmatrix}y^2\\\sqrt{2}y\\1\end{bmatrix}\\&=\langle\phi(x),\phi(y)\rangle\end{aligned}$$

高斯核：

$$\begin{aligned}K(x,y)&=e^{-\frac{(x-y)^2}{2}}\\&=\sum_{n=0}^{\infty}\left(e^{-\frac{x^2}{2}}\frac{x^n}{\sqrt{n!}}\right)\cdot\left(e^{-\frac{y^2}{2}}\frac{y^n}{\sqrt{n!}}\right)\\&=\langle\phi(x),\phi(y)\rangle\end{aligned}$$

神经正切核NTK (2018)

Neural Tangent Kernel: Convergence and Generalization in Neural Networks

Arthur Jacot
École Polytechnique Fédérale de Lausanne
arthur.jacot@netopera.net

Franck Gabriel
Imperial College London and École Polytechnique Fédérale de Lausanne
franckrgabriel@gmail.com

Clément Hongler
École Polytechnique Fédérale de Lausanne
clement.hongler@gmail.com

核心前提：对于一个无限宽的神经网络 $f(x;\theta)$，使用梯度下降训练，参数只需产生非常微小的变化就可以拟合训练数据（Lazy Training）

在初始化 θ_0 位置，可以对该网络进行一阶泰勒近似展开：
$f(x;\theta) \approx f(x;\theta_0) + \langle \nabla_\theta f(x;\theta_0), \theta - \theta_0 \rangle$

此时，网络输出 $f(x)$ 变成了关于参数 θ 的**线性函数**, 令特征向量 $\phi(x) = \nabla_\theta f(x;\theta_0)$, 那么网络更新的过程本质上就是在这个特征空间内作线性回归.

假设损失函数 $L = \frac{1}{2}\sum (f(x_i) - y_i)^2$, "训练"过程如下:

$$\frac{d\theta}{dt} = -\eta \frac{\partial L}{\partial \theta} = -\eta \sum_{i=1}^{n} (f(x_i) - y_i) \underbrace{\nabla_\theta f(x_i;\theta_0)}_{\text{训练样本的梯度}}$$

我们关心输出 $f(x)$ 如何随训练过程变化, 即

$$\begin{aligned}
\frac{df(x)}{dt} &= \underbrace{\nabla_\theta f(x;\theta_0)^T}_{\text{x 对参数的梯度}} \cdot \underbrace{\frac{d\theta}{dt}}_{\text{参数的变化速度}} \\
&= \nabla_\theta f(x;\theta_0)^T \cdot \left(-\eta \sum_{i=1}^{n} (f(x_i) - y_i) \nabla_\theta f(x_i;\theta_0) \right) \\
&= -\eta \sum_{i=1}^{n} \underbrace{\langle \nabla_\theta f(x;\theta_0), \nabla_\theta f(x_i;\theta_0) \rangle}_{\text{这就是 NTK!}} (f(x_i) - y_i) \\
&= -\eta \sum_{i=1}^{n} K(x, x_i)(f(x_i) - y_i)
\end{aligned}$$

对应的线性核函数 $K(x,y)=\langle\nabla_\theta f(x;\theta_0),\nabla_\theta f(y;\theta_0)\rangle$, 这就是神经正切核 (NTK), 它衡量了两个样本 x 和 y 在参数变化方向上的相似度, 并且在训练过程中保持不变!

这个微分方程存在解析解

$$f^*(x)=\underbrace{K(x,X_{train})}_{\text{相似度向量}}\cdot K(X_{train},X_{train})^{-1}\cdot\underbrace{Y_{train}}_{\text{训练标签}}$$

在有限宽度的实际网络中，原始网络的预测与线性化模型的预测之间也有很好的经验一致性（2019）：

Wide Neural Networks of Any Depth Evolve as Linear Models Under Gradient Descent

Jaehoon Lee*, Lechao Xiao*, Samuel S. Schoenholz, Yasaman Bahri
Roman Novak, Jascha Sohl-Dickstein, Jeffrey Pennington
Google Brain

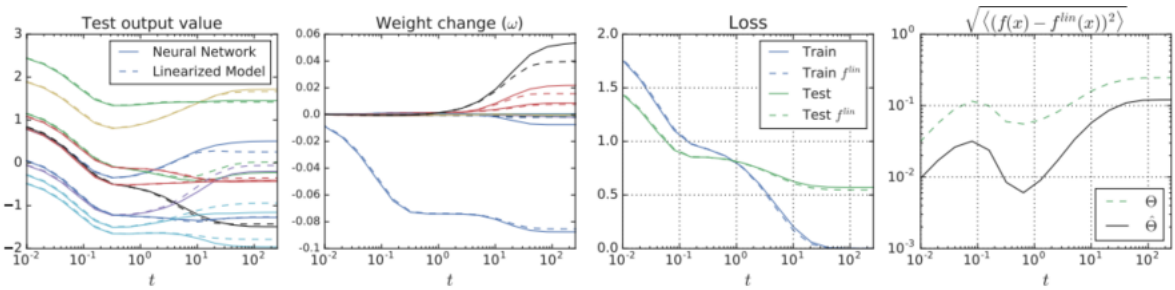


Figure 3: **Full batch gradient descent on a model behaves similarly to analytic dynamics on its linearization, both for network outputs, and also for individual weights.** A binary CIFAR classification task with MSE loss and a ReLU fully-connected network with 5 hidden layers of width $n=2048$, $\eta=0.01$, $|\mathcal{D}|=256$, $k=1$, $\sigma_w^2=2.0$, and $\sigma_b^2=0.1$. Left two panes show dynamics for a randomly selected subset of datapoints or parameters. Third pane shows that the dynamics of loss for training and test points agree well between the original and linearized model. The last pane shows the dynamics of RMSE between the two models on test points. We observe that the empirical kernel $\hat{\Theta}$ gives more accurate dynamics for finite width networks.

NTK的局限性

1. 局限于“无限宽”的神经网络，懒惰训练，网络提取的特征在初始化时就已固定；然而在实际应用中效果好的往往是那些比较“深”的模型（特征学习）——发展了一些其它理论用于特征学习分析（平均场理论，Mean Field Theory, MFT）

2. 架构拓展，从MLP拓展到CNN、RNN、ResNet、Transformer

NTK目前的主要作为一种理论分析工具研究神经网络的特征

NTK与任务算术（NIPS2023 oral）

Task Arithmetic in the Tangent Space: Improved Editing of Pre-Trained Models

Guillermo Ortiz-Jimenez*
EPFL, Lausanne, Switzerland
guillermo.ortizjimenez@epfl.ch

Alessandro Favero*
EPFL, Lausanne, Switzerland
alessandro.favero@epfl.ch

Pascal Frossard
EPFL, Lausanne, Switzerland
pascal.frossard@epfl.ch

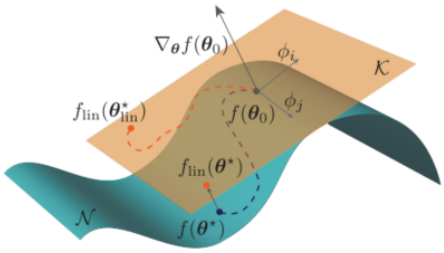
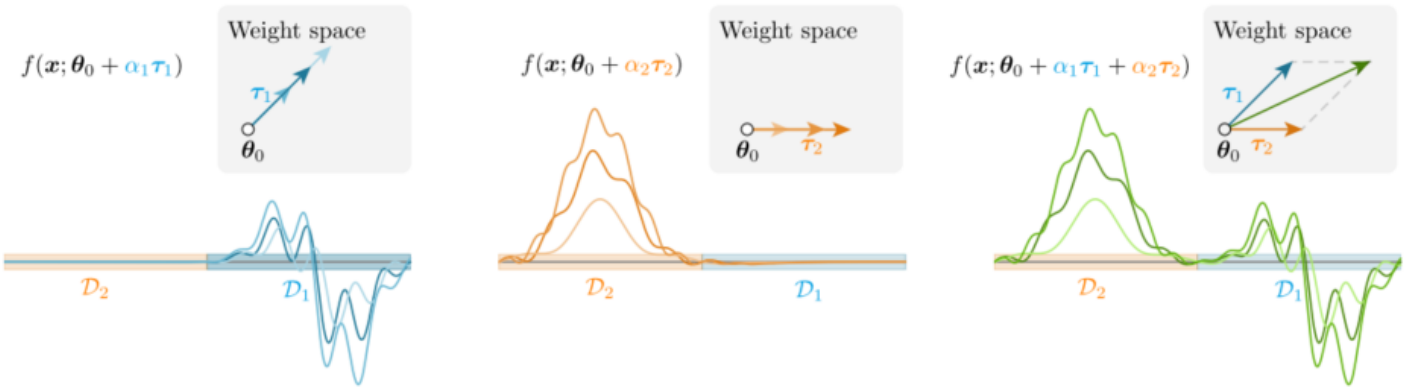


Figure 4: Conceptual illustration of the different approaches we use to edit a pretrained model $f(\cdot\,;\boldsymbol{\theta}_0)$. Here $\mathcal{N}$ represents the space of neural network functions f, non-linearly parameterized by $\boldsymbol{\theta} \in \boldsymbol{\Theta}$; and $\mathcal{K}$ its tangent space, given by the space of linearized functions f_{lin}.

研究思路：先前的研究认为任务算术之所以有效，是因为微调发生在模型的线性区域——通过实验验证CLIP模型微调时并不服从线性特征——将模型线性化后，进行”任务算术“时表现出更好的相对稳定性——想办法限制模型微调位于一个线性空间内——将模型近似为一个线性化模型进行训练



理想状态下，任务向量 τ_1 和 τ_2 互相解耦，叠加也不会影响对方的任务效果

先前的研究认为，任务算术之所以有效，是因为微调发生在预训练模型的一个局部线性区域

实验验证：首先构建一个由微调后模型衍生的post-hoc线性化模型 f_{lin}

Property 2 (Post-hoc linearization). *The change in the network output after training can be approximated by its first-order Taylor expansion, i.e.,* $f(\boldsymbol{x};\boldsymbol{\theta}^\star) - f(\boldsymbol{x};\boldsymbol{\theta}_0) \approx (\boldsymbol{\theta}^\star - \boldsymbol{\theta}_0)^\top \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x};\boldsymbol{\theta}_0)$.

In simple terms, the approximation of the network in the tangent space around initialization must hold after fine-tuning. To test this, we evaluate the performance of the *post-hoc* linearized version of f, f_{lin}. That is, we apply the fine-tuned task vectors $\boldsymbol{\tau} = \boldsymbol{\theta}^\star - \boldsymbol{\theta}_0$ to the linear approximation of f at $\boldsymbol{\theta}_0$, *i.e.*,

$$f_{\text{lin}}(\boldsymbol{x};\boldsymbol{\theta}_0 + \boldsymbol{\tau}) = f(\boldsymbol{x};\boldsymbol{\theta}_0) + \boldsymbol{\tau}^\top \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x};\boldsymbol{\theta}_0), \tag{3}$$

and we check whether $f_{\text{lin}}(\cdot\,;\boldsymbol{\theta}^\star)$ performs similarly to $f(\cdot\,;\boldsymbol{\theta}^\star)$[2].

Table 1: **Task addition.** Average absolute (%) and normalized accuracies (%) of different CLIP ViTs edited by adding the sum of the task vectors of 8 tasks. We report results for the non-linear and linearized models of Sections 3 and 5 normalizing performance by their single-task accuracies.

Method	ViT-B/32		ViT-B/16		ViT-L/14	
	Abs. (↑)	Norm. (↑)	Abs. (↑)	Norm. (↑)	Abs. (↑)	Norm. (↑)
Pre-trained $f(\cdot\,;\boldsymbol{\theta}_0)$	48.4	–	55.2	–	64.4	–
Non-lin. FT $f(\cdot\,;\boldsymbol{\theta}_0+\boldsymbol{\tau})$	71.4	76.5	75.5	80.0	85.1	88.8
Post-hoc lin. $f_{\text{lin}}(\cdot\,;\boldsymbol{\theta}_0+\boldsymbol{\tau})$	57.1	81.9	65.0	85.2	75.2	90.0
Linear. FT $f_{\text{lin}}(\cdot\,;\boldsymbol{\theta}_0+\boldsymbol{\tau}_{\text{lin}})$	**76.5**	**85.4**	**81.3**	**86.0**	**88.5**	**93.5**

实验发现post-hoc lin的准确率很差（57.1%），远低于常规操作的Non-lin FT（71.4%），说明微调不只发生在预训练模型的线性区域

但是，post-hoc lin的相对准确率（81.9%）大于Non-lin FT（76.5%），说明**线性化模型**的任务算术能够更好地保留微调模型的能力（也通过实验验证了）

$$\text{Normalized accuracy} = \frac{1}{T}\sum_{t=1}^{T} \frac{\underset{\boldsymbol{x}\sim\mu_t}{\text{acc}}\left[f(\boldsymbol{x};\boldsymbol{\theta}_0 + \sum_{t'}\boldsymbol{\tau}_{t'})\right]}{\underset{\boldsymbol{x}\sim\mu_t}{\text{acc}}\left[f(\boldsymbol{x};\boldsymbol{\theta}_0 + \boldsymbol{\tau}_t)\right]}.$$

因此如果能让模型一开始就在一个线性空间内微调，那么进行任务算术效果肯定很好——直接用线性化模型的输出计算loss作为微调目标进行优化

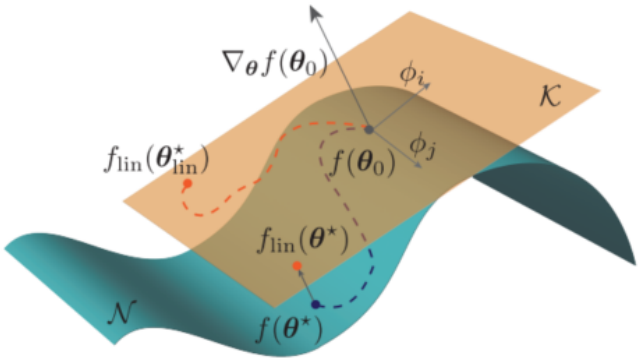


Figure 4: Conceptual illustration of the different approaches we use to edit a pretrained model $f(\cdot\,; \boldsymbol{\theta}_0)$. Here $\mathcal{N}$ represents the space of neural network functions f, non-linearly parameterized by $\boldsymbol{\theta} \in \boldsymbol{\Theta}$; and $\mathcal{K}$ its tangent space, given by the space of linearized functions f_{lin}.

使用NTK进行分析部分——证明任务算术有效的原因，是因为在预训练模型中任务向量互相解耦，这种能力是在预训练中涌现的：

To explore whether CLIP models use localized eigenfunctions for task arithmetic, we diagonalize the matrix $(K_{\text{NTK}})_{ij} = k_{\text{NTK}}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ with $\boldsymbol{x}_i \in \mathcal{D}_t$, *i.e.*, the task on which we trained, and $\boldsymbol{x}_j \in \mathcal{D}_t \cup \mathcal{D}_{t'}$, where $\mathcal{D}_{t'}$ is the support of a control task. If the eigenfunctions used to represent $f^\star(\boldsymbol{x})$ are localized, then the power of the eigenvectors of K_{NTK} must be concentrated in the points belonging to the dataset used for training. To measure this concentration, we introduce the local energy $\mathcal{E}_{\text{loc}}(\boldsymbol{x}) = \sum_\rho \phi_\rho^2(\boldsymbol{x})$, which sums the power of all the eigenfunctions ϕ_ρ at a given point $\boldsymbol{x}$.

构建NTK矩阵 $(K_{NTK})_{ij} = k_{NTK}(x_i, x_j)$，其中 x_i 来自训练集 D_t，其中 x_j 来自训练集和参照集的并集 $D_t \bigcup D_{t'}$，对 $(K_{NTK})_{ij}$ 进行svd分解得到若干特征函数 $\phi_\rho(x)$。期望在 D_t 上训练得到的特征函数 $\phi_\rho(x)$ 应只对训练任务响应高，对参照集响应低

核函数：给定两个样本输入x和y，返回它们在特征空间（高维）中对应向量的点积

$$K(x,y) = \langle \phi(x), \phi(y) \rangle$$

线性核：$K(x,y) = xy$

多项式核：

$$\begin{aligned} K(x,y) &= (xy+1)^2 \\ &= \begin{bmatrix} x^2 \\ \sqrt{2}x \\ 1 \end{bmatrix}^T \cdot \begin{bmatrix} y^2 \\ \sqrt{2}y \\ 1 \end{bmatrix} \\ &= \langle \phi(x), \phi(y) \rangle \end{aligned}$$

高斯核：

$$\begin{aligned} K(x,y) &= e^{-\frac{(x-y)^2}{2}} \\ &= \sum_{n=0}^{\infty} \left(e^{-\frac{x^2}{2}} \frac{x^n}{\sqrt{n!}} \right) \cdot \left(e^{-\frac{y^2}{2}} \frac{y^n}{\sqrt{n!}} \right) \\ &= \langle \phi(x), \phi(y) \rangle \end{aligned}$$

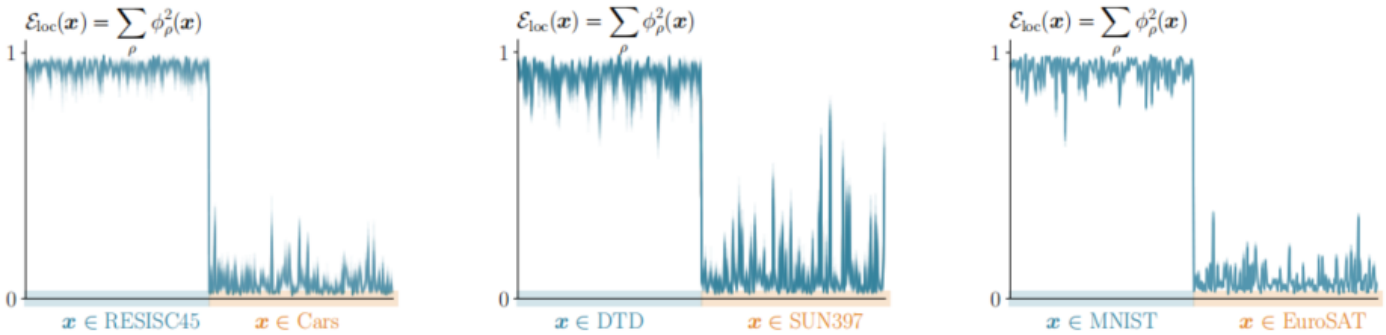


Figure 14: **Eigenfunction localization.** Estimated support of the eigenfunctions of the NTK of a ViT-B/32 CLIP model trained on different datasets. The plot shows the sum of the local energy of the eigenfunctions over a random subset of the training and control supports

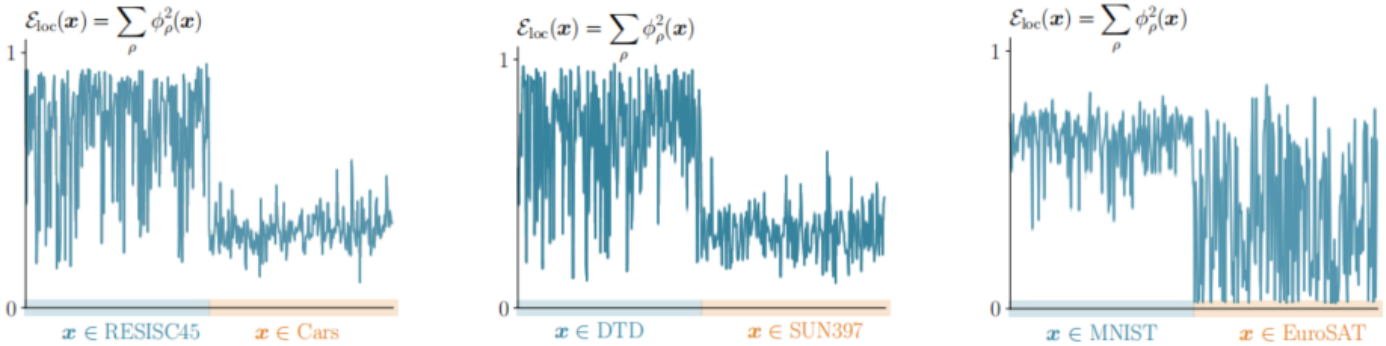


Figure 15: **Eigenfunction localization.** Estimated support of the eigenfunctions of the NTK of a randomly initialized ViT-B/32 model trained on different datasets. The plot shows the sum of the local energy of the eigenfunctions over a random subset of the training and control supports

NTK与微调（ICML2023）

A Kernel-Based View of Language Model Fine-Tuning

Sadhika Malladi [1] **Alexander Wettig** [1] **Dingli Yu** [1] **Danqi Chen** [1] **Sanjeev Arora** [1]

利用合适的prompt将下游任务转化为预训练任务的子任务->冻结预训练权重，计算 NTK 矩阵->使用NTK矩阵进行核回归，效果和微调相比差距在1~2点内->微调本质上非常接近于使用“预训练 NTK”进行线性分类(Lazy training)

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 2. 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.