Neat: Nonlinear Parameter-efficient Adaptation of Pre-trained Models

Anonymous ACL submission

Abstract

Fine-tuning pre-trained models often yields state-of-the-art performance but is computationally expensive when updating all parameters. Parameter-efficient fine-tuning (PEFT) methods, such as Low-Rank Adaptation (LoRA), address this by freezing pre-trained weights and introducing low-rank matrices. However, because LoRA relies on low-rank decomposition, it struggles to capture complex nonlinear dynamics and optimal optimization trajectories, resulting in a performance gap relative to full fine-tuning and inefficient parameter utilization. To overcome these issues, we propose NEAT, a nonlinear PEFT approach that employs a lightweight neural network to learn a nonlinear transformation of the pre-trained weights, thereby better approximating cumulative weight updates. Our theoretical analysis shows that NEAT achieves greater efficiency than LoRA while maintaining equivalent expressivity. Extensive experiments on four benchmarks and over twenty datasets demonstrate that NEAT significantly outperforms state-of-the-art baselines in both vision and text tasks.

1 Introduction

Pre-trained models, trained on extensive and diverse general-domain corpora, demonstrate exceptional generalization capabilities, benefiting a range of fundamental tasks, such as natural language understanding (Devlin, 2018; Liu, 2019), natural language generation (Touvron et al., 2023; AI@Meta, 2024), and image classification (Dosovitskiy et al., 2020). In order to adapt pre-trained models to specific downstream tasks, fine-tuning is typically employed. However, due to the large number of parameters in pre-trained models, full fine-tuning requires significant computational resources and incurs substantial memory overhead (Qin et al., 2024).

To address this challenge, various parameter-efficient fine-tuning (PEFT) techniques (Ding et al., 2023; Han et al., 2024) have been developed, enabling pre-trained models to be fine-tuned in resource-constrained environments (Lin et al., 2024). PEFT methods reduce the memory overhead of fine-tuning by introducing a small set of learnable parameters, updating only these lightweight components. These approaches allow pre-trained models to effectively adapt to new tasks while minimizing resource consumption. Among PEFT techniques, the Low-Rank Adaptation (LoRA) family (Hu et al., 2021a; Liu et al., 2024; Song et al., 2024; Büyükakyüz, 2024; Zhao et al., 2024) is widely regarded as one of the most effective and popular approaches due to its minimal architectural modifications, high efficiency, and strong performance. Instead of updating pre-trained model weight directly, LoRA instead introduces two learnable low-rank matrices for it, and approximate weight updates through their product. Since these low-rank matrices are much smaller than the original pre-trained weights, LoRA significantly reduces the memory overhead during fine-tuning.

Despite its widespread success, LoRA has limitations, particularly in capturing complex relationships in weight updates. By decomposing weight updates into low-rank matrices, LoRA effectively reduces the fine-tuning parameter space, but this comes at the cost of failing to capture complex patterns that are critical for many downstream tasks (Pan et al., 2024). Specifically, LoRA approximation often struggles to model the intricate optimization trajectories required for high performance, especially when the rank of the low-rank matrices is small. Consequently, LoRA-based methods often require higher ranks to close the gap between their performance and that of full fine-tuning, which increases the number of parameters and undermines the efficiency gains they were designed to achieve.

To overcome these limitations, we propose a novel nonlinear parameter-efficient adaptation method, NEAT, which incorporates a lightweight neural network into the adaptation process. Unlike LoRA, which approximates weight updates linearly through low-rank decomposition, NEAT models cumulative weight updates as explicit functions of the pre-trained model's original weights. This enables NEAT to capture complex, non-linear patterns in the weight space, improving adaptation performance without increasing the number of parameters. The key innovation in NEAT lies in introducing a neural network that transforms the pre-trained weights, approximating the updates with minimal additional computation. This nonlinear transformation enhances the expressiveness of the parameter updates while maintaining the efficiency. Importantly, this architecture facilitates a more efficient exploration of the optimization landscape, leading to better task adaptation, particularly in cases where linear methods like LoRA would require much larger ranks to achieve competitive results. We theoretically demonstrate that NEAT can achieve the same or greater expressivity than LoRA with fewer parameters.

The contributions are summarized as follows: 1) We propose NEAT, a novel parameter-efficient fine-tuning method that leverages nonlinear transformations, effectively capturing more complex weight updates. To the best of our knowledge, this is the first work to introduce nonlinear adaptation for LoRA-based PEFT methods; 2) The proposed NEAT enhances model performance while maintaining the efficiency. We theoretically show that NEAT can achieve a possibly improved parameter efficiency compared to LoRA; 3) We conduct extensive experiments on four benchmarks covering over twenty datasets. The experiments show that the proposed NEAT can outperform baselines on both vision and text tasks.

2 Related Works

In this section, we provide a concise overview of related work on Parameter-Efficient Fine-Tuning (PEFT) methods. PEFT methods aim to reduce the memory overhead of fine-tuning pre-trained models, enabling fine-tuning in resource-constrained environments. According to Han et al. (2024), PEFT methods can be categorized into: 1) **Additive PEFT methods** (Chronopoulou et al., 2023; Edalati et al., 2022; Lester et al., 2021; Wang et al., 2024c; Liu et al., 2022), 2) **Selective PEFT methods** (Guo et al., 2020; Das et al., 2023; Sung et al., 2021; Ansell et al., 2021; Zaken et al., 2021; Vucetic et al., 2022), 3) **Reparameterized PEFT methods** (Hu et al., 2021b; Valipour et al., 2022; Zhang et al., 2023; Karimi Mahabadi et al., 2021; Liu et al., 2024; Kopiczko et al., 2023), and 4) **Hybrid PEFT methods** (Mao et al., 2021; Chen et al., 2023; He et al., 2021; Zhang et al., 2022; Zhou et al., 2024). Among these, Low-Rank Adaptation (LoRA)-based methods, which are representative of reparameterized PEFT approaches, have gained significant attention due to their minimal architectural changes, no additional inference costs, and high efficiency. LoRA (Hu et al., 2021b) introduces two trainable low-rank matrices for each pre-trained model weight to approximate the desired updates of the original model. Extensions of LoRA include DyLoRA (Valipour et al., 2022), which dynamically adjusts the rank of the low-rank matrices during training to optimize for specific tasks; AdaLoRA (Zhang et al., 2023), which adaptively allocates the parameter budget among weight matrices based on their importance scores; and DoRA (Liu et al., 2024), which decomposes the pre-trained weight into magnitude and direction, applying LoRA only for direction updates. Other variants include VeRA (Kopiczko et al., 2023), which introduces shared frozen random matrices across layers to improve efficiency further, and RoseLoRA (Wang et al., 2024b), which employs a row- and column-wise sparse low-rank adaptation mechanism to selectively update the most significant parameters. FourierFT (Gao et al.) replaces the matrix multiplication in LoRA with a Fourier transform, while PiSSA (Meng et al., 2024) and MiLoRA (Wang et al., 2024a) update the principal and minor singular components of the weight matrix, respectively. However, existing PEFT methods rely on linear transformations to approximate pre-trained weight updates, which struggle to capture the complex relationships inherent in weight updates, leading to a significant performance gap compared to full fine-tuning. Meanwhile, existing research like (Teney et al., 2024) also demonstrates that nonlinear activation is an integral part of the neural network driving its success.

3 Methodology

In this section, we start with a brief introduction of LoRA. Motivated by a key limitation in LoRA parameter efficiency that roots from LoRA parameterization form, we propose NEAT, a novel PEFT method to solve the issue. Notably, NEAT is able to achieves better parameter efficiency provably.

3.1 Preliminary

LoRA (Hu et al., 2021b) assumes that the updates to model weights during the fine-tuning exhibit low-rank properties. Built upon this, LoRA models the *incremental update* of some weight matrix $\mathbf{W}^0 \in \mathbb{R}^{d_1 \times d_2}$ in a pre-trained model approximately by the product of two learnable low-rank matrices

$$\mathbf{W} = \mathbf{W}^0 + \Delta\mathbf{W} = \mathbf{W}^0 + \mathbf{A}\mathbf{B},$$

where $\mathbf{A} \in \mathbb{R}^{d_1 \times r}$ and $\mathbf{B} \in \mathbb{R}^{r \times d_2}$ with $r \ll \min(d_1, d_2)$. When conducting fine-tuning, only introduced two low-rank matrices $\mathbf{A}$ and $\mathbf{B}$ will be updated and the pre-trained weight $\mathbf{W}^0$ is frozen, as represented by the following optimization

$$\min_{\mathbf{A},\mathbf{B}} \mathcal{L}(\mathcal{D}_{\text{train}}; \mathbf{W}^0 + \mathbf{A}\mathbf{B}), \quad (1)$$

where $\mathcal{D}_{\text{train}}$ is the training set used for fine-tuning and $\mathcal{L}$ is the loss function. Since $\mathbf{A}$ and $\mathbf{B}$ are both low-rank matrices that contain significantly fewer parameters compared with the original $\mathbf{W}^0$, the LoRA costs much less memory space compared to the fully fine-tuning.

3.2 Inherent Limitation of LoRA Formulation

While LoRA family have demonstrated remarkable parameter efficiency in fine-tuning pre-trained models for diverse downstream tasks, we argue that their product-based formulation are suboptimal for capturing the full fine-tuning dynamics in an efficient way.

Specifically, when fully fine-tuning a pre-trained model, the update process of weight $\boldsymbol{W}$ is typically performed through an iterative gradient descent:

$$\mathbf{W}_t^0 = \mathbf{W}_{t-1}^0 - \eta \nabla_{\mathbf{W}_{t-1}^0} \mathcal{L},$$

where $\mathbf{W}_0^0 = \mathbf{W}^0$ is the initial state, η is the learning rate, and $\mathbf{W}_t^0$ represents the weights after t iterations. The cumulative change in the weights over time can be represented as:

$$\Delta\mathbf{W} = \mathbf{W}_t^0 - \mathbf{W}_0^0.$$

This weight change $\Delta\mathbf{W}$ can be interpreted as a function of the original pre-trained weights $\mathbf{W}^0$, capturing the model's adaptation to the specific task during fine-tuning.

Nonetheless, LoRA matrices $\mathbf{A}$ and $\mathbf{B}$ are parameterized in a free way without any dependency on $\mathbf{W}^0$. While gradient $\nabla_{\boldsymbol{A}}\mathcal{L}$ and $\nabla_{\boldsymbol{B}}\mathcal{L}$ are implicit functions of $\mathbf{W}^0$, making final learned $\mathbf{A}_t, \mathbf{B}_t$ indirectly depends on $\mathbf{W}^0$ as well, as will be proved shortly, the lack of explicit dependency still makes LoRA *inherently suboptimal* for fine-tuning pre-trained models.

3.3 Nonlinear Parameter-efficient Adaptation

Motivated by the above analysis on LoRA's limitation, we propose to approximate $\Delta\mathbf{W}$ using a lightweight neural network that *explicitly* takes pre-trained model weight $\mathbf{W}^0$ as input and outputs the weight update directly. By doing so, our approach captures more complex and richer transformation of the weights in a more efficient manner. We refer to our method as **n**onlinear parameter-**e**fficient **a**dap**t**ation method (NEAT).

Inspired by the effectiveness of LoRA's incremental updates paradigm, the proposed NEAT also provides incremental update of pre-trained models. However, NEAT modifies the forward pass of the model by introducing a dynamic *nonlinear* weight transformation. Specifically, the modified model's forward propagation is formulated as:

$$\boldsymbol{y} = (\mathbf{W}^0 + f(\mathbf{W}^0; \boldsymbol{\theta}))\boldsymbol{x}.$$

Here $\boldsymbol{x}$ and $\boldsymbol{y}$ are the input and output with respect to the current layer, respectively, and $f(\cdot; \boldsymbol{\theta}) : \mathbb{R}^{d_1 \times d_2} \to \mathbb{R}^{d_1 \times d_2}$ is a nonlinear neural network parameterized by learnable parameter $\boldsymbol{\theta}$. The neural network $f(\mathbf{W}^0; \boldsymbol{\theta})$ generates the weight update as a function of $\mathbf{W}^0$.

To ensure the parameter efficiency of our NEAT, the learnable neural network $f(\mathbf{W}^0; \boldsymbol{\theta})$ should be lightweight, i.e., the number of parameters in $f(\mathbf{W}^0; \boldsymbol{\theta})$ should be much fewer than that of the original pre-trained weight $\mathbf{W}^0$. Therefore, we parametrize $f(\mathbf{W}^0; \boldsymbol{\theta})$ as a neural network with *bottleneck* layers. For example, a simple case is $f(\mathbf{W}^0; \boldsymbol{\theta}) = \sigma(\mathbf{W}^0 \mathbf{\Theta}_1)\mathbf{\Theta}_2$, where $\boldsymbol{\theta} = (\mathbf{\Theta}_1, \mathbf{\Theta}_2) \in \mathbb{R}^{d_2 \times r} \times \mathbb{R}^{r \times d_2}$ with $r \ll \min(d_1, d_2)$, and $\sigma(\cdot)$ is some activation function such as ReLU (Glorot et al., 2011). We can also increase the layers or add activation function for the output of $f(\mathbf{W}^0; \boldsymbol{\theta})$ to enhance the model expressiveness.

During fine-tuning, the optimization objective is to minimize the task-specific loss function, which

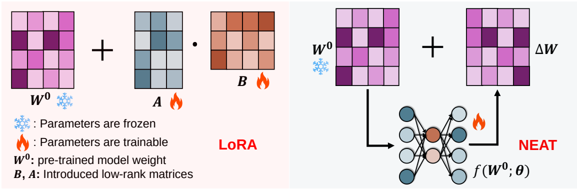
W^0
A
B
❄ : Parameters are frozen
🔥 : Parameters are trainable
W^0: pre-trained model weight
B, A: Introduced low-rank matrices
LoRA
W^0
ΔW
NEAT
$f(W^0;\theta)$

Figure 1: Framework of proposed NEAT.

can be represented as

$$\min_{\boldsymbol{\theta}} \ \mathcal{L}(\mathcal{D}_{\text{train}}; \mathbf{W}^0 + f(\mathbf{W}^0; \boldsymbol{\theta})),$$

where the original pre-trained weight $\mathbf{W}^0$ is frozen, and only neural network parameters θ are updated. The overview of NEAT is shown in Fig. 1.

Remark 3.1. The benefit of our new formulation lies in two folds. First, our incremental update $f(\mathbf{W}^0; \boldsymbol{\theta})$ is an explicit function of $\mathbf{W}^0$, allowing it to capture updates in a more effective way. Second, the neural network-based $f(\mathbf{W}^0; \boldsymbol{\theta})$ allows for dynamic, non-linear weight updates that can capture more complex interactions. These two advantages make NEAT a more effective and efficient PEFT method than existing LoRA-based approaches.

3.4 Theoretical Analysis

We end up this section with a theoretical analysis of the sub-optimality of LoRA parameter efficiency by proving that NEAT is capable of achieving equivalent or even higher efficiency certain conditions. Specifically, suppose NEAT adopts the following lightweight architecture as mentioned in Sec 3.3

$$f(\mathbf{W}^0; \boldsymbol{\theta}) = \sigma(\mathbf{W}^0 \boldsymbol{\Theta}_1) \boldsymbol{\Theta}_2.$$

Then below proposition shows that, NEAT can achieve the same expressivity offered by LoRA using fewer parameters under certain conditions. Here the expressivity is measured in terms of minimum attainable loss.

Proposition 3.2. *Given pre-trained weight matrix* $\mathbf{W}^0$*. Let* σ *denote ReLU activation function, and* $\boldsymbol{U}^0 \in \mathbb{R}^{d_1 \times \operatorname{rank}(\mathbf{W}^0)}$ *be the left singular vectors of* $\mathbf{W}^0$*. Suppose that the fine-tuning loss* $\mathcal{L}$ *is invariant under the the projection of the weight matrix to the left singular space of* $\mathbf{W}^0$*, i.e.,* $\mathcal{L}(\mathcal{D}_{train}; \mathbf{W}) = \mathcal{L}(\mathcal{D}_{train}; \boldsymbol{U}^0 \boldsymbol{U}^{0\top} \mathbf{W})$ *for any* $\mathbf{W} \in \mathbb{R}^{d_1 \times d_2}$*. Then, for any* $r \geq 1$*,*

$$\begin{aligned}
&\min_{\substack{\boldsymbol{\Theta}_1 \in \mathbb{R}^{d_2 \times 2r}, \\ \boldsymbol{\Theta}_2 \in \mathbb{R}^{2r \times d_2}}} \mathcal{L}(\mathcal{D}_{train}; \mathbf{W}^0 + f(\mathbf{W}^0; (\boldsymbol{\Theta}_1, \boldsymbol{\Theta}_2))) \\
&\leq \min_{\substack{\mathbf{A} \in \mathbb{R}^{d_1 \times r}, \\ \mathbf{B} \in \mathbb{R}^{r \times d_2}}} \mathcal{L}(\mathcal{D}_{train}; \mathbf{W}^0 + \mathbf{A}\mathbf{B}) \\
&\leq \min_{\substack{\boldsymbol{\Theta}_1 \in \mathbb{R}^{d_2 \times r}, \\ \boldsymbol{\Theta}_2 \in \mathbb{R}^{r \times d_2}}} \mathcal{L}(\mathcal{D}_{train}; \mathbf{W}^0 + f(\mathbf{W}^0; (\boldsymbol{\Theta}_1, \boldsymbol{\Theta}_2))).
\end{aligned}$$

In words, Prop 3.2 demonstrates the (approximate) equivalence of LoRA and NEAT in terms of their expressivity. Specifically, the minimum attainable loss using rank-r LoRA can be achieved by NEAT with $2r$ hidden units, and conversely, the minimum attainable loss using NEAT with r hidden units can be achieved rank-r LoRA, provided the invariance assumption holds. This equivalence further implies that the function classes realized by NEAT with $O(r)$ hidden dimensions and rank-r LoRA are equivalent in expressivity, as the result holds for any loss functions.

Importantly, *this highlights a potential improvement in parameter efficiency by NEAT*. Namely, NEAT with $O(rd_2)$ parameters maintains the expressivity of LoRA with $r(d_1 + d_2)$ parameters. That it to say, NEAT offers a significant improvement in parameter efficiency when $d_2 \ll d_1$ (a condition that widely holds for the down projection matrix of transformers fully-connected layers (Vaswani, 2017; Dosovitskiy et al., 2021)). In such cases, NEAT provably achieves better parameter efficiency than LoRA. The added parameter efficiency can also improve sample efficiency by allowing the model to learn representations with the same or fewer data points.

The invariance assumption in Proposition 3.2 pertains to the pre-trained model, and asserts that the later layers of the model depends solely on the task-relevant feature space. Given that we fine-tune a pre-trained model, the later layers are expected to capture this task-relevant feature space, which is described by the left singular space of $\mathbf{W}^0$. In practice, since the later layers primarily rely on this pre-trained feature space, the principal directions of the pre-trained weight matrix, represented by its singular vectors, encode most of the useful features for downstream tasks. This makes the loss largely invariant to changes outside this subspace. The proof is available in Section A.1.

If we consider a sinusoid activation function $\sigma_{\text{p}}(x) = \sin(2\pi x)$, then stronger result that NEAT has expressivity (almost) greater than or equal to a LoRA with possibly more parameters can be established without the invariance assumption. We defer the result to the Appendix A.2.

4 Experiment

In the experiments, we evaluate the proposed NEAT and answer the following questions: **RQ1**) How does NEAT compare to state-of-the-art PEFT methods on NLP tasks? **RQ2**) How does NEAT compare to state-of-the-art PEFT methods on vision tasks? **RQ3**) How does the performance of NEAT vary with different fine-tuned modules, depths of the lightweight neural network, or non-linear activation functions?

4.1 Benchmarks and Experiment Setups

We experiment NEAT on datasets from four representative benchmarks:

Commonsense Reasoning covers diverse multi-choice problems from BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2019), ARC-e and ARC-c (Clark et al., 2018), and OpenBookQA (Mihaylov et al., 2018) datasets. Following Wang et al. (2024a), we finetune LLaMA2-7B (Touvron et al., 2023) and LLaMA3-8B (AI@Meta, 2024) on Commonsense170K (Hu et al., 2023) benchmark which combines all previous training sets, and evaluate the accuracy on their testing sets separately.

Arithmetic Understanding consists of two math reasoning datasets: GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021). We finetune LLaMA2-7B (Touvron et al., 2023) and LLaMA3-8B (AI@Meta, 2024) on MetaMath (Yu et al., 2023) dataset following Wang et al. (2024a). Models need to generate correct answers, and accuracy is used as the evaluation metric.

Natural Language Understanding consists of eight datasets from the GLUE benchmark (Wang et al., 2018). We follow the evaluation metrics and setups from Gao et al. (2024); Wu et al. (2024b).

Image Classification consists of Oxford-Pets (Parkhi et al., 2012), CIFAR10 (Krizhevsky, 2009), DTD (Cimpoi et al., 2014), EuroSAT (Helber et al., 2019), RESISC45 (Cheng et al., 2017), StanfordCars (Krause et al., 2013), FGVC (Maji et al., 2013), and CIFAR100 (Krizhevsky, 2009) following Gao et al. (2024). The first five datasets have small label spaces, while the last three have large label spaces.

Baselines methods are chosen on a task basis. On each task, NEAT is compared with representative baselines from the corresponding domain.

See our appendix for details about the datasets (App E), baseline methods (App C), and hyperparameters (App D).

4.2 Performance Comparison

We showcase NEAT performance on different tasks.

4.2.1 Commonsense Reasoning

We experiment NEAT with eight commonsense reasoning datasets to address RQ1, results are shown in Tab 1. We compare the performance of three state-of-the-art baselines with the proposed NEAT, and NEAT consistently outperforms all of them, achieving the highest accuracy on all tasks. Specifically, NEAT surpasses LoRA, PiSSA, and MiLoRA in terms of average accuracy by 4.6%, 10%, and 2.5%, respectively, when using LLaMA2-7B as the backbone. On LLaMA3-8B as the backbone, NEAT demonstrates average improvements of 4.9%, 11.8%, and 2.9% over LoRA, PiSSA, and MiLoRA, respectively. These results highlight the effectiveness and superiority of NEAT as a PEFT method.

4.2.2 Arithmetic Reasoning

In this section, we present results on two arithmetic reasoning tasks in Tab 2 to help address RQ1. From the table, while full fine-tuning (FFT) achieves highest accuracy across the two datasets, the performance gap between the proposed NEAT and FFT is very small, despite that NEAT relies on significantly fewer trainable parameters. Moreover, compared to state-of-the-art PEFT baselines, NEAT achieves remarkable performance improvements. In terms of average accuracy, NEAT demonstrates improvements of 7.5%, 12.4%, and 2.4% over LoRA, PiSSA, and MiLoRA, respectively. These results on clearly confirm that NEAT is highly effective and efficient for complex reasoning tasks.

4.2.3 Natural Language Understanding

We further conduct experiments on the GLUE to answer RQ1, results are shown in Tab 3. From the table, NEAT significantly outperforms state-of-the-art PEFT methods. Specifically, NEAT-S, which uses a similar number of trainable parameters as FourierFT (Gao et al., 2024), DiReFT (Wu et al., 2024b), and LoReFT (Wu et al., 2024b), surpasses

Table 1: Common Reasoning performance of NEAT and PEFT baselines on LLaMA 2-7B and LLaMA 3-8B. Results marked with "+" are taken from Liu et al. (2024), and those marked with "∗" are taken from Wang et al. (2024a). Best results are in **bold**. "AVG" means the average accuracy of all datasets.

Model	PEFT	Accuracy (↑)								
		BoolQ	**PIQA**	**SIQA**	**HellaSwag**	**WinoGrande**	**ARC-e**	**ARC-c**	**OBQA**	**AVG**
LLaMA2-7B	LoRA$^+$	69.8	79.9	79.5	83.6	82.6	79.8	64.7	81.0	77.6
	PiSSA*	67.6	78.1	78.4	76.6	78.0	75.8	60.2	75.6	73.8
	MiLoRA*	67.6	83.8	80.1	88.2	82.0	82.8	68.8	80.6	79.2
	NEAT	**71.9**	**84.0**	**80.4**	**88.9**	**84.6**	**86.5**	**71.6**	**83.0**	**81.4**
LLaMA3-8B	LoRA$^+$	70.8	85.2	79.9	91.7	84.3	84.2	71.2	79.0	80.8
	PiSSA*	67.1	81.1	77.2	83.6	78.9	77.7	63.2	74.6	75.4
	MiLoRA*	68.8	86.7	77.2	92.9	85.6	86.8	75.5	81.8	81.9
	NEAT	**72.1**	**87.0**	**80.9**	**94.3**	**86.7**	**91.4**	**78.9**	**84.8**	**84.5**

Table 2: Arithmetic Reasoning performance on LLaMA 2-7B. Results marked with "+" are taken from Yu et al. (2023), and those marked with "∗" are taken from Wang et al. (2024a). Best results are in **bold**. "AVG" means the average accuracy of all datasets.

Method	**GSM8K**	**MATH**	**AVG**
FFT $^+$	66.50	19.80	43.20
LoRA*	60.58	16.88	38.73
PiSSA*	58.23	15.84	37.04
MiLoRA*	63.53	17.76	40.65
NEAT	**65.05**	**18.30**	**41.68**

all PEFT baselines and experiences only a small performance drop (0.2%) compared to FFT. Additionally, NEAT-L exceeds the performance of all baselines, including FFT, with roughly the same number of trainable parameters as in LoRA. These results demonstrate that NEAT exhibits excellent generalization ability while maintaining great parameter efficiency.

4.2.4 Image Classification

In this section, we conduct experiments on image classification tasks to address RQ2, NEAT uses depth of 6, and results are shown in Tab 4. From the table, NEAT significantly outperforms LoRA and FourierFT using the same number of trainable parameters. Specifically, NEAT achieves performance improvements of 11.05%, 7.30%, and 26.02% compared to LoRA, FourierFT, and LP, respectively. Furthermore, compared to FFT, the proposed NEAT shows negligible performance drop (86.49% v.s. 86.34%), while using only 0.3% of the trainable parameters required by FFT. This demonstrates that NEAT exhibits exceptional adaptation capability not only on NLP tasks, but also on vision tasks as well. Additionally, it verifies the effectiveness of the nonlinear adaptation used in NEAT.

4.3 Ablation Study

In this section, we present an ablation study with two variants of LoRA to validate the effectiveness of our proposed framework: 1) nonlinear LoRA $\boldsymbol{y} = (\boldsymbol{W_0} + \boldsymbol{\sigma}(\mathbf{A})\mathbf{B})\boldsymbol{x}$, and 2) multiplicative LoRA $\boldsymbol{y} = (\mathbf{W}_0 + \mathbf{W}_0\mathbf{A}\mathbf{B})\boldsymbol{x}$. Experiments are conducted on image classification benchmarks, and results are reported in Tab 5. According to the table, both nonlinear LoRA and multiplicative LoRA perform worse than NEAT. This highlights the effectiveness of incorporating nonlinear approximations and explicitly using model weights as input to the nonlinear function in NEAT.

4.4 Sensitivity w.r.t. Depth

The expressiveness of neural networks grow with its depth (Raghu et al., 2017). This implies that deeper NEAT may be able to capturing more complex patterns required by ideal weight updates. In this section, we experiments NEAT with varying depth on vision, NLU, commonsense reasoning, and arithmetic reasoning benchmarks to address RQ3. Results are shown in Fig 2. See NEAT layer architecture in Fig 5 and more details and results in App F To ensure fair comparison, we fix all other hyper-parameters when varying depth.

From Fig 2, increasing network depth leads to better performance. Specifically, when using depth of 6, the classification accuracy achieves 81.04%, marking a 1.7% improvement compared with using depth of 2. On RTE, using 6 layers also boosts

Table 3: GLUE benchmark performance on RoBERTa-base. Results marked with "$*$" are taken from Wu et al. (2024a). Best results are in **bold**. "AVG" means the average accuracy of all datasets. NEAT-S applies trainable modules to layers starting from the 4th layer, with hidden dimensions set to 1. This matches the parameter numbers of FourierFT. NEAT-L applies NEAT to all layers with hidden dimension 8, aligning the parameter budget of LoRA.

PEFT	Params (%)	Accuracy (↑)								
		MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	AVG
FFT	100%	87.3	94.4	87.9	62.4	92.5	91.7	78.3	90.6	85.6
Adapter*	0.318%	87.0	93.3	88.4	60.9	92.5	**90.5**	76.5	90.5	85.0
LoRA*	0.239%	86.6	93.9	88.7	59.7	**92.6**	90.4	75.3	90.3	84.7
Adapter$^{\text{FNN}*}$	0.239%	**87.1**	93.0	88.8	58.5	92.0	90.2	77.7	90.4	84.7
BitFit*	0.080%	84.7	94.0	88.0	54.0	91.0	87.3	69.8	89.5	82.3
RED*	0.016%	83.9	93.9	89.2	61.0	90.7	87.2	78.0	90.4	84.3
FourierFT	0.019%	84.7	94.2	90.0	63.8	92.2	88.0	79.1	**90.8**	85.3
DiReFT*	0.015%	82.5	92.6	88.3	58.6	91.3	86.4	76.4	89.3	83.2
LoReFT*	0.015%	83.1	93.4	89.2	60.4	91.2	87.4	79.0	90.0	84.2
NEAT-S	0.019%	84.9	94.3	90.2	64.6	92.0	88.3	78.3	90.5	85.4
NEAT-L	0.241%	86.9	**95.2**	**90.0**	**64.8**	92.3	90.3	**82.7**	90.7	**86.6**

Table 4: Image Classification performance on ViT-base. Best results are in **bold**. "AVG" means the average accuracy of all datasets. Results marked with "$*$" are taken from Gao et al. (2024).

Method	Params (M)	OxfordPets	StanfordCars	CIFAR10	DTD	EuroSAT	FGVC	RESISC45	CIFAR100	AVG
FFT*	85.8M	93.14	79.78	**98.92**	77.68	**99.05**	**54.84**	**96.13**	**92.38**	**86.49**
LP*	-	90.28	25.76	96.41	69.77	88.72	17.44	74.22	84.28	68.36
LoRA*	581K	93.19	45.38	98.78	74.95	98.44	25.16	92.70	92.02	77.58
FourierFT*	239K	93.05	56.36	98.69	77.30	98.78	32.44	94.26	91.45	80.29
NEAT	263K	**93.62**	**80.21**	98.78	**79.61**	98.85	52.93	94.71	**92.02**	86.34

Table 5: Ablation Study on image classification task. The parameters count is the same and "AVG" means the average accuracy of all datasets. For simple and fair comparison, NEAT uses depth of 2.

Method	OxfordPets	StanfordCars	CIFAR10	DTD	EuroSAT	FGVC	RESISC45	CIFAR100	AVG
Nonlinear LoRA	94.11	72.84	98.68	79.16	98.61	39.33	93.79	92.38	83.31
Multiplicative LoRA	93.57	77.32	98.68	77.57	98.81	46.79	94.34	91.86	84.81
NEAT	93.77	80.03	98.70	77.57	98.79	53.60	94.27	92.47	86.15

the performance by a remarkable 4% compared to using 2. In addition, deeper networks tend to bring more improvement on complex benchmarks, such as MATH and PIQA. Finally, we would like to highlight that NEAT intermediate layers consist much fewer parameters ($\mathbb{R}^{r\times r}$, where r is the hidden layer dimension) compared to the pre-trained model's weight. Therefore, the additional parameter overhead of stacking more hidden layers is negligible and does not affect the parameter efficiency of NEAT. These results further validate the effectiveness of introducing non-linear adaptation.

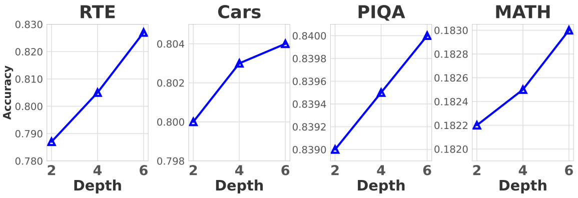

Figure 2: Accuracy on the RTE, StanfordCars, PIQA and MATH dataset with varying depths of the neural network used in NEAT. The depth here represents the total number of layers in the neural network. We choose depth equals to 2, 4 and 6 layers in the figure.

4.5 Sensitivity w.r.t. Activations

One key innovation of NEAT compared to LoRA and other PEFT methods, which rely solely on linear transformations for modeling weight updates, is the introduction of non-linear activations within the adaptation neural network. Since the choice of non-linear activations directly affects the learning pro-

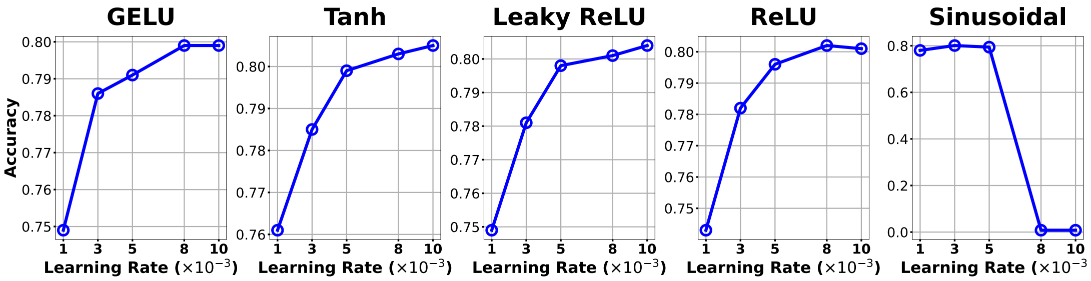

Figure 3: Influence of different nonlinear activations choices for NEAT. Experiments are conducted on StanfordCars, NEAT depth is fixed to 2. Different activations share a similar pattern of dependency on learning rate.

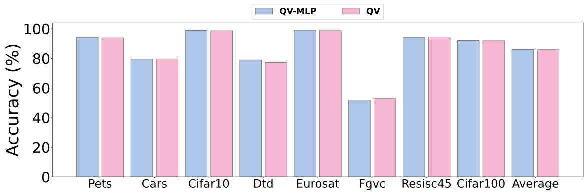

Figure 4: Accuracy of NEAT with different targeted fine-tuning modules, including just QV layers and a combination of QV and MLP layers, on image classification datasets.

cess and the dynamics of weight updates, we investigate how different non-linear activations affects the adaptation performance to address RQ3. To this end, we perform experiments on the StanfordCars benchmark using various non-linear activations, including ReLU, Leaky ReLU, GELU, Tanh, and sinusoidal activation ($\sigma_{\mathrm{p}}(x) = \sin(2\pi x)$). Corresponding results are presented in Fig 3. To ensure a fair comparison, the number of trainable parameters is fixed. We optimize other hyperparameters such as learning rate for better performance.

From the figure, the best performance achieved by different activation functions is similar, indicating that the adaptation potential of various activations is comparable. This implies that NEAT can benefit from various type of nonlinearity induced by different activations. However, it is also worth noting that sinusoidal activations encounters a performance drop at large learning rates. Consequently, tuning basic hyperparameters such as learning rate can still be beneficial. In conclusion, we suggest ReLU as a default choice in execution, given its practical simplicity (Teney et al., 2024).

5 Sensitivity w.r.t. Fine-tuned Module

We end up this section with a study on applying NEAT to different modules in a ViT, to help better understand RQ3. Specifically, given the importance of MLP in Transformer architecture, we compare two settings: 1) Following Hu et al. (2021a), we apply NEAT to the query and value layers (QV layers) in the multi-head self-attention module (MHSA) in ViT. 2) Besides QV layers, we also apply NEAT to MLP layers. We tune the hidden dimension r to ensure the same parameter scale for fair comparison, and tune the hyperparameters to maximize performance. Corresponding results are shown in Fig. 4.

From the figure, applying NEAT to the QV layers yields results comparable to applying NEAT to both the QV and MLP layers. This indicates that NEAT is robust to the selections of fine-tuning different modules. This finding confirms another key advantage of NEAT: it does not require extensive manual tuning on which parts (modules, layers) of the foundation model NEAT should be applied. Consequently, NEAT can be easily incorporated to a wide range of scenarios.

6 Conclusion

In this work, we propose NEAT, a novel parameter-efficient fine-tuning (PEFT) method that introduces nonlinear transformations to enhance model adaptation while maintaining efficiency. By incorporating a lightweight neural network that models cumulative weight updates as functions of the pre-trained weights, NEAT effectively captures complex, nonlinear structures in the weight space, allowing for more expressive and accurate adaptation to downstream tasks. Our theoretical analysis supports the efficacy of NEAT, demonstrating that it can achieve greater or equivalent expressiveness compared to existing LoRA, a popular and state-of-the-art PEFT method, with fewer number of parameters. Through extensive experiments on four benchmarks encompassing over twenty datasets with various pre-trained backbones, NEAT demonstrated superior performance on both NLP and vision tasks compared to existing state-of-the-art methods.

Limitations

The proposed NEAT framework requires to choose the activation function for the introduced lightweight neural network, which might require additional effort to tune. Fortunately, we observe that the model performance is not sensitive to the activation function and we set it to the widely used ReLU to achieve good performance in this paper.

References

AI@Meta. 2024. Llama 3 model card.

Alan Ansell, Edoardo Maria Ponti, Anna Korhonen, and Ivan Vulić. 2021. Composable sparse fine-tuning for cross-lingual transfer. *arXiv preprint arXiv:2110.07560*.

Tom M Apostol. 1990. Modular functions and dirichlet series in number theory.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. *arXiv preprint arXiv:1911.11641*.

Kerim Büyükakyüz. 2024. Olora: Orthonormal low-rank adaptation of large language models. *arXiv preprint arXiv:2406.01775*.

Jiaao Chen, Aston Zhang, Xingjian Shi, Mu Li, Alex Smola, and Diyi Yang. 2023. Parameter-efficient fine-tuning design spaces. *arXiv preprint arXiv:2301.01821*.

Gong Cheng, Junwei Han, and Xiaoqiang Lu. 2017. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883.

Alexandra Chronopoulou, Matthew E Peters, Alexander Fraser, and Jesse Dodge. 2023. Adaptersoup: Weight averaging to improve generalization of pretrained language models. *arXiv preprint arXiv:2302.07027*.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. 2014. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3606–3613.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Sarkar Snigdha Sarathi Das, Ranran Haoran Zhang, Peng Shi, Wenpeng Yin, and Rui Zhang. 2023. Unified low-resource sequence labeling by sample-aware dynamic sparse finetuning. *arXiv preprint arXiv:2311.03748*.

Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. 2023. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

Ali Edalati, Marzieh Tahaei, Ivan Kobyzev, Vahid Partovi Nia, James J Clark, and Mehdi Rezagholizadeh. 2022. Krona: Parameter efficient tuning with kronecker adapter. *arXiv preprint arXiv:2212.10650*.

Ziqi Gao, Qichao Wang, Aochuan Chen, Zijing Liu, Bingzhe Wu, Liang Chen, and Jia Li. Parameter-efficient fine-tuning with discrete fourier transform. In *Forty-first International Conference on Machine Learning*.

Ziqi Gao, Qichao Wang, Aochuan Chen, Zijing Liu, Bingzhe Wu, Liang Chen, and Jia Li. 2024. Parameter-efficient fine-tuning with discrete fourier transform. *arXiv preprint arXiv:2405.03003*.

Michael S Gashler and Stephen C Ashmore. 2014. Training deep fourier neural networks to fit time-series data. In *Intelligent Computing in Bioinformatics: 10th International Conference, ICIC 2014, Taiyuan, China, August 3-6, 2014. Proceedings 10*, pages 48–55. Springer.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings.

Demi Guo, Alexander M Rush, and Yoon Kim. 2020. Parameter-efficient transfer learning with diff pruning. *arXiv preprint arXiv:2012.07463*.

Zeyu Han, Chao Gao, Jinyang Liu, Sai Qian Zhang, et al. 2024. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*.

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. 2019. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021a. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021b. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. 2023. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. *arXiv preprint arXiv:2304.01933*.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. *Advances in Neural Information Processing Systems*, 34:1022–1035.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki Markus Asano. 2023. Vera: Vector-based random matrix adaptation. *arXiv preprint arXiv:2310.11454*.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 2013. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561.

A Krizhevsky. 2009. Learning multiple layers of features from tiny images. *Master's thesis, University of Tront*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of Machine Learning and Systems*, 6:87–100.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024. DoRA: Weight-decomposed low-rank adaptation. *arXiv:2402.09353*.

Yinhan Liu. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. 2013. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*.

Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Wen-tau Yih, and Madian Khabsa. 2021. Unipelt: A unified framework for parameter-efficient language model tuning. *arXiv preprint arXiv:2110.07577*.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. 2024. Pissa: Principal singular values and singular vectors adaptation of large language models. *arXiv preprint arXiv:2404.02948*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*.

Rui Pan, Xiang Liu, Shizhe Diao, Renjie Pi, Jipeng Zhang, Chi Han, and Tong Zhang. 2024. Lisa: Layerwise importance sampling for memory-efficient large language model fine-tuning. *arXiv preprint arXiv:2403.17919*.

Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. 2012. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3498–3505. IEEE.

Ruiyang Qin, Dancheng Liu, Zheyu Yan, Zhaoxuan Tan, Zixuan Pan, Zhenge Jia, Meng Jiang, Ahmed Abbasi, Jinjun Xiong, and Yiyu Shi. 2024. Empirical guidelines for deploying llms onto resource-constrained edge devices. *arXiv preprint arXiv:2406.03777*.

Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. 2017. On the expressive power of deep neural networks. In *international conference on machine learning*, pages 2847–2854. PMLR.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*.

Lin Song, Yukang Chen, Shuai Yang, Xiaohan Ding, Yixiao Ge, Ying-Cong Chen, and Ying Shan. 2024. Low-rank approximation for sparse attention in multimodal llms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13763–13773.

Yi-Lin Sung, Varun Nair, and Colin A Raffel. 2021. Training neural networks with fixed sparse masks. *Advances in Neural Information Processing Systems*, 34:24193–24205.

Damien Teney, Armand Mihai Nicolicioiu, Valentin Hartmann, and Ehsan Abbasnejad. 2024. Neural redshift: Random networks are not random functions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4786–4796.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, and et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. 2022. Dylora: Parameter efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. *arXiv preprint arXiv:2210.07558*.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. 2021. Spot: Better frozen model adaptation through soft prompt transfer. *arXiv preprint arXiv:2110.07904*.

Danilo Vucetic, Mohammadreza Tayaranian, Maryam Ziaeefard, James J Clark, Brett H Meyer, and Warren J Gross. 2022. Efficient fine-tuning of bert models on the edge. In *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1838–1842.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Hanqing Wang, Zeguan Xiao, Yixia Li, Shuo Wang, Guanhua Chen, and Yun Chen. 2024a. Milora: Harnessing minor singular components for parameter-efficient llm finetuning. *arXiv preprint arXiv:2406.09044*.

Haoyu Wang, Tianci Liu, Tuo Zhao, and Jing Gao. 2024b. Roselora: Row and column-wise sparse low-rank adaptation of pre-trained language model for knowledge editing and fine-tuning. *arXiv preprint arXiv:2406.10777*.

Yihan Wang, Jatin Chauhan, Wei Wang, and Cho-Jui Hsieh. 2024c. Universality and limitations of prompt tuning. *Advances in Neural Information Processing Systems*, 36.

Muling Wu, Wenhao Liu, Xiaohua Wang, Tianlong Li, Changze Lv, Zixuan Ling, Jianhao Zhu, Cenyuan Zhang, Xiaoqing Zheng, and Xuanjing Huang. 2024a. Advancing parameter efficiency in fine-tuning via representation editing. *arXiv:2402.15179*.

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D. Manning, and Christopher Potts. 2024b. ReFT: Representation finetuning for language models.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*.

Yuanhan Zhang, Kaiyang Zhou, and Ziwei Liu. 2022. Neural prompt search. *arXiv preprint arXiv:2206.04673*.

Hongyu Zhao, Hao Tan, and Hongyuan Mei. 2022. Tiny-attention adapter: Contexts are more important than the number of parameters. *arXiv preprint arXiv:2211.01979*.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. 2024. Galore: Memory-efficient llm training by gradient low-rank projection. *arXiv preprint arXiv:2403.03507*.

Han Zhou, Xingchen Wan, Ivan Vulić, and Anna Korhonen. 2024. Autopeft: Automatic configuration search for parameter-efficient fine-tuning. *Transactions of the Association for Computational Linguistics*, 12:525–542.

Appendix

A Details of Theoretical Results

In this section, we provide the proof of Proposition 3.2 and introduce additional theoretical results when we assume sinusoid activation.

A.1 Proof of Proposition 3.2

The intuition behind the proof is that we can always restore an identity function using two ReLU activation functions, i.e., $x = \sigma(x) - \sigma(-x)$ for any $x \in \mathbb{R}$

Proof of Proposition 3.2. We first show that

$$\begin{aligned}&\min_{\mathbf{\Theta}_1\in\mathbb{R}^{d_2\times 2r},\mathbf{\Theta}_2\in\mathbb{R}^{2r\times d_2}} \mathcal{L}(\mathcal{D}_{\text{train}}; \mathbf{W}^0 + f(\mathbf{W}^0; (\mathbf{\Theta}_1, \mathbf{\Theta}_2)))\\ \le &\min_{\mathbf{A}\in\mathbb{R}^{d_1\times r},\mathbf{B}\in\mathbb{R}^{r\times d_2}} \mathcal{L}(\mathcal{D}_{\text{train}}; \mathbf{W}^0 + \mathbf{AB}).\end{aligned}$$

Let $(\mathbf{A}^*, \mathbf{B}^*) = \arg\min_{\mathbf{A}\in\mathbb{R}^{d_1\times r},\mathbf{B}\in\mathbb{R}^{r\times d_2}} \mathcal{L}(\mathcal{D}_{\text{train}}; \mathbf{W}^0 + \mathbf{AB})$. Take $\mathbf{\Theta}_1^\# := [(\mathbf{W}^0)^\dagger\mathbf{A}^*; -(\mathbf{W}^0)^\dagger\mathbf{A}^*] \in \mathbb{R}^{d_2\times 2r}$ and $\mathbf{\Theta}_2^\# := [\mathbf{B}^{*\top}; -\mathbf{B}^{*\top}]^\top \in \mathbb{R}^{2r\times d_2}$, where $(\mathbf{W}^0)^\dagger \in \mathbb{R}^{d_2\times d_1}$ is the Moore-Penrose inverse of $\mathbf{W}^0$. Then, since σ is a ReLU activation function,

$$\begin{aligned}&f(\mathbf{W}^0; (\mathbf{\Theta}_1^\#, \mathbf{\Theta}_2^\#))\\ =&\sigma(\mathbf{W}^0\mathbf{\Theta}_1^\#)\mathbf{\Theta}_2^\#\\ =&\sigma(\mathbf{W}^0(\mathbf{W}^0)^\dagger\mathbf{A}^*)\mathbf{B}^* - \sigma(-\mathbf{W}^0(\mathbf{W}^0)^\dagger\mathbf{A}^*)\mathbf{B}^*\\ =&\mathbf{W}^0(\mathbf{W}^0)^\dagger\mathbf{A}^*\mathbf{B}^*.\end{aligned}$$

Note that $\mathbf{W}^0(\mathbf{W}^0)^\dagger = \boldsymbol{U}^0\boldsymbol{U}^{0\top}$ is the projection to the left singular space of $\mathbf{W}^0$. Hence

$$\begin{aligned}&\mathcal{L}(\mathcal{D}_{\text{train}}; \mathbf{W}^0 + f(\mathbf{W}^0; (\mathbf{\Theta}_1^\#, \mathbf{\Theta}_2^\#)))\\ =&\mathcal{L}(\mathcal{D}_{\text{train}}; \boldsymbol{U}^0\boldsymbol{U}^{0\top}\mathbf{W}^0 + \boldsymbol{U}^0\boldsymbol{U}^{0\top}\mathbf{A}^*\mathbf{B}^*)\\ =&\mathcal{L}(\mathcal{D}_{\text{train}}; \mathbf{W}^0 + \mathbf{A}^*\mathbf{B}^*),\end{aligned}$$

where the last equality follows from the invariance assumption. This gives the first inequality:

$$\begin{aligned}&\min_{\mathbf{\Theta}_1\in\mathbb{R}^{d_2\times 2r},\mathbf{\Theta}_2\in\mathbb{R}^{2r\times d_2}} \mathcal{L}(\mathcal{D}_{\text{train}}; \mathbf{W}^0 + f(\mathbf{W}^0; (\mathbf{\Theta}_1, \mathbf{\Theta}_2)))\\ \le &\mathcal{L}(\mathcal{D}_{\text{train}}; \mathbf{W}^0 + f(\mathbf{W}^0; (\mathbf{\Theta}_1^\#, \mathbf{\Theta}_2^\#)))\\ = &\mathcal{L}(\mathcal{D}_{\text{train}}; \mathbf{W}^0 + \mathbf{A}^*\mathbf{B}^*)\\ = &\min_{\mathbf{A}\in\mathbb{R}^{d_1\times r},\mathbf{B}\in\mathbb{R}^{r\times d_2}} \mathcal{L}(\mathcal{D}_{\text{train}}; \mathbf{W}^0 + \mathbf{AB}).\end{aligned}$$

We next show the following inequality:

$$\begin{aligned}&\min_{\mathbf{A}\in\mathbb{R}^{d_1\times r},\mathbf{B}\in\mathbb{R}^{r\times d_2}} \mathcal{L}(\mathcal{D}_{\text{train}}; \mathbf{W}^0 + \mathbf{AB})\\ \le &\min_{\mathbf{\Theta}_1\in\mathbb{R}^{d_2\times r},\mathbf{\Theta}_2\in\mathbb{R}^{r\times d_2}} \mathcal{L}(\mathcal{D}_{\text{train}}; \mathbf{W}^0 + f(\mathbf{W}^0; (\mathbf{\Theta}_1, \mathbf{\Theta}_2))).\end{aligned}$$

Take $\mathbf{A}^\# = \sigma(\mathbf{W}^0\mathbf{\Theta}_1^*) \in \mathbb{R}^{d_1\times r}$ and $\mathbf{B}^\# = \mathbf{\Theta}_2^* \in \mathbb{R}^{r\times d_2}$, where $(\mathbf{\Theta}_1^*, \mathbf{\Theta}_2^*) = \arg\min_{\mathbf{\Theta}_1\in\mathbb{R}^{d_2\times r},\mathbf{\Theta}_2\in\mathbb{R}^{r\times d_1}} \mathcal{L}(\mathcal{D}_{\text{train}}; \mathbf{W}^0 + f(\mathbf{W}^0; (\mathbf{\Theta}_1, \mathbf{\Theta}_2)))$. The conclusion follows from

$$\begin{aligned}&\min_{\mathbf{A}\in\mathbb{R}^{d_1\times r},\mathbf{B}\in\mathbb{R}^{r\times d_2}} \mathcal{L}(\mathcal{D}_{\text{train}}; \mathbf{W}^0 + \mathbf{AB})\\ \le &\mathcal{L}(\mathcal{D}_{\text{train}}; \mathbf{W}^0 + \mathbf{A}^\#\mathbf{B}^\#)\\ = &\mathcal{L}(\mathcal{D}_{\text{train}}; \mathbf{W}^0 + \sigma(\mathbf{W}^0\mathbf{\Theta}_1^*)\mathbf{\Theta}_2^*)\\ = &\min_{\mathbf{\Theta}_1\in\mathbb{R}^{d_2\times r},\mathbf{\Theta}_2\in\mathbb{R}^{r\times d_1}} \mathcal{L}(\mathcal{D}_{\text{train}}; \mathbf{W}^0 + f(\mathbf{W}^0; (\mathbf{\Theta}_1, \mathbf{\Theta}_2))).\end{aligned}$$

□

A.2 Theoretical Analysis of NEAT under sinusoid activation function

Here we consider a sinusoid activation function $\sigma_{\text{p}}(x) = \sin(2\pi x)$ (Gashler and Ashmore, 2014) and design $f(\mathbf{W}^0; \boldsymbol{\theta}) = \sigma_{\text{p}}(\mathbf{W}^0\mathbf{\Theta}_1)\mathbf{\Theta}_2$ with $\boldsymbol{\theta} = (\mathbf{\Theta}_1, \mathbf{\Theta}_2)$. With this periodic activation function, we can show a stronger result that NEAT has expressivity (almost) greater than or equal to a LoRA with more parameters when $d_1 \gg d_2$.

Proposition A.1 (Expressivity of NEAT with Sine Activation)**.** *Suppose that there exists a row of* $\mathbf{W}^0$*, whose entries are linearly independent over the rationals. Then, for any* $r > 0$*,* $\mathbf{A} \in \mathbb{R}^{d_1\times r}$ *and* $\mathbf{B} \in \mathbb{R}^{r\times d_2}$*, and* $\epsilon > 0$*, there exists some* $\mathbf{\Theta}_1^* \in \mathbb{R}^{d_2\times r}$ *and* $\mathbf{\Theta}_2^* \in \mathbb{R}^{r\times d_2}$ *such that*

$$\|\mathbf{AB} - \sigma_{\text{p}}(\mathbf{W}^0\mathbf{\Theta}_1^*)\mathbf{\Theta}_2^*\|_{\text{F}} \le \epsilon.$$

Proposition A.1 shows that the class of updates $\Delta\mathbf{W} = \sigma_{\text{p}}(\mathbf{W}^0\mathbf{\Theta}_1)\mathbf{\Theta}_2$ by NEAT with $2rd_2$ parameters is dense in the class of updates $\Delta\mathbf{W} = \mathbf{AB}$ by LoRA with $r(d_1 + d_2)$ parameters. When $d_2 \ll d_1$, this shows better parameter efficiency of NEAT.

Examining the proof of Proposition A.1, it is straightforward to show that the result holds for any continuous and periodic activation function whose range contains an open interval centered at 0.

Proof. This proof relies on Kronecker's theorem (Theorem 7.9 in Apostol (1990)) from number theory, which shows that for all $j \in \mathbb{R}^q$, the fractional parts of $(ct_1, ct_2, \ldots, ct_q)^\top$ is dense in $[0,1]^q$ over $c \in \mathbb{R}$, as long as $t_1, \ldots, t_q$ are linearly independent over the rationals.

Let $\mathbf{W}_{j^*}$ be the j^*-th column of $\mathbf{W}^0$ whose entries are linearly independent over the rationals. Since $\mathbf{AB}$ has a scale ambiguity, we can assume that $\mathbf{A}$ is a matrix whose entries are bounded by 1 without loss of generality. Write $\mathbf{A} = (\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_r)$.

Take $\epsilon' > 0$ whose value will be determined later. From Kronecker's theorem, for each $\mathbf{A}_j$ there exists some $c_j \in \mathbb{R}$ such that

$$\left| \{c_j \mathbf{W}_{j^*}\} - \frac{\arcsin(\mathbf{A}_j)}{2\pi} \right| \leq \epsilon',$$

where $\{\mathbf{B}\}$ is a vector whose entries are the fractional part of the corresponding entry of $\mathbf{B}$, and $\arcsin$ is applied elementwisely.

Let $\mathbf{\Theta}_1^* = (c_1 \boldsymbol{e}_{j^*}, c_2 \boldsymbol{e}_{j^*}, \ldots, c_r \boldsymbol{e}_{j^*})$, where $\boldsymbol{e}_{j^*}$ is the j^*-th standard basis vector in $\mathbb{R}^{d_2}$. Using the fact that $2\pi\{c_j \mathbf{W}_{j^*}\} = 2\pi c_j \mathbf{W}_{j^*} \mod 2\pi$, we have

$$\begin{aligned} &\left\|\sigma_{\text{p}}(\mathbf{W}^0 \mathbf{\Theta}_1^*) - \mathbf{A}\right\|_{\text{F}}^2 \\ &= \left\|\sigma_{\text{p}}((c_1 \mathbf{W}_{j^*}, c_2 \mathbf{W}_{j^*}, \ldots c_r \mathbf{W}_{j^*})) - \mathbf{A}\right\|_{\text{F}}^2 \\ &\leq \sum_j \|\sin(2\pi c_j \mathbf{W}_{j^*}) - \mathbf{A}_j\|^2 \leq 4\pi^2 r \epsilon'^2, \quad (2) \end{aligned}$$

where the last inequality follows from equation 2 and the fact that $\sin(x)$ is Lipschitz continuous with Lipschitz constant 1. Hence by choosing $\mathbf{\Theta}_2^* \leftarrow \mathbf{B}$, we have

$$\begin{aligned} &\left\|\mathbf{A}\mathbf{B} - \sigma_{\text{p}}(\mathbf{W}^0 \mathbf{\Theta}_1^*) \mathbf{\Theta}_2^*\right\|_{\text{F}}^2 \\ \leq &\|\mathbf{B}\|^2 \left\|\sigma_{\text{p}}(\mathbf{W}^0 \mathbf{\Theta}_1^*) - \mathbf{A}\right\|_{\text{F}}^2 \\ \leq &4\pi^2 \|\mathbf{B}\|^2 r \epsilon'^2. \end{aligned}$$

Choose $\epsilon' = \epsilon / (2\pi \sqrt{r} \|\mathbf{B}\|)$, then the proof is complete. □

B Additional Related Work

B.1 Additive PEFT Methods

Additive PEFT methods (Chronopoulou et al., 2023; Edalati et al., 2022; Lester et al., 2021; Wang et al., 2024c; Liu et al., 2022) introduces a small set of additional trainable parameters strategically placed within the model. One of the most prominent additive PEFT approaches is Adapter (Chronopoulou et al., 2023; Edalati et al., 2022; Zhao et al., 2022), which involves inserting small adapter layers between pre-trained weight blocks. Prompt Tuning (Wang et al., 2024c; Lester et al., 2021; Vu et al., 2021; Li and Liang, 2021) is another technique, where learnable vectors, or "soft prompts," are prepended to the input sequence without modifying the model's weights. This method is particularly effective for large-scale models and has inspired variants such as Prefix Tuning (Li and Liang, 2021).

B.2 Selective PEFT Methods

Selective PEFT focuses on optimizing the fine-tuning process by selectively adjusting a subset of the model's parameters rather than introducing additional ones. For instance, Diff Pruning (Guo et al., 2020) uses a learnable binary mask to select parameters for fine-tuning. Similarly, FishMask (Sung et al., 2021) and Fish-Dip (Das et al., 2023) leverage Fisher information to determine parameter importance and identify the most crucial ones for updates. Additionally, BitFit (Zaken et al., 2021) fine-tunes only the bias terms in the model, significantly reducing the number of trainable parameters.

B.3 Hybrid PEFT method

Hybrid PEFT methods aim to combine the strengths of various existing PEFT techniques to enhance model performance across diverse tasks. UniPELT (Mao et al., 2021) integrates LoRA, prefix-tuning, and adapters within each Transformer block, employing a gating mechanism to determine which module should be active during fine-tuning. S4 (Chen et al., 2023) further explores the design space by partitioning layers into groups and assigning different PEFT methods to each group. Additionally, NOAH (Zhang et al., 2022) and AUTOPEFT (Zhou et al., 2024) leverage neural architecture search (NAS) to automatically discover optimal combinations of PEFT techniques tailored to specific tasks.

C Baselines

Our baselines are constructed on a task basis. Specifically, for each task, the proposed NEAT is compared with representative baselines from the corresponding domain, as listed below.

- For both Commonsense Reasoning and Arithmetic Understanding, following Wang et al. (2024a), LoRA (Hu et al., 2021a), PiSSA (Meng et al., 2024) and MiLoRA (Wang et al., 2024a) are employed as baselines. NEAT is applied to query, key, value, MLP up and MLP down layers.

- For Natural Language Understanding, we follow the setup from prior works (Gao et al., 2024; Wu et al., 2024b) that evaluate various representative PEFT methods, including LoRA (Hu et al., 2021a), Adapter (Houlsby et al., 2019), BitFit (Zaken et al., 2021), RED (Wu et al., 2024a), DoRA (Liu et al., 2024), ReFT (Wu et al., 2024b), and FourierFT (Gao et al., 2024).

- For Image Classification, we follow the setting of Gao et al. (2024) and take linear probing (LP), LoRA (Hu et al., 2021a) and FourierFT (Gao et al., 2024) as baselines. NEAT is applied to the query and value layers.

D Hyperparameters

We provide the specific hyperparameters used in our experiments to ensure reproducibility. For most of our experiments, we use the standard implementation of NEAT, which we refer to as vanilla NEAT. The neural network architecture in vanilla NEAT consists of only two layers: an input layer and an output layer. We selecte this approach because vanilla NEAT offers the benefits of simplicity in implementation, a low parameter count, and sufficient adaptation power. Nonetheless, we dedicate Section 4.4 and Appendix F to exploring more complex adaptation networks and their effect on performance.

D.1 Image Classification

Hyperparameters for NEAT for Fig. 4 are provided in Table 6. We tune the classification head and the backbone separately and provide detailed settings for each dataset. All weight decay values are not tuned and follow the settings from Gao et al. (2024). The scaling factor s is set to 1.0. The hidden layer dimension r for MHSA is set to 7 in the QV-setting, while both hidden layer dimensions for MHSA and MLP are set to 2 in the QV-MLP-setting described in Section 5.

D.2 Natural Language Understanding

We provide used hyper-parameters for NEAT in natural language understanding on the GLUE benchmark in Table 7 and Table 8. The reported results are obtained when using a depth of 6 for NEAT. The learning rates for the head and the backbone are tuned separately. The scaling factor s is searched in $\{0.01, 0.1, 1.0\}$. For reproducibility, we fix the seed as 0. The hidden layer dimension r is set to 8 in NEAT-L and 1 in NEAT-S. More specifically, we apply NEAT to all layers in RoBERTa-base for NEAT-L, while only applying NEAT to layers $\{4, 5, 6, 7, 8, 9, 10, 11\}$ for NEAT-S to reduce the number of trainable parameters. The seed is fixed for reproducibility.

D.3 Commonsense Reasoning

We provide hyperparameters settings of NEAT for commonsense reasoning task in Table 9. We follow the hyperparameters settings in MiLoRA (Wang et al., 2024a). We limit all samples to a maximum of 256 tokens. For evaluation, we set a maximum token number of 32.

D.4 Arithmetic Reasoning

We provide hyperparameters settings of NEAT for arithmetic reasoning task in Table 10. We follow the hyper-parameters settings in MiLoRA (Wang et al., 2024a). We limit all samples to a maximum of 2048 tokens. For evaluation, we set a maximum token number of 256 on GSM8K (Cobbe et al., 2021) dataset. On MATH (Hendrycks et al., 2021), we set the maximum new token to 512.

E Datasets

In this section, we provide a detailed description of the datasets used in our experiments.

E.1 Image Classification

For image classification, we provide detailed information about the used datasets in Table 11.

E.2 Natural Language Understanding

The GLUE benchmark comprises 8 NLP datasets: MNLI, SST-2, MRPC, CoLA, QNLI, QQP, RTE, and STS-B, covering tasks such as inference, sentiment analysis, paraphrase detection, linguistic acceptability, question-answering, and textual similarity. We provide detailed information about them in Table 12.

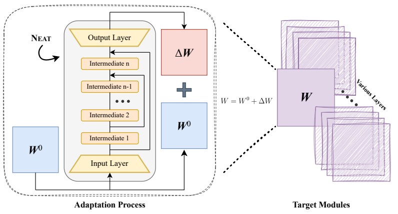

Figure 5: Implementation of introducing more depths to NEATt. We insert multiple intermediate layers into the layers from vanilla NEAT, with non-linear activation in between. The depth is described as the number of layers in NEAT, with vanilla NEAT having a depth of 2 (i.e. the input and output layers).

E.3 Commonsense Reasoning

For commonsense reasoning task, we use 8 datasets, including BoolQ, PIQA, SIQA, Hel-

Table 6: Hyperparameter of image classification for NEAT.

Hyperparameter	OxfordPets	StanfordCars	CIFAR10	DTD	EuroSAT	FGVC	RESISC45	CIFAR100
Epochs	10							
Optimizer	AdamW							
LR Schedule	Linear							
Weight Decay	8E-4	4E-5	9E-5	7E-5	3E-4	7E-5	3E-4	1E-4
QV								
Learning Rate (NEAT)	5E-3	1E-2	5E-3	1E-2	5E-3	1E-2	5E-3	5E-3
Learning Rate (Head)	5E-3	1E-2	5E-3	1E-2	5E-3	1E-2	1E-2	5E-3
QV-MLP								
Learning Rate (NEAT)	5E-3	5E-3	5E-3	1E-2	5E-3	5E-3	1E-2	5E-3
Learning Rate (Head)	5E-3	1E-2	5E-3	1E-2	5E-3	1E-2	1E-2	5E-3

Table 7: Hyperparameter of GLUE benchmark for NEAT-L.

Hyperparameter	STS-B	RTE	MRPC	CoLA	SST-2	QNLI	MNLI	QQP
Optimizer	AdamW							
LR Schedule	Linear							
Learning Rate (NEAT)	5E-3	5E-3	5E-3	1E-3	5E-3	1E-3	5E-3	5E-3
Learning Rate (Head)	5E-3	5E-3	5E-3	1E-3	5E-3	1E-3	5E-3	5E-3
Scaling	0.1	0.01	0.01	0.1	0.01	0.01	0.01	0.01
Max Seq. Len	512	512	512	512	512	512	512	512
Batch Size	64	32	64	64	32	32	32	64

laSwag, WinoGrande, ARC-e, ARC-c and OBQA. The detailed information is provided in Table 13.

E.4 Arithmetic Reasoning

Detailed information for arithmetic reasoning task is provided in Table 14. GSM8K consists of high quality grade school math problems, typically free-form answers. MATH includes classifications from multiple mathematical domains, such as algebra, counting_and_probability, geometry, intermediate_algebra, number_theory, prealgebra and precalculus.

F Depths of NEAT

We provide a comprehensive explanation of our approach to increasing the depth of the adaptation neural network in NEAT. As depicted in Fig. 5, we introduce multiple deeply stacked intermediate layers between the layers of the vanilla NEAT. These intermediate layers are essentially small adapters with a minimal parameter count ($\mathbb{R}^{r\times r}$, where r is the hidden layer dimension), and we retain non-linear activations between them, as proposed by NEAT. The adaptation process begins by feeding the weight matrix $\mathbf{W}^0$—the initialized value of the adaptation target $\mathbf{W}$—into NEAT's input layer. After undergoing multiple non-linear transformations through the intermediate layers, the final layer projects $\mathbf{W}^0$ back to its original shape, producing the adaptation result ΔW. Throughout this process, the adaptation target remains fixed, while all the intermediate layers, as well as the input and output layers in NEAT, are trainable parameters.

Furthermore, an implementation example of NEAT with four layers using the PyTorch library is illustrated in Fig. 6. As previously mentioned, we apply non-linear activations (ReLU in this case) to model more complex transformations. The intermediate layers have the same shape, $\mathbb{R}^{r\times r}$, which adds minimal overhead compared to $\mathbf{A} \in \mathbb{R}^{d_2\times r}$ and $B \in \mathbb{R}^{r\times d_2}$—the input and output layers, respectively, which are also present in the vanilla NEAT. Since d_2 is typically in the range of hundreds to thousands, while r is commonly set to 8,

Table 8: Hyperparameter of GLUE benchmark for Neat-S.

Hyperparameter	STS-B	RTE	MRPC	CoLA	SST-2	QNLI	MNLI	QQP
Optimizer	AdamW							
LR Schedule	Linear							
Learning Rate (Neat)	5E-3	1E-3	5E-3	5E-3	5E-3	1E-3	5E-3	1E-3
Learning Rate (Head)	1E-3	1E-3	5E-3	1E-3	5E-3	1E-3	5E-3	1E-3
Scaling	0.1	1.0	0.01	0.1	0.01	0.1	0.01	1.0
Max Seq. Len	512	512	512	512	512	512	512	512
Batch Size	64	32	64	64	32	32	32	64

```python
class neat_depth_four(nn.Module):
    """
    Example of 4-layer implementation for Neat with residual.
    Using ReLU as the default non-linear activation function.
    args:
        dim: hidden dimension (a.k.a. rank)
        out_dim: output dimension
    """
    def __init__(self, dim=32, out_dim=768):
        super().__init__()
        self.non_linear = nn.ReLU()
        self.A = nn.Linear(out_dim, dim, bias=False)
        self.i1 = nn.Linear(dim, dim, bias=False)
        self.i2 = nn.Linear(dim, dim, bias=False) # two intermediate layers
        self.B = nn.Linear(dim, out_dim, bias=False)
        nn.init.zeros_(self.B.weight)

    def forward(self, x, weight):
        delta_w = self.non_linear(weight @ self.A.weight.t())  # non-linear(W_0 A)
        residual = delta_w.clone()
        delta_w = self.non_linear(self.i1_(delta_w))
        delta_w = self.non_linear(self.i2_(delta_w))
        delta_w = delta_w + residual
        delta_w = self.B(delta_w) # obtain the approximated delta W
        return x @ delta_w
```

Figure 6: An example of the actual implementation applying 4 layers in Neat (depth = 4) with Pytorch.

Table 9: Hyperparameter of commonsense reasoning for Neat.

Hyperparameter	Commonsense Reasoning
Hidden Layer Dimension	32
α	32
Dropout	0.05
Optimizer	Adam W
Learning Rate	3e-4
Batch Size	16
Warmup Steps	100
Epochs	1

Table 10: Hyperparameter of arithmetic reasoning for Neat.

Hyperparameter	Arithmetic Reasoning
Hidden Layer Dimension	64
α	64
Dropout	0.05
Optimizer	Adam W
Learning Rate	3e-4
Batch Size	16
Warmup Steps	100
Epochs	3

16, or 32, the parameter efficiency of Neat with deeper layers remains comparable to that of vanilla Neat without the intermediate layers. As shown, we first transform $\mathbf{W}^0$ into the desired adaptation result ΔW and subsequently use ΔW to perform the actual computation on the input data. The use of residuals is based on empirical observations, as incorporating residual connections in the adaptation process results in faster convergence, more stable loss curves, and significantly improved overall performance.

Meanwhile, to provide a comprehensive understanding of the impact of depth on model performance, we present the results of different depth configurations on the GLUE benchmark in Tab 15.

Table 11: Detailed information of image classification tasks.

Dataset	#Class	#Train	#Val	#Test	Rescaled resolution
OxfordPets	37	3,312	368	3,669	224 × 224
StandfordCars	196	7,329	815	8,041	
CIFAR10	10	45,000	5,000	10,000	
DTD	47	4,060	452	1,128	
EuroSAT	10	16,200	5,400	5,400	
FGVC	100	3,000	334	3,333	
RESISC45	45	18,900	6,300	6,300	
CIFAR100	100	45,000	5,000	10,000	

Table 12: Detailed information of the GLUE benchmark. STS-B is a regression task, while all other tasks are either single-sentence or sentence-pair classification tasks.

Corpus	**Task**	**Metrics**	**# Train**	**# Val**	**# Test**	**# Labels**
Single-Sentence Tasks						
CoLA	Acceptability	Matthews Corr.	8.55k	1.04k	1.06k	2
SST-2	Sentiment	Accuracy	67.3k	872	1.82k	2
Similarity and Paraphrase Tasks						
MRPC	Paraphrase	Accuracy/F1	3.67k	408	1.73k	2
STS-B	Sentence similarity	Pearson/Spearman Corr.	5.75k	1.5k	1.38k	1
QQP	Paraphrase	Accuracy/F1	364k	40.4k	391k	2
Inference Tasks						
MNLI	NLI	Accuracy	393k	19.65k	19.65k	3
QNLI	QA/NLI	Accuracy	105k	5.46k	5.46k	2
RTE	NLI	Accuracy	2.49k	277	3k	2

Table 13: Detailed information of commonsense reasoning task.

Dataset	#Class	#Train	#Dev	#Test
BoolQ	Binary classification	9,427	3,270	3,245
PIQA	Binary classification	16,113	1,838	3,000
SIQA	Ternary classification	33,410	1,954	2,224
HellaSwag	Quaternary classification	39,905	10,042	10,003
WinoGrande	Binary classification	40,398	1,267	1,767
ARC-e	Quaternary classification	2,251	570	2,376
ARC-c	Quaternary classification	1,119	229	1,172
OBQA	Quaternary classification	4,957	500	500

Table 14: Detailed information of arithmetic reasoning task.

Dataset	#Train	#Dev	#Test
GSM8K	7,473	1,319	1,319
MATH	12,500	500	5,000

Our analysis reveals that increasing the number of layers in the neural network architecture introduced by NEAT consistently enhances performance while maintaining the same level of parameter efficiency. Furthermore, we extend our experiments to vision tasks, where even deeper networks are explored, and summarize the results in Fig. 7. We find that when the depth is increased to 8 and 10 layers, the accuracy slightly decreases compared to the 6-layer model but remains higher than that of the 2-layer configuration. A possible explanation is that as depth increases—particularly at 10 layers—the training process becomes more challenging, possibly requiring more careful hyper-parameter tuning. In general, these findings demonstrate that increasing the depth of the network is consistently beneficial for improving performance across both language and vision tasks.

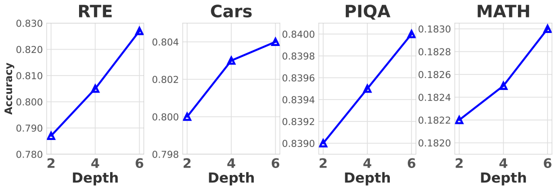

Figure 7: Accuracy on the StanfordCars, FGVC and Oxford-Pets dataset with a wider range of depths for the neural network used in NEAT. We choose depth equals to 2, 4, 6, 8 and 10 layers in the figure. In many cases, NEAT with more depths outperforms vanilla NEAT.

Table 15: Accuracy comparison of NEAT using RoBERTa-base with different depth configurations on the GLUE benchmark. The highest accuracy of methods per category are in **bold**. “AVG” means the average accuracy of all datasets.

depth	**Params** (%)	**Accuracy** (↑)								
		MNLI	**SST-2**	**MRPC**	**CoLA**	**QNLI**	**QQP**	**RTE**	**STS-B**	**AVG**
2	0.239%	86.6	94.6	90.0	64.4	**92.7**	89.7	78.7	**90.9**	86.0
4	0.239%	86.7	94.5	**90.2**	**65.1**	92.4	**90.5**	80.5	90.8	86.3
6	0.241%	**86.9**	**95.2**	90.0	64.8	92.3	90.3	**82.7**	90.7	**86.6**