Flat-LoRA: Low-Rank Adaption over a Flat Loss Landscape

Tao Li[1], **Zhengbao He**[1]*, **Yujun Li**[2], **Yasheng Wang**[2], **Lifeng Shang**[2], **Xiaolin Huang**[1]
[1] Department of Automation, Shanghai Jiao Tong University
[2]Huawei Noah's Ark Lab
`{li.tao, xiaolinhuang}@sjtu.edu.edu`

Abstract

Fine-tuning large-scale pre-trained models is prohibitively expensive in terms of computational and memory costs. Low-Rank Adaptation (LoRA), a popular Parameter-Efficient Fine-Tuning (PEFT) method, provides an efficient way to fine-tune models by optimizing only a low-rank matrix. Despite recent progress made in improving LoRA's performance, the connection between the LoRA optimization space and the original full parameter space is often overlooked. A solution that appears flat in the LoRA space may exist sharp directions in the full parameter space, potentially harming generalization performance. In this paper, we propose Flat-LoRA, an efficient approach that seeks a low-rank adaptation located in a flat region of the full parameter space. Instead of relying on the well-established sharpness-aware minimization approach, which can incur significant computational and memory burdens, we utilize random weight perturbation with a Bayesian expectation loss objective to maintain training efficiency and design a refined perturbation generation strategy for improved performance. Experiments on natural language processing and image classification tasks with various architectures demonstrate the effectiveness of our approach.

1 Introduction

Pre-training followed by fine-tuning is a widely adopted training pipeline among modern machine learning practitioners for achieving state-of-the-art (SOTA) performance [1, 2, 3, 4], leveraging the versatile knowledge within the pre-trained models. However, the enormous size of these pre-trained models makes fine-tuning all parameters for downstream tasks resource-intensive, making it impractical to store optimizer states or multiple model weights when dealing with multiple tasks. Recently, Low-Rank Adaptation (LoRA) [5] has been proposed to address this resource challenge. In LoRA fine-tuning, only a low-rank matrix is optimized and then added to the pre-trained weights after training, incurring no additional computational or memory costs during inference. This approach significantly reduces the number of trainable parameters, thereby lowering the training cost as well as storage cost when dealing with different tasks.

Many works have been proposed to enhance the performance of LoRA by introducing more dedicated budgets for rank allocation [6], decomposing optimization for direction and magnitude updates [7], or designing better initialization strategy for LoRA parameters [8, 9], etc. These studies demonstrate the significant potential for improving LoRA performance. However, the connection between the LoRA optimization space and the original full parameter space is often overlooked. Essentially, LoRA restricts training to a much lower-dimensional subspace, and its performance depends on the properties of the solutions within this subspace in relation to the full parameter space, as the merged

*The first two authors contribute equally.

weights are ultimately used during inference. As illustrated in Figure 1, a flat minima in the LoRA space (blue) may exhibit sharp direction (red) in the view of the full parameter space, which may degenerate the generation performance.

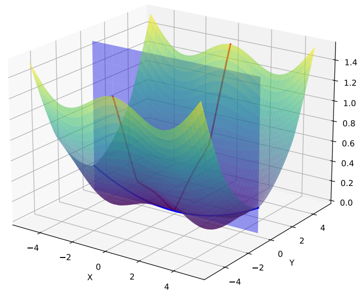


Figure 1: Illustration of LoRA optimization. LoRA constrains training in a lower-dimensional subspace (blue). A flat minima in LoRA subspace (blue curve) may exhibit sharp direction in full parameter space (red curve).

It is widely believed that a flatter loss landscape can lead to better generalization performance [10, 11]. This idea has given rise to a well-established training strategy called Sharpness-Aware Minimization (SAM), which has shown great generalization improvement in training neural networks. Applying SAM to large language models (LLMs)' training together with LoRA is certainly promising, but there are several issues should be discussed. First, unlike the existing attempts that flatten the landscape in a LoRA subspace [12], which is not aware of the sharpness outside the LoRA space, we pursue a solution that aligns with a flatter loss landscape in the full weight space. Second, the original SAM doubles the training time cost, which is impractical for fine-tuning large models. Additionally, to capture the sharpness in the full parameter space, we need to calculate the gradients of the full weight parameters, which contradicts the principles of parameter-efficient fine-tuning (PEFT). To cope with these challenges, we propose using random weight perturbations to maintain time and memory efficiency and design effective generation strategies to improve generalization performance.

Our main contribution can be summarized as follows:

- We propose Flat-LoRA that firstly aims to optimize the sharpness of the loss landscape within the full parameter space where the low-rank adaptation resides. It incurs minimal additional computational and memory costs and can be easily integrated with existing techniques to enhance LoRA performance, delivering consistent improvements.
- We propose to use expected Bayesian loss to optimize the sharpness for keeping the training efficiency and design effective generation strategy to generate random weight perturbation to enhance the generalization performance, making it easy for practical usage.
- Experiments on natural language processing and computer vision tasks with various architectures to demonstrate that our approach can achieve state-of-the-art performance.

2 Related Work

Flat minima and generalization. The connection between the flatness of local minima and generalization has received much attention [11, 13, 14, 15, 16, 17, 18, 19, 20, 21]. Recently, many works have tried to improve the model generalization by seeking flat minima [22, 23, 24]. For example, [13] proposed Entropy-SGD to search for flat regions by minimizing local entropy. [25] designed SmoothOut framework to smooth out the sharp minima. Notably, Sharpness-Aware Minimization (SAM) [26] established a generic training scheme for seeking flat minima by formulating the optimization as a min-max problem and encouraged parameters sitting in neighborhoods with uniformly low loss, achieving state-of-the-art generalization improvements across various tasks. However, it requires twice the training time as regular training, limiting its application to large model training. Another promising branch of methods recovers flat minima by minimizing the expected Bayesian training loss under random weight perturbation (RWP) [24], which is efficient as no additional gradient step is required. [27] further enhance generalization performance by introducing an adaptive random perturbation generation strategy and a mixed loss objective. However, when applying these approaches to PEFT training, we must be mindful of the additional memory and time costs they may introduce.

Low-rank adaption. Recent works have indicated that the intrinsic dimension for optimizing deep neural networks (DNNs) may be much lower than the number of parameters [28, 29, 30]. [31] demonstrated that the training trajectory of DNNs can be low-dimensional and propose subspace optimization to enhance training efficiency and robustness [32]. Low-Rank Adaptation (LoRA) was proposed to model the weight changes for each layer during fine-tuning, aiming to reduce training costs. It effectively decreases the number of trainable parameters, thereby lowering the memory burden for training and storage. This approach is currently mainstream because it avoids adding any overhead during inference while often demonstrating strong performance. Many works have been proposed to enhance the performance of LoRA by introducing more dedicated budgets for rank allocation [6], decomposing optimization for direction and magnitude updates [7], designing better initialization strategy for LoRA parameters [8, 9], or better aligning each gradient step to the full fine-tuning [33], etc.

3 Method

In this section, we first give a brief review on the low-rank adaption (LoRA). We then introduce our LoRA optimization objective considering the flatness of the landscape. We finally describe our random perturbation generation strategy for effectively improving the generalization performance.

3.1 LoRA: Low-Rank Adaption

Based on the finding that DNNs' optimization happens in a subspace with much smaller dimensions than the number of parameters [28, 31], LoRA utilizes low-rank matrices to model the weight change for each layers' weights $\mathbf{W} \in \mathbb{R}^{n\times m}$ during the fine-tuning as $\Delta\mathbf{W} = s \cdot \mathbf{BA}$, where s is a scaling factor, $\mathbf{B} \in \mathbb{R}^{n\times r}$ and $\mathbf{A} \in \mathbb{R}^{r\times m}$ with the rank $r \ll \{n, m\}$ to achieve parameter efficiency.

For the original output $\mathbf{h} = \mathbf{Wx}$, the modified forward pass is

$$\mathbf{h} = \mathbf{Wx} + \Delta\mathbf{Wx} = (\mathbf{W} + s \cdot \mathbf{BA})\mathbf{x}. \tag{1}$$

During the initialization of LoRA, matrix $\mathbf{A}$ is commonly initialized with Kaiming distribution [34] and matrix $\mathbf{B}$ is set to zeros. During the training, only the low-rank matrices $\mathbf{A}$ and $\mathbf{B}$ are optimized with the pre-trained weight $\mathbf{W}$ being frozen. During the inference, the low-rank matrices $\Delta\mathbf{W}$ are merged to the pre-trained weight $\mathbf{W}$, and in this way there is no additional computational or memory costs.

3.2 LoRA with a Flat Landscape

Despite recent efforts to improve LoRA performance, most studies focus solely on finding solutions that perform well within the LoRA solution space, specifically the rank r matrix space $\mathcal{M}_r = \{\Delta\mathbf{W} \in \mathbf{R}^{m\times n} \mid \operatorname{rank}(\Delta\mathbf{W}) = r\}$. For example, following the well-established sharpness-aware minimization (SAM) objective [26], [12] apply SAM to LoRA parameters and study the scale-invariant properties of these parameters with SAM:

$$\min_{\mathbf{A},\mathbf{B}} \max_{\|(\epsilon_\mathbf{A},\epsilon_\mathbf{B})\|\le\rho} L\left(\mathbf{W} + (\mathbf{B} + \epsilon_\mathbf{B})(\mathbf{A} + \epsilon_\mathbf{A})\right), \tag{2}$$

where $L(\cdot)$ denotes the loss objective. However, focusing solely on the properties of the optimization space defined by LoRA parameters may have limitations. During inference, the low-rank adaption $\Delta\mathbf{W}$ is merged into the pre-trained weights $\mathbf{W}$. A solution that performs well within the LoRA space may be situated in a sharp region of the full parameter space, as illustrated in Figure 1, which could potentially harm overall generalization. To be more clear, we have

$$\nabla L_{\epsilon_\mathbf{A}}(\mathbf{W}) = \mathbf{B}^\top \nabla L(\mathbf{W}), \tag{3}$$

$$\nabla L_{\epsilon_\mathbf{B}}(\mathbf{W}) = \nabla L(\mathbf{W})\mathbf{A}^\top, \tag{4}$$

and then we can obtain the worst case perturbation $\epsilon_\mathbf{A}$ and $\epsilon_\mathbf{B}$ via first-order Taylor expansion:

$$\epsilon_\mathbf{A} = \rho \frac{\mathbf{B}^\top \nabla L(\mathbf{W})}{\sqrt{\|\mathbf{B}^\top \nabla L(\mathbf{W})\|^2 + \|\nabla L(\mathbf{W})\mathbf{A}^\top\|^2}}, \tag{5}$$

$$\epsilon_\mathbf{B} = \rho \frac{\nabla L(\mathbf{W})\mathbf{A}^\top}{\sqrt{\|\mathbf{B}^\top \nabla L(\mathbf{W})\|^2 + \|\nabla L(\mathbf{W})\mathbf{A}^\top\|^2}}, \tag{6}$$

The equivalent weight perturbation applied to $\mathbf{W}$ by Equ (2) is

$$\mathbf{B}\epsilon_{\mathbf{A}} + \epsilon_{\mathbf{B}}\mathbf{A} + \epsilon_{\mathbf{B}}\epsilon_{\mathbf{A}} = c\mathbf{B}\mathbf{B}^\top \nabla L(\mathbf{W}) + c\nabla L(\mathbf{W})\mathbf{A}^\top\mathbf{A} + c^2\nabla L(\mathbf{W})\mathbf{A}^\top\mathbf{B}^\top\nabla L(\mathbf{W}), \quad (7)$$

where $c = \frac{\rho}{\sqrt{\|\mathbf{B}^\top\nabla L(\mathbf{W})\|^2 + \|\nabla L(\mathbf{W})\mathbf{A}^\top\|^2}}$ is a scaling factor. One can see that the perturbation direction is not aligned with the direction $\nabla L(\mathbf{W})$, which maximizes the loss of the merged weights as in SAM. Notably, when $\mathbf{B}$ is initialized as zero as defaulted in [5], $\mathbf{B}$ will remain small during the training [35] and Equ. (7) becomes:

$$\mathbf{B}\epsilon_{\mathbf{A}} + \epsilon_{\mathbf{B}}\mathbf{A} + \epsilon_{\mathbf{B}}\epsilon_{\mathbf{A}} \approx c\nabla L(\mathbf{W})\mathbf{A}^\top\mathbf{A}. \quad (8)$$

This means Equ (2) only optimizes the sharpness along the column space spanned by $\mathbf{A}$, which constitutes a small subspace of the full parameter space. As demonstrated in Table 4, solely applying SAM constraints on the LoRA parameters does not effectively improve the generalization.

Therefore, it is crucial to consider the loss landscape of $L(\mathbf{W} + \Delta\mathbf{W})$, and we need to find a low rank adaption $\Delta\mathbf{W}$ that positions the merged weights in a flat region of the full parameter space. Our flat loss objective can be formulated as follows:

$$\min_{\mathbf{A},\mathbf{B}} \max_{\|\boldsymbol{\epsilon}\|\le\rho} L(\mathbf{W} + \mathbf{B}\mathbf{A} + \boldsymbol{\epsilon}). \quad (9)$$

However, directly applying SAM to optimize the sharpness of the merged weight space has several disadvantages: 1) it doubles the training cost, which is less desirable with large models, and 2) it requires storing an additional copy of weights for perturbation, which contradicts the principle of parameter-efficient fine-tuning. To achieve a flatter loss landscape while maintaining time and memory efficiency, we propose relaxing the maximization problem in Eq. (9) to an expectation, resulting in the following Bayesian expected loss objective:

$$\min_{\mathbf{A},\mathbf{B}} \mathop{\mathbb{E}}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,\sigma^2\mathbf{I})} L(\mathbf{W} + \mathbf{B}\mathbf{A} + \boldsymbol{\epsilon}), \quad (10)$$

where σ controls the variance magnitude of the noise, which we will describe in the next section. This expected loss can be seen as applying a smoothing filter over the loss landscape within the full parameter space, and optimizing it can help recover flatter minima [24]. For each optimization step, we would sample a noise $\boldsymbol{\epsilon}$ and calculate the perturbed gradient to optimize the low-rank matrices $\mathbf{A}$ and $\mathbf{B}$. Note that the noise is generated based on the model weights, thus incurring no additional gradient steps as SAM does.

3.3 Effective Random Perturbation Generation

We then describe how to effectively generate random weight perturbation, which is the core for improving the generalization performance. Let $\mathbf{W}' = \mathbf{W} + s \cdot \mathbf{B}\mathbf{A}$. In this paper, we have experimented with various schemes for noise generation and hope to find the most effective one for random weight perturbation generation that could benefit the community. For the merged weight $\mathbf{W}' \in \mathbb{R}^{m\times n}$ that represents a linear layer with input dimension n and output dimension m, our design considers the following two perspectives:

- Filter structure: we aim to generate the weight noise by filter [24]. There contains m filters $\mathbf{W}' = (\mathbf{W}'_{1,:}, \mathbf{W}'_{2,:}, \cdots, \mathbf{W}'_{m,:})$ that process the input $\mathbf{X} \in \mathbb{R}^n$. Elements within a filter of larger norm should receive a larger strength of perturbation.
- Input dimension: the variance introduced to the forward pass by the added random weight perturbation is independent of the input dimension, which helps transfer hyper-parameters across models with different widths. Given an input dimension n, the magnitude of noise added to each element should be scaled by a factor of $1/n$.

Finally, our new weight noise generation scheme is formulated as follows:

$$\boldsymbol{\epsilon} \sim \mathcal{N}\left(0, \frac{\sigma^2}{n}\mathrm{diag}(\|\mathbf{W}'_{1,:}\|_2^2, \|\mathbf{W}'_{2,:}\|_2^2, \cdots, \|\mathbf{W}'_{m,:}\|_2^2)\mathbf{I}_{m\times n}\right), \quad (11)$$

where $\mathbf{I}_{m\times n}$ denotes a matrix of size $m \times n$ with all ones. Here σ is the hyper-parameter that needs to be chosen for controlling the perturbation strength.

Analysis on forward variance. We then interpret adding random weight perturbation as variance injection during the forward propagation. Consider the hypothesis that the variance of the input $\mathbf{X}$ is $\mathrm{var}(\mathbf{X})$. Then we have:

$$\mathrm{var}(\mathbf{W}'_{i,:}\mathbf{X}) = \|\mathbf{W}'_{i,:}\|_2^2 \cdot \mathrm{var}(\mathbf{X}), \tag{12}$$

$$\mathrm{var}\left((\mathbf{W}' + \boldsymbol{\epsilon})_{i,:}\mathbf{X}\right) = (1+\sigma^2)\|\mathbf{W}'_{i,:}\|_2^2 \cdot \mathrm{var}(\mathbf{X}). \tag{13}$$

Thus, by injecting weight perturbations, we introduce variance with a rate of σ^2 into the forward pass. It is important to note that since we introduce a magnitude of $1/n$ for noise generation (see Equ. (11)), this variance ratio is independent of the input dimension. Additionally, we note that this variance will not increase exponentially during the forward propagation of the network due to the presence of layer normalization.

Storing random seed for memory efficiency. Memory is an important factor to consider to PEFT training. To optimize Eqn. (10), we first generate random perturbation $\boldsymbol{\epsilon}$ and then perform gradient descent with $\nabla L(\mathbf{W} + s \cdot \mathbf{BA} + \boldsymbol{\epsilon})$. Thus, we need to store the weight perturbation for recovering the weight after obtaining the perturbed gradient. When model is large, storing a copy of weight perturbation is prohibitive. Luckily, for random weight perturbation, we only need to store the seed for random generator and corresponding norms for each filter $\|\mathbf{W}'_{1,:}\|_2^2, \|\mathbf{W}'_{2,:}\|_2^2, \cdots, \|\mathbf{W}'_{m,:}\|_2^2$, allowing us to recover the random perturbation $\boldsymbol{\epsilon}$ when necessary. This approach incurs minimal additional memory and offers significant advantages over SAM, which requires calculating the full gradient, thereby necessitating a hard copy of the perturbation that cannot be reduced.

Integration to mixed precision training. Finally, we note that the noise injection step can be seamlessly integrated into mixed-precision training even without the need to store the filter norms, which is commonly used in large-scale training. Specifically, in mixed-precision training, we maintain two copies of the weights: the full-precision FP32 weights and the half-precision BF16 weights in memory. We can inject noise during the half-precision auto-cast step, thus eliminating the need to store a copy of the weight perturbation or the filter norms.

4 Experiments

In this section, we evaluate the performance of Flat-LoRA on various benchmark datasets. We first conduct experiments on Natural Language Understanding tasks using a subset of GLUE datasets [36] with T5-base model [37]. We then experiment over image classification tasks with CLIP ViT-B/32 model [38]. We finally give ablation studies and discussions on our method.

4.1 Experiments on Natural Language Understanding

Setting. We finetune the T5-Base model on several datasets from GLUE benchmark, including MNLI, SST, CoLA, QNLI, and MRPC. Performance is evaluated on the development set using accuracy as the primary metric. We use lora with rank 8 and 16 with lora alpha 16. We finetune the models with 10 epochs with a cosine learning rate schedule, except for MNLI and QNLI we use 1 epochs. We use learning rate of 0.0005 for LoRA fine-tuning and 0.0001 for full fine-tuning with weight decay 0.1. The random perturbation strength σ is set to 0.05 with an cosine increasing strategy. Mean and standard deviations are calculated over 3 independent trials.

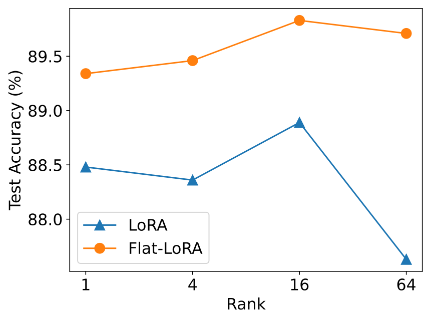
Results. As shown in Table 1, Flat-LoRA consistently outperforms LoRA for ranks 8 and 16, achieving average performance gains of 0.38% and 0.56%, respectively. In some cases, the performance of LoRA deteriorates when increasing the rank from 8 to 16, as seen with the MRPC dataset, which is relatively small and susceptible to overfitting. Flat-LoRA effectively addresses the overfitting issue and achieves greater improvements with increasing LoRA rank, demonstrating the advantages of our flat loss objective.

4.2 Experiments on Image Classification

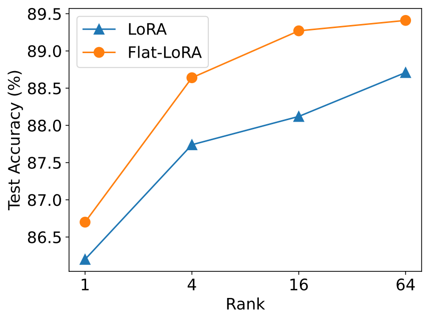
Setting. We finetune the CLIP-ViT-B/32 model on five image classification tasks, including CIFAR-10/100 [39], Cars [40], SVHN [41], and DTD [42]. We resize all input image to a size of 224x224 and freeze the classification head. We try LoRA with rank 8 and 16 and finetune the models with 10

Table 1: Results (%) on fine-tuning T5-base with Full Fine-tuning and LoRA variants on a subset of GLUE datasets.

Method	MNLI	SST2	CoLA	QNLI	MRPC	Avg.
Full FT	$86.19_{\pm 0.04}$	$94.15_{\pm 0.09}$	$82.84_{\pm 0.12}$	$93.10_{\pm 0.04}$	$89.22_{\pm 0.23}$	89.10
LoRA (r=8)	$\mathbf{86.24}_{\pm 0.02}$	$94.55_{\pm 0.07}$	$82.20_{\pm 0.22}$	$93.06_{\pm 0.03}$	$88.97_{\pm 0.42}$	89.00
Flat-LoRA (r=8)	$86.20_{\pm 0.04}$	$\mathbf{94.75}_{\pm 0.20}$	$\mathbf{83.19}_{\pm 0.38}$	$\mathbf{93.16}_{\pm 0.09}$	$\mathbf{89.59}_{\pm 0.37}$	**89.38**
LoRA (r=16)	$86.49_{\pm 0.06}$	$94.52_{\pm 0.21}$	$82.89_{\pm 0.44}$	$92.97_{\pm 0.05}$	$88.89_{\pm 0.64}$	89.15
Flat-LoRA (r=16)	$\mathbf{86.51}_{\pm 0.01}$	$\mathbf{94.84}_{\pm 0.02}$	$\mathbf{84.08}_{\pm 0.31}$	$\mathbf{93.28}_{\pm 0.03}$	$\mathbf{89.83}_{\pm 0.64}$	**89.71**



(a) MRPC with T5-Base



(b) CIFAR-100 with ViT-B/32

Figure 2: Performance comparison under different lora ranks. We keep lora alpha to 16 and vary the lora ranks among $\{1, 4, 16, 64\}$. Experiments are conducted with three independent trials.

epochs with a cosine annealing schedule. The learning rate is set to 0.0005 for LoRA and 1×10^{-5} for full fine-tuning with weight decay 0.1. The random perturbation strength σ is set to 0.15 with an cosine increasing strategy. Mean and standard deviations are calculated over 3 independent trials.

Results. We measure the performance with classification accuracy and report the results in Table 2. We observe that Flat-LoRA consistently outperforms LoRA with ranks 8 and 16, showing average improvements of 0.56% and 0.74%, respectively. Notably, Flat-LoRA with rank 8 surpasses both LoRA with rank 16 and full fine-tuning by 0.28%. These results confirm the effectiveness of our flat loss objective on improving LoRA performance.

Table 2: Results (%) on fine-tuning CLIP ViT-B/32 with full fine-tuning and LoRA variants on image classification datasets.

Method	CIFAR-10	CIFAR-100	Cars	SVHN	DTD	Avg.
Full FT	$97.99_{\pm 0.01}$	$89.06_{\pm 0.11}$	$73.30_{\pm 0.43}$	$97.44_{\pm 0.03}$	$76.80_{\pm 0.25}$	86.92
LoRA (r=8)	$97.90_{\pm 0.02}$	$87.74_{\pm 0.13}$	$73.22_{\pm 0.53}$	$97.49_{\pm 0.08}$	$76.86_{\pm 0.34}$	86.64
Flat-LoRA (r=8)	$\mathbf{98.09}_{\pm 0.04}$	$\mathbf{88.64}_{\pm 0.23}$	$\mathbf{74.17}_{\pm 0.71}$	$\mathbf{97.59}_{\pm 0.04}$	$\mathbf{77.51}_{\pm 0.28}$	**87.20**
LoRA (r=16)	$97.99_{\pm 0.03}$	$88.12_{\pm 0.23}$	$73.80_{\pm 0.42}$	$97.56_{\pm 0.08}$	$77.34_{\pm 0.32}$	86.92
Flat-LoRA (r=16)	$\mathbf{98.21}_{\pm 0.04}$	$\mathbf{89.27}_{\pm 0.07}$	$\mathbf{74.89}_{\pm 0.52}$	$\mathbf{97.71}_{\pm 0.10}$	$\mathbf{78.24}_{\pm 0.44}$	**87.66**

4.3 Results on Llama-2-7b

Setting. To evaluate the scalability of Flat-LoRA, we fine-tune Llama 2-7B [43] on two tasks: *math* and *code*. We use a learning rate of $5e-4$ and cosine learning rate scheduler with a warmup ratio of 0.03. We use LoRA with rank 8 and alpha 16 and the training epoch is set to 2. Following [9], the backbone of Lllma 2-7B uses BF16 precision and the parameters of LoRA modules use FP32 precision for better performance. For *math* task, we finetune the model on MetaMathQA [44] and evaluate it on GSM8K evaluation set [45]. For *code* task, we finetune the model on Code-Feedback

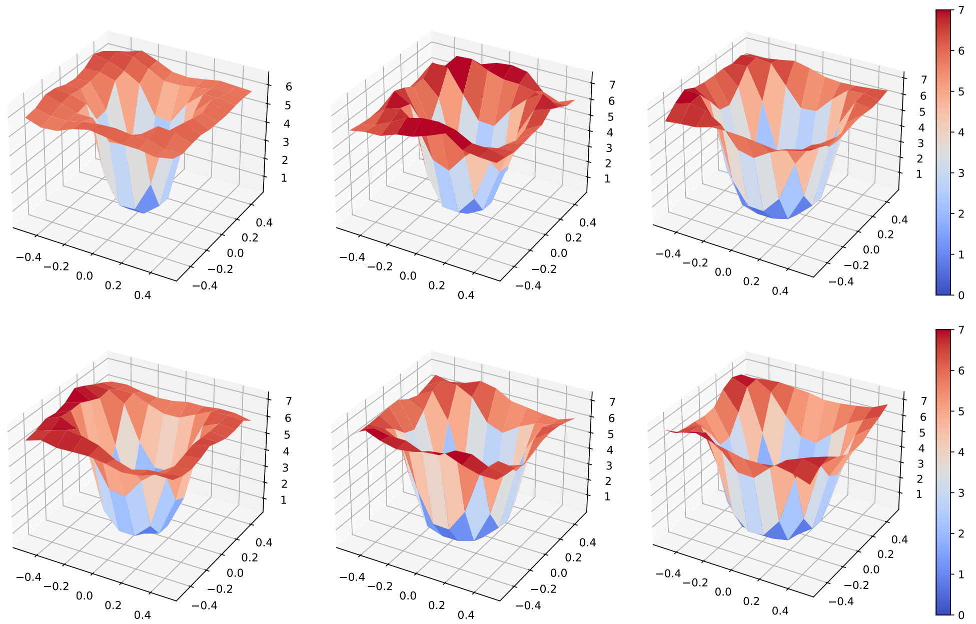


Figure 3: **Loss landscape visualization** with different LoRA ranks: 1 (**Left**) and 16 (**Middle**), and Full FT (Right), as well as different LoRA approaches: LoRA (**Up**) and Flat-LoRA (**Down**). Models are fine-tuned on CIFAR-100 with CLIP ViT-B/32.

[46] and evaluate it on HumanEval [47]. We only use 100k training subsets for both tasks. The random perturbation strength σ is set to 0.10 with an cosine increasing strategy.

Table 3: Results (%) on fine-tuning Llama 2-7B with GSM8K and Human-Eval datasets.

Method	GSM8K	Human-Eval
LoRA ($r = 8$)	$57.47_{\pm 0.35}$	$26.22_{\pm 0.52}$
Flat-LoRA ($r = 8$)	$\mathbf{60.65}_{\pm 0.23}$	$\mathbf{27.93}_{\pm 0.79}$

Results. We measure the performance of the *math* task by accuracy and the *code* task by PASS@1 metric. From the results in Table 3, we observe that Flat-LoRA significantly enhances LoRA's performance under large-scale fine-tuning scenarios, achieving an improvement of +3.18% on the GSM8K dataset and 1.37% on the Human-Eval dataset. It is important to note that here our LoRA performance is much stronger than the results reported in previous works, e.g., 57.47% (ours) v.s. 42.08% [9] on GSM8K. Still, Flat-LoRA continues to demonstrate significant accuracy improvements over the baseline approach, highlighting the effectiveness of pursuing the flatness of the full parameter space when fine-tuning large LLM models.

4.4 Ablation Study and Discussion

Comparison with Other Methods We then compare our approach with other recently proposed methods for improving LoRA, including initialization-based methods such as PiSSA and LoRA-GA, as well as optimization-based methods like DoRA and LoRA+. Our experiments are conducted on the CoLA and MRPC datasets using the T5-base model with lora rank 8. The results are presented in Table 4. We observe that Flat-LoRA consistently outperforms previous methods by 0.53%. Furthermore, our flat loss objective can be easily integrated with earlier approaches to yield consistent improvements by 0.31% to 0.93%. This highlights the effectiveness of considering the sharpness of the full parameter space.

Note that here we adopt a stronger training baseline, including employing a larger learning rate and longer training epochs, which achieves significantly better performance than the results reported in previous work [9]. In fact, CoLA and MRPC are two datasets that achieve that most significant improvement by LoRA-GA reported in the original paper [9]. Under our experimental settings, LoRA-GA does not demonstrate advantages over vanilla LoRA and may even perform worse. This may be because LoRA-GA provides a smart initialization strategy by maximizing gradient alignment with full parameter training, allowing for quicker convergence to a good local optimum (e.g., achieved in just one epoch as noted in [9]). However, this approach may not be optimal for reaching a better global optimum.

Table 4: Comparison with other methods on GLUE subsets with T5-Base.

Method	CoLA	MRPC
LoRA [5]	$82.20_{\pm 0.22}$	$88.03_{\pm 0.14}$
PiSSA [8]	$82.44_{\pm 0.14}$	$88.96_{\pm 0.44}$
LoRA-GA [9]	$81.83_{\pm 0.21}$	$87.58_{\pm 0.41}$
DoRA [7]	$83.16_{\pm 0.15}$	$89.46_{\pm 0.37}$
LoRA+ [48]	$81.65_{\pm 0.34}$	$89.30_{\pm 0.47}$
Flat-LoRA (ours)	$83.19_{\pm 0.38}$	$89.59_{\pm 0.37}$
Flat-PiSSA (ours)	$83.35_{\pm 0.48}$	$89.89_{\pm 0.71}$
Flat-LoRA-GA (ours)	$82.23_{\pm 0.34}$	$88.15_{\pm 0.54}$
Flat-DoRA (ours)	**83.56**$_{\pm 0.27}$	**89.99**$_{\pm 0.47}$
Flat-LoRA+ (ours)	$82.56_{\pm 0.23}$	$89.61_{\pm 0.44}$

Results under different lora ranks. Following the settings in Section 4.1 and 4.2, we evaluate the performance of Flat-LoRA under different LoRA ranks. The results are shown in Figure 2. We observe that Flat-LoRA consistently outperforms LoRA across different LoRA ranks by +1.10% on MRPC and +1.15% on CIFAR-100. Even at lora rank 1, which is typically underfitting, Flat-LoRA still delivers a significant performance boost over LoRA. This highlights the importance of considering the sharpness of the full parameter space. Additionally, as the LoRA rank increases, we observe that LoRA's performance can degrade due to overfitting, particularly on MRPC, which is a small dataset with 3.7k data points. Flat-LoRA effectively mitigates this overfitting issue by identifying flatter minima that generalize better. Thus, we conclude that Flat-LoRA enhances LoRA fine-tuning performance not only in underfitting scenarios, where the rank is low and limited information from the full parameter space is explored, but also in high LoRA rank situations, where the risk of overfitting is more pronounced.

Landscape visualization. In Figure 3, we plot the loss landscape of the merged weights of LoRA and Flat-LoRA with different loRA ranks. Following the plotting technique in [17], we uniformly sample 11×11 grid points in the range of $[-0.5, 0.5]$ from random "filter-normalized" direction. We observe that Flat-LoRA obtains consistently flatter loss landscape than LoRA in both LoRA fine-tuning and full fine-tuning senarios. An interesting observation is that when the LoRA rank is small, the loss landscape of the merged weight space is typical sharper, indicating the significance of considering the sharpness of the full parameter space when utilizing LoRA fine-tuning. Our Flat-LoRA can enable flatter loss landscape as higher LoRA ranks, e.g., Flat-LoRA with rank=16 attains similar flat landscape as full fine-tuning, and obtain similar performance.

5 Conclusion

In this paper, we introduce Flat-LoRA, an efficient low-rank adaptation approach that aims to optimize the sharpness of the loss landscape within the full parameter space that LoRA situates in. Deviating from standard sharpness-aware approach that incurs significant computation and memory burdens, we employ a Bayesian expectation loss objective minima and utilize designed random weight perturbations to pursuit flat minima, maintaining the training speed and memory efficiency characteristic of parameter-efficient fine-tuning. Flat-LoRA achieves state-of-the-art performance in LoRA fine-tuning and can be easily integrated with previous methods for consistent improvements. Extensive experiments on natural language processing and computer vision tasks demonstrate the effectiveness of our approach.

References

[1] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[2] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. In *European conference on computer vision (ECCV)*, 2020.

[3] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning (ICML)*, 2022.

[4] Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

[5] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

[6] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *International Conference on Learning Representations (ICLR)*, 2023.

[7] Shih-yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. In *International Conference on Machine Learning (ICML)*, 2024.

[8] Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: Principal singular values and singular vectors adaptation of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[9] Shaowen Wang, Linxi Yu, and Jian Li. Lora-ga: Low-rank adaptation with gradient approximation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[10] Sepp Hochreiter and Jürgen Schmidhuber. Simplifying neural nets by discovering flat minima. In *Advances in Neural Information Processing Systems (NeurIPS)*, 1994.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural computation*, 1997.

[12] Bingcong Li, Liang Zhang, and Niao He. Implicit regularization of sharpness-aware minimization for scale-invariant problems. In *ICML 2024 Workshop on Theoretical Foundations of Foundation Models*, 2024.

[13] Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. In *International Conference on Learning Representations (ICLR)*, 2017.

[14] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations (ICLR)*, 2017.

[15] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning (ICML)*, 2017.

[16] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

[17] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[18] Pan Zhou, Jiashi Feng, Chao Ma, Caiming Xiong, Steven Chu Hong Hoi, et al. Towards theoretically understanding why sgd generalizes better than adam in deep learning. *Advances in Neural Information Processing Systems*, 33:21285–21296, 2020.

[19] Bingcong Li and Georgios B Giannakis. Enhancing sharpness-aware optimization through variance suppression. *arXiv preprint arXiv:2309.15639*, 2023.

[20] Tao Li, Pan Zhou, Zhengbao He, Xinwen Cheng, and Xiaolin Huang. Friendly sharpness-aware minimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5631–5640, 2024.

[21] Tao Li, Weisen Jiang, Fanghui Liu, Xiaolin Huang, and James T Kwok. Scalable learned model soup on a single gpu: An efficient subspace training strategy. 2024.

[22] Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. Normalized flat minima: Exploring scale invariant definition of flat minima for neural networks using pac-bayesian analysis. In *International Conference on Machine Learning*, pages 9636–9647. PMLR, 2020.

[23] Yaowei Zheng, Richong Zhang, and Yongyi Mao. Regularizing neural networks via adversarial model perturbation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[24] Devansh Bisla, Jing Wang, and Anna Choromanska. Low-pass filtering sgd for recovering flat optima in the deep learning optimization landscape. In *International Conference on Artificial Intelligence and Statistics*, pages 8299–8339. PMLR, 2022.

[25] Wei Wen, Yandan Wang, Feng Yan, Cong Xu, Chunpeng Wu, Yiran Chen, and Hai Li. Smoothout: Smoothing out sharp minima to improve generalization in deep learning. *arXiv preprint arXiv:1805.07898*, 2018.

[26] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

[27] Tao Li, Qinghua Tao, Weihao Yan, Yingwen Wu, Zehao Lei, Kun Fang, Mingzhen He, and Xiaolin Huang. Revisiting random weight perturbation for efficiently improving generalization. *Transactions on Machine Learning Research*, 2024.

[28] Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. In *International Conference on Learning Representations (ICLR)*, 2018.

[29] Guy Gur-Ari, Daniel A Roberts, and Ethan Dyer. Gradient descent happens in a tiny subspace. *arXiv preprint arXiv:1812.04754*, 2018.

[30] Yingwen Wu, Tao Li, Xinwen Cheng, Jie Yang, and Xiaolin Huang. Low-dimensional gradient helps out-of-distribution detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[31] Tao Li, Lei Tan, Zhehao Huang, Qinghua Tao, Yipeng Liu, and Xiaolin Huang. Low dimensional trajectory hypothesis is true: Dnns can be trained in tiny subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3411–3420, 2022.

[32] Tao Li, Yingwen Wu, Sizhe Chen, Kun Fang, and Xiaolin Huang. Subspace adversarial training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13409–13418, 2022.

[33] Zhengbo Wang and Jian Liang. Lora-pro: Are low-rank adapters properly optimized? *arXiv preprint arXiv:2407.18242*, 2024.

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[35] Yongchang Hao, Yanshuai Cao, and Lili Mou. Flora: Low-rank adapters are secretly gradient compressors. In *International Conference on Machine Learning (ICML)*, 2024.

[36] 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.

[37] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research (JMLR)*, 21(140):1–67, 2020.

[38] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, 2021.

[39] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Technical Report*, 2009.

[40] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops (CVPRW)*, 2013.

[41] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*. Granada, 2011.

[42] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

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

[44] Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

[45] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[46] Tianyu Zheng, Ge Zhang, Tianhao Shen, Xueling Liu, Bill Yuchen Lin, Jie Fu, Wenhu Chen, and Xiang Yue. Opencodeinterpreter: Integrating code generation with execution and refinement. *arXiv preprint arXiv:2402.14658*, 2024.

[47] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[48] Soufiane Hayou, Nikhil Ghosh, and Bin Yu. Lora+: Efficient low rank adaptation of large models. In *International Conference on Machine Learning (ICML)*, 2024.