VeRA on Images, A Benchmarking for Parameter Fine Tuning on Stable Diffusion and Object Detection Tasks

Nick Sullivan Jack McCall Sanketh Gudapati Ravi Dantuluri

Abstract

This paper explores the application of Vector-Based Random Adaptation Matrix (VeRA) for fine-tuning convolutional layers in computer vision tasks, which extends from Low-Rank Adaptation (LoRA) tuning to enhance computational efficiency and reduce parameter counts (Hu et al., 2022). VeRA employs a pair of low-rank matrices across all model layers, adjusting the influence through compact scaling vectors (Kopiczko et al., 2024). This innovation can significantly minimize trainable parameters, which addresses certain scalability and efficiency challenges that are possible when utilizing advanced models. We implemented VeRA on the ResNet-50 model and trained on the Intel Image and MNIST datasets for object detection and classification tasks (He et al., 2015). Our results indicate that VeRA not only reduces the number of parameters but also maintains or enhances accuracy compared to fully trained and LoRA-tuned models. We believe this shows VeRA's potential and applicability to a wide range of computer vision tasks, and its ability to transform fine-tuning practices in computer vision.

1 Introduction

Recent advancements in deep learning have been characterized by a significant scale-up of model sizes and complexities. While these models offer impressive performance gains, their use is often limited by high computational demands and extensive resource requirements. Specifically, within the field of computer vision, many sophisticated models such as YOLO and Dall-E have remarkable capabilities that have set new benchmarks in image recognition and generation but also require a ton of resources to do so.

Traditional fine-tuning approaches, while effective in adapting these models to new tasks, do not address the inefficiencies that arise from their large parameter spaces. For example, recent Parameter Efficient Tuning Techniques(PEFT) like Low-Rank Adaptation (LoRA) have made some efforts in reducing trainable parameters but still encounter limitations, especially when scaled up or adapted for multiple tasks (Hu et al., 2022).

In response to LoRA, a new PEFT method named VeRA, which stands for Vector-Based Random Adaptation Matrix, has been introduced as an improvement upon LoRA. VeRA extends the principles of LoRA by employing a single pair of low-rank matrices across all layers of a model and learning compact scaling vectors rather than the matrices themselves. This approach significantly diminishes the number of trainable parameters, thereby reducing memory overhead and computational complexity. While VeRA has been used on Large Language Model(LLM) tuning due to the large amount of parameters required for training, we have not found any notable work on using VeRA for vision tasks or specifically implementing VeRA in convolutional layers.

Our goal is to investigate VeRA for its potential for application in computer vision. VeRA for fine-tuning LLM's has been shown to enhance model accuracy while substantially minimizing the parameter count. We want to investigate whether similar results will hold for vision tasks and convolutional layers. For instance, by applying VeRA to the ResNet-50 model, trained on diverse datasets such as Intel Image and MNIST, we hypothesize an increase in training efficiency/reduction of parameters without sacrificing accuracy.

Our approach, in short, is to fine-tune the ResNet-50 model using VeRA on the Intel Image dataset and MNIST dataset for object detection and classification tasks and report observations and insights into VeRA within the realm of computer vision.

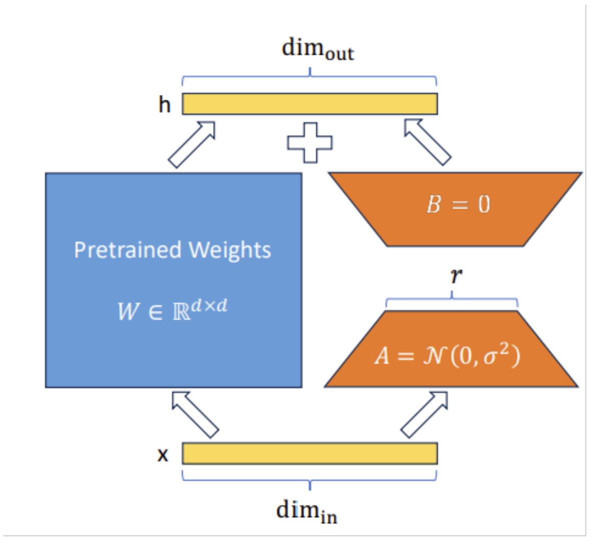

Figure 1: LoRA visualized (Hu et al., 2022). The method freezes the original weights and updates only the low-rank approximation of the original matrix. B is initialized to 0 and A to a normal distribution.

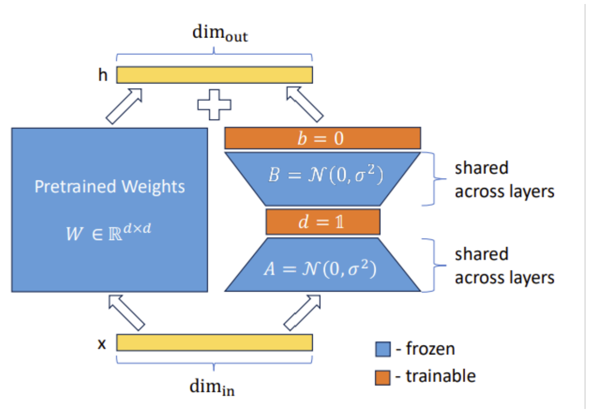

Figure 2: VeRA visualized (Kopiczko et al., 2024) The method freezes A and B and inserts two trainable matrices, d and b, in the middle. A, B, and d are initialized to be random while b is initialized to be 0.

2 Related Work

2.1 LoRA

(Hu et al., 2022) Low-rank decomposition is suggested by LoRA (fig:Implementing LoRA) as a replacement for current parameters while updating weights. LoRA utilizes low-rank matrices to replace full parameter updates, enabling the updating of model weights while significantly reducing the number of trainable parameters. Despite its reduced parameter space, LoRA-tuned models maintain performance levels comparable to their fully-tuned counterparts. Additionally, LoRA introduces zero additional inference latency, as it does not alter the underlying model architecture; instead, it merges trainable low-rank matrices with existing weights during the inference phase, thus preserving the original computational graph.

The relationship of LoRA to our work is foundational. Our proposed method, Vector-based Random Matrix Adaptation (VeRA), builds on the principles established by LoRA. Like LoRA, VeRA aims to reduce the trainable parameter count and computational overhead. However, VeRA extends these concepts by employing a single pair of low-rank matrices across all layers and learning compact scaling vectors rather than adjusting the matrices themselves. This adaptation results in an even more significant reduction in the number of trainable parameters, thus addressing some of the scalability and efficiency challenges still present with LoRA, particularly in the context of deploying AI models to resource-constrained environments. Our implementation of VeRA in the fine-tuning of the ResNet-50 model on diverse datasets serves as a direct extension and application of LoRA's low-rank techniques, specifically tailored to enhance performance in computer vision tasks.

2.2 VeRA

VeRA, as presented by (Kopiczko et al., 2024), is a technique that resembles LoRA. VeRA modifies the standard LoRA approach by introducing two trainable scaling vectors, which are applied to each low-rank matrix. These scaling vectors adjust the influence of the low-rank matrices across different layers, enhancing the model's ability to fine-tune with significantly fewer parameters, as seen in the figure below (fig:Implementing VeRA).

The efficacy of VeRA has been demonstrated through many comparisons with LoRA as well as other fine-tuning methodologies. Specifically, something that is notable is VeRA's performance on the E2E dataset using GPT-2 as the base model, where it outperforms LoRA while using a fraction of the parameters. Similar to LoRA, VeRA does not introduce additional latency during inference, as the scaling vectors and low-rank matrices are integrated into the original model weights, which maintains the style and efficiency of the original architecture.

We believe that VeRA extends beyond its initial applications in tasks such as NLP, which suggests potential benefits in other domains such as image classification and object detection in computer vision. Its ability to significantly reduce parameter counts while maintaining/enhancing model performance shows the value of low-rank approximation techniques. This makes VeRA particularly promising for using models in environments with fewer resources/storage, where maintaining computational efficiency and storage is important. Our work seeks to utilize VeRA's capabilities to fine-tune the ResNet-50 model for enhanced performance on various computer vision and image-related tasks, which can be a good segue into broader VeRA applications in the computer vision realm.

As we can see from the results reported on their paper on page 3, VeRA has improved LoRA for finetuning LLM models such as GPT-3 and BERT. We hypothesize that there will be this same improvement for vision tasks and specifically within VeRA implementation in convolutional layers.

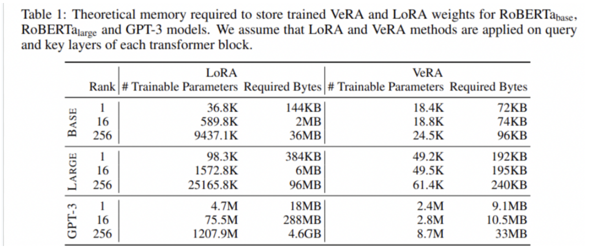

Table 1: Theoretical memory required to store trained VeRA and LoRA weights for RoBERTa$_{base}$, RoBERTa$_{large}$ and GPT-3 models. We assume that LoRA and VeRA methods are applied on query and key layers of each transformer block.

		LoRA		VeRA	
	Rank	# Trainable Parameters	Required Bytes	# Trainable Parameters	Required Bytes
BASE	1	36.8K	144KB	18.4K	72KB
	16	589.8K	2MB	18.8K	74KB
	256	9437.1K	36MB	24.5K	96KB
LARGE	1	98.3K	384KB	49.2K	192KB
	16	1572.8K	6MB	49.5K	195KB
	256	25165.8K	96MB	61.4K	240KB
GPT-3	1	4.7M	18MB	2.4M	9.1MB
	16	75.5M	288MB	2.8M	10.5MB
	256	1207.9M	4.6GB	8.7M	33MB

Figure 3: LoRA vs VeRA accuracy and parameter reduction

2.3 LoRA for Vision Tasks

LoRA, or Low-Rank Adaptation, has been effectively applied to vision tasks in various contexts. Here are a few notable examples:

1. **Vision Transformer Adaptation:** LoRA has been used to frequently in Vision Transformer (ViT) models. Specifically, this involves implementing low-rank matrices that fine-tune the model more efficiently without a significant increase in parameter count within the Vision Transformer architecture. This approach has been shown to be beneficial for tasks involving image recognition and could potentially be adapted for broader vision tasks (Dosovitskiy et al., 2020).

2. **Mixture of Cluster-conditional LoRA Experts (MoCLE):** This is another vision architecture that leverages LoRA, specifically for vision-language instruction tuning. It addresses some of the limitations of instruction tuned vision models and the task conflict in training diverse vision-language tasks by activating task-customized model parameters based on instruction clusters. This method has demonstrated effectiveness in zero-shot generalization across a variety of downstream vision-language tasks, largely in part due to LoRA's implementation (Gou et al., 2024)

In addition, during our research of LoRA's applications in vision tasks, we encountered two significant papers that worked with LoRA specifically for vision tasks:

1. **Paper 1: LoraRetriever: Input-Aware LoRA Retrieval and Composition for Mixed Tasks in the Wild.** This paper introduces a framework named "LoraRetriever," which utilizes LoRA for mixed tasks in vision and language applications. The framework features an input-aware retrieval process to dynamically select and apply suitable LoRA modules based on the specific task, which aims to increase efficiency and performance across a variety of scenarios (Zhao et al., 2024).

2. **Paper 2: Mixture of Cluster-conditional LoRA Experts for Vision-language Instruction Tuning.** This study discusses the "Mixture of Cluster-conditional LoRA Experts" (MoCLE), which is designed for vision-language instruction tuning. By clustering similar tasks and applying task-specific adaptations, this approach uses LoRA to tune models adaptively. It focuses on improving zero-shot generalization to unseen tasks by optimizing the application of LoRA modules within a specific vision-language model context (Gou et al., 2024).

2.4 VeRA for Vision Tasks

After we went through an exhaustive search focused on VeRA and its applications, we found that a lot of the existing literature mainly discusses VeRA in the context of fine-tuning large language models (LLMs) and tasks within the field of natural language processing (NLP). Surprisingly, there seems to be a lack of research into VeRA's applications in vision-related tasks. We hypothesize that one possible explanation for this observation is that parameter-efficient fine-tuning techniques, such as VeRA, are predominantly prioritized in NLP due to the exceptionally high parameter counts typical of language models, which significantly benefit from reductions in trainable parameters. In contrast, vision tasks and convolutional neural networks often work with fewer parameters, which can possibly leading some researchers to consider the gains from parameter reduction less impactful. However, we argue that the principles underlying VeRA could be equally beneficial for vision tasks, particularly for organizations/research groups that are limited by computational resources or storage capabilities. Adopting VeRA in vision tasks could potentially enhance model efficiency without sacrificing performance, making advanced computer vision more accessible to a broader range of users and applications.

3 Approach

Our approach was to utilize the new found VeRA for vision tasks and specifically apply VeRA to the convolutional layers.

As explained above, we see from Figure 1 how VeRA can be a lighter alternative to LoRA, as it uses b and d scalable, trainable vectors instead of full low-rank matrices A and B in LoRA. This reduces our number of trainable parameters from $2 * d * r$ (LoRA) to $r + d$ (VeRA) (r is a hyperparameter, and d is the number of channels of our input). Something worth considering is how kernel size interacts with LoRA and VeRA. Since we are approximating the channels with our layers, the kernel size will affect the number of parameters we use. A more accurate size of each matrix for convolutional layers is the following where d_{in} is the number of in channels,d_{out} is the number of out channels, k is kernel size, g is convolution groups. R is the rank of LoRA/VeRA.

LoRA A size: $d_{in} * k, r * k$
LoRA B size: $r * k, \frac{d_{out}}{g} * k$
Total LoRA: $r * k^2 * (d_{in} + \frac{d_{out}}{g})$

VeRA d size: $r * k$
LoRA b size: $\frac{d_{out}}{g} * k$
Total LoRA: $r * k^2 * \frac{d_{out}}{g}$

Thus, VeRA has $r * k^2 * d_{in}$ fewer parameters per convolutional layer than LoRA leading. For linear layers, the difference would be $r * d_{out}$. Since r is relatively small, the number of parameters will be dominated by the kernel size, leading to a smaller reduction of parameters than in linear layers.

We hypothesized that using VeRA for vision tasks, specifically within convolutional layers, can reduce the number of training parameters and also maintain/increase the accuracy of tasks, specifically within vision tasks such as object detection and classification. We did not use VeRA for the linear layers or any other layers in the network except the convolutional layers. VeRA has already been very effective on non-convolutional layers, especially for Natural Language Processing Tasks.

4 Results

We trained a Resnet-50 model (He et al., 2015) on two separate datasets. The MNIST dataset is a standard classification dataset that has often been used to test the capabilities of visual models. It comprises 60k 32x32 images depicting handwritten numbers. This is a relatively simple dataset. We decided to also train and test on a more complex dataset. Intel Image Dataset is a set of 25k 150x150 images of the natural world. Each image falls under 6 categories. We chose this dataset because of its complexity and difficulty.

We trained the model from scratch each time. While LoRA (Hu et al., 2022) and VeRA (Kopiczko et al., 2024) were originally used for fine-tuning large models we believe that it is appropriate to do so in this care. These tasks are relatively simple and ResNet-50 is a highly complex model with many weights. This suggests each layer likely has a low rank. We show this is true in our results.

We train the Resnet-50 in 3 different ways. We use the fully trained Resnet-50 and the LoRA trained Resnet-50 as our baseline. We train for a maximum of 10 epochs (we ended early when accuracy stopped improving) with a batch size of 32 and rank of 4 for both LoRA and VeRA.

We apply LoRA and VeRA on the convolutional layers only leaving the other layers to be fully trained. Using LoRA and VeRA for convolution differs from linear layers slightly because of how convolution works. All matrices in both techniques are multiplied by the kernel size of the layer they are approximating.

Training Method	# Parameters	Test Acc
Fully Trained	15,434,314	**96.01**
VeRA	**2,868,390**	95.55
LoRA	3,106,506	95.57

Table 1: Comparison of Test Accuracy for MNIST Classification

Training Method	# Parameters	Test Acc
Fully Trained	15,434,314	**85.12**
VeRA	**2,868,390**	81.21
LoRA	3,106,506	81.28

Table 2: Comparison of Test Accuracy for Intel Image Dataset r = 4

5 Discussion and Conclusions

VeRA is effective at being a lightweight alternative for convolutional layers. We found that our implementation of VeRA decreased the number of trainable parameters by 19% compared to the fully trained model and 9% compared to LoRA. This is with minimal degradation in performance. Both techniques drastically decrease the number of parameters for language tasks 3. As mentioned before each low-rank matrix's size depends on the kernel size. This dominates the size of the matrices. Language models do not use convolutional layers meaning they do not suffer from the same problem.

Because VeRA performs well on vision tasks without degradation of performance it can be used as a lightweight adapter for diffusion models. LoRA is already used as a lightweight adapter in order to do style transfer. This means one can easily share a fine-tuned diffusion model by simply sharing the LoRA weights which are smaller than the backbone model. Users apply different LoRAs like adapters onto a backbone model. This allows for a ton of different styles without having to store a ton of different models. VeRA can act in the same way but can be lighter weight.

More interestingly one can insert the VeRA matrices into the pre-trained LoRA matrices. This can allow for different finer-tuned style transfers. For example, one might fine-tune a stable diffusion model using LoRA on oil paintings. Then one could freeze the LoRA matricies and insert the VeRA scaling layers. Then fine-tune the model on impressionism paintings and save the VeRA layers. Then one can insert a fresh pair of VeRA layers and fine-tune on realism paintings. This may lead to better performance for these subsections since we don't need to learn how oil paintings look. This can be a useful tool for specific style transfers which are related to each other. This is the obvious next step for this project and has the best real-world use case. We experimented a little bit with the DiffusionDB dataset as well as implementing VeRA for diffusion related tasks, which can be a great next step for future work for VeRA for vision (Wang et al., 2023).

.

6 Statement of Individual Contribution

Group coordination was a team effort. Everyone regularly communicated with another regarding deadlines and times that the group should gather to work on the project. We met close to weekly to provide updates and any progress on the work that we had done, either together or independently. Nick took charge on the object detection VeRA implementation, while everyone else worked to apply VeRA for diffusion tasks. However, realizing that it would take more time than anticipated for diffusion due to computational costs/complexity, the object diffusion aspect was scrapped, and we all collaboratively worked on object detection/classification with VeRA, as well as the final presentation slides. In addition, we all collaboratively worked together on the final report, adding in the sections we worked on.

7 External Resources Used

For the ResNet-50 model, we utilized the following codebase: `https://github.com/nickdsullivan/vera_convolution`. We used this as a framework to implement VeRA into ResNet-50.

LoRA GitHub: `https://github.com/microsoft/LoRA` We used the original LoRA code and modified it to use VeRA

ResNet-50 implementation GitHub: `https://github.com/Ugenteraan/ResNet-50-CBAM-PyTorch/blob/main/models/resnet50.py` We implemented the

LoRA layers for each convolutional layer.

References

Alejandro Baldominos, Yago Saez, and Pedro Isasi. 2019. A survey of handwritten character recognition with mnist and emnist. *Applied Sciences* 9(15):3169. https://doi.org/10.3390/app9153169.

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. 2020. An image is worth 16x16 words: Transformers for image recognition at scale.

Yunhao Gou, Zhili Liu, Kai Chen, Lanqing Hong, Hang Xu, Aoxue Li, Dit-Yan Yeung, James T. Kwok, and Yu Zhang. 2024. Mixture of cluster-conditional lora experts for vision-language instruction tuning.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. https://openreview.net/pdf?id=nZeVKeeFYf9.

Dawid J. Kopiczko, Tijmen Blankevoort, and Yuki M. Asano. 2024. Vera: Vector-based random matrix adaptation.

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.

Zijie J. Wang, Evan Montoya, and David Munechika. 2023. Diffusiondb: A large-scale prompt gallery dataset for text-to-image generative models.

Shin-Ying Yeh, Yu-Guan Hsieh, Zhidong Gao, Bernard B W Yang, Giyeong Oh, and Yanmin Gong. 2023. Navigating text-to-image customization:from lycoris fine-tuning to model evaluation.

Ziyu Zhao, Leilei Gan, Guoyin Wang, Wangchunshu Zhou, Hongxia Yang, Kun Kuang, and Fei Wu. 2024. Loraretriever: Input-aware lora retrieval and composition for mixed tasks in the wild.