A. Appendix

A.1. Theoretical Analysis

Using the definitions 3.1, 3.2, 3.5, 3.4 and 3.3 we state the following theorem;

Theorem A.1. *For a task t_{d+1}, we assume a hypothesis $h \in \mathcal{H}_{W_{d+1}}$ expressed as $h(W_{d+1}, X) = W_{d+1}X_{d+1} + W_0X_{d+1} + b$ where W_{d+1} has rank m, b is some constant and W_0 represents weights of a pretrained foundation model that is frozen during finetuning respectively. We have $h^{\mathcal{E}} \in \mathcal{H}_{W_{\mathcal{E}}}, h^* \in \mathcal{H}_{W^*{}_{d+1}}$ such that $h^{\mathcal{E}}(W_{\mathcal{E}}, X_{d+1}) = \alpha_{d+1}\mathcal{V}_K^T X_{d+1} + W_0X_{d+1} + b$ where $W_{\mathcal{E}}$ has rank K and $h^*(W^*{}_{d+1}, X_{d+1}) = C\hat{W}X_{d+1} + W_0X_{d+1} + b$ where $h^*(W^*{}_{d+1}, X_{d+1}) = Y_{d+1}$ is the true solution for task t_{d+1}. For a Lipschitz continuous loss ($\ell^F_{\mathcal{S}_t}(h)$) that is strong convex within the shared principal subspace spanned by principal components $\mathcal{V}_K^T$ with some Lipschitz constant (L), the risk can be written as $R^F_{\mathcal{S}_{d+1}}(h_W) = E_{\mathcal{S}_t}[\ell^F_{\mathcal{S}_t}(h)]$, and using Rademacher complexity bounds we can say with probability at least $1 - 4\delta$ for some $\delta > 0$,*

$$\|W^*{}_{d+1} - W_{d+1}\|_F^2 \leq C_1 \cdot \left(\frac{\sqrt{m}}{\sqrt{s_t}}\right) + C_2 \tag{4}$$

$$\|\alpha^*_{d+1}\mathcal{V}_K^T - W_{\mathcal{E}}\|_F^2 \leq C_1 \cdot \left(\frac{\sqrt{K}}{\sqrt{s_t}}\right) + \|C\|_2^2 \sum_{i=K+1}^{nd} \sigma_i^2 + C_2 \tag{5}$$

where σ_i are singular values of $\hat{W}$, C is some constant such that $W^*{}_{d+1} = C\hat{W}$ and C_1, C_2 are some constants.

Proof. The derivation is straightforward, we can write the difference in risks for $h^{\mathcal{E}}$ and h^* as

$$R^F_{\mathcal{S}_{d+1}}(h^{\mathcal{E}}) - R^F_{\mathcal{S}_{d+1}}(h^*) = \mathbb{E}_{\mathcal{S}_t}\left[\ell^F_{\mathcal{S}_t}(h^{\mathcal{E}}) - \ell^F_{\mathcal{S}_t}(h^*)\right]$$

By definition of strong convex loss function for some constant $\mu \geq 0$,

$$\mathbb{E}_{\mathcal{S}_t}\left[\ell^F_{\mathcal{S}_t}(h^{\mathcal{E}}) - \ell^F_{\mathcal{S}_t}(h^*)\right] \geq \frac{\mu}{2}\|W_{\mathcal{E}} - W^*{}_{d+1}\|_F^2$$

We also know from generalization error bounds using Rademacher Complexity from [2] that with probability at least $1 - 2\delta$,

$$|R^F_{\mathcal{S}_{d+1}}(h^{\mathcal{E}}) - \hat{R}^F_{\mathcal{S}_{d+1}}(h^{\mathcal{E}})| \leq \frac{\mathcal{R}_{s_{d+1}}(\mathcal{H}_{W_{\mathcal{E}}})}{2} + \sqrt{\frac{\ln(1/\delta)}{2s_t}}$$

We can rewrite risk as

$$\begin{aligned} R^F_{\mathcal{S}_{d+1}}(h^*) - R^F_{\mathcal{S}_{d+1}}(h^{\mathcal{E}}) =& R^F_{\mathcal{S}_{d+1}}(h^*) - \hat{R}^F_{\mathcal{S}_{d+1}}(h^*) \\ &- R^F_{\mathcal{S}_{d+1}}(h^{\mathcal{E}}) + \hat{R}^F_{\mathcal{S}_{d+1}}(h^{\mathcal{E}}) \\ &+ \hat{R}^F_{\mathcal{S}_{d+1}}(h^*) - \hat{R}^F_{\mathcal{S}_{d+1}}(h^{\mathcal{E}}) \end{aligned}$$

Since we know by definition of h^* that $\hat{R}^F_{\mathcal{S}_{d+1}}(h^*) \leq \hat{R}^F_{\mathcal{S}_{d+1}}(h^{\mathcal{E}})$, we can say

$$\begin{aligned} R^F_{\mathcal{S}_{d+1}}(h^*) - R^F_{\mathcal{S}_{d+1}}(h^{\mathcal{E}}) \leq& R^F_{\mathcal{S}_{d+1}}(h^*) - \hat{R}^F_{\mathcal{S}_{d+1}}(h^*) \\ &- R^F_{\mathcal{S}_{d+1}}(h^{\mathcal{E}}) + \hat{R}^F_{\mathcal{S}_{d+1}}(h^{\mathcal{E}}) \end{aligned}$$

Then we take a union bound to conclude that with probability at least $1 - 4\delta$,

$$\begin{aligned} R^F_{\mathcal{S}_{d+1}}(h^*) - R^F_{\mathcal{S}_{d+1}}(h^{\mathcal{E}}) \leq& \frac{\mathcal{R}_{s_{d+1}}(\mathcal{H}_{W_{\mathcal{E}}})}{2} + \sqrt{\frac{2\ln(1/\delta)}{s_{d+1}}} \\ &+ \frac{\mathcal{R}_{s_{d+1}}(\mathcal{H}_{W^*{}_{d+1}})}{2} \end{aligned}$$

Hence, we can also say that with probability at least $1 - 4\delta$,

$$\frac{\mu}{2}\|{W^*}_{d+1} - W_{\mathcal{E}}\|_F^2 \leq \frac{\mathcal{R}_{s_{d+1}}(\mathcal{H}_{W_{\mathcal{E}}})}{2} + \sqrt{\frac{2\ln(1/\delta)}{s_t}} + \frac{\mathcal{R}_{s_{d+1}}(\mathcal{H}_{{W^*}_{d+1}})}{2} \tag{6}$$

The Rademacher complexity of a low-rank weight matrix class $\mathcal{H}_{W_{\mathcal{E}}}$ with rank K can be directly bounded using results from [2] as

$$\mathcal{R}_{s_{d+1}}(\mathcal{H}_{W_{\mathcal{E}}}) = \mathcal{O}(\frac{\sqrt{K}\|W_{\mathcal{E}}\|_F}{\sqrt{s_t}})$$

We can separate the constants including $\mathcal{R}_{s_{d+1}}(\mathcal{H}_{{W^*}_{d+1}})$ from 6 and assume that, for a normalised $\|W_{\mathcal{E}}\|$, it is usually bounded, then we can write:

$$\|{W^*}_{d+1} - W_{\mathcal{E}}\|_F^2 \leq C_1 \cdot \left(\frac{\sqrt{K}}{\sqrt{s_t}}\right) + C_2 \tag{7}$$

Similarly, we can also say for W_{d+1} that

$$\|{W^*}_{d+1} - W_{d+1}\|_F^2 \leq C_1 \cdot \left(\frac{\sqrt{m}}{\sqrt{s_t}}\right) + C_2 \tag{8}$$

This proves 4. Now to further prove 5, we use properties of Frobenius norm,

$$\begin{aligned}\|W_{\mathcal{E}} - \alpha^*_{d+1}\mathcal{V}_K^T\|_F^2 - \|{W^*}_{d+1} - \alpha^*_{d+1}\mathcal{V}_K^T\|_F^2 \\ \leq \|W_{\mathcal{E}} - {W^*}_{d+1}\|_F^2\end{aligned}$$

Then following from the definition of ${W^*}_{d+1}$, we can say that,

$$\|W_{\mathcal{E}} - \alpha^*_{d+1}\mathcal{V}_K^T\|_F^2 - \|C\|_2^2 \sum_{i=K+1}^{nd} \sigma_i^2 \leq \|W_{\mathcal{E}} - {W^*}_{d+1}\|_F^2$$

Finally, using the Rademacher complexity bound we provided earlier, we can say that with probability at least $1 - 4\delta$

$$\begin{aligned}\|\alpha^*_{d+1}\mathcal{V}_K^T - W_{\mathcal{E}}\|_F^2 &\leq \|{W^*}_{d+1} - W_{\mathcal{E}}\|_F^2 \\ &\leq C_1 \cdot \left(\frac{\sqrt{K}}{\sqrt{s_t}}\right) + \|C\|_2^2 \sum_{i=K+1}^{nd} \sigma_i^2 + C_2\end{aligned}$$

We can just rewrite $W_{\mathcal{E}} = \alpha^*_{d+1}\mathcal{V}_K^T$ and get the same bound as above for $\|\alpha^*_{d+1} - \alpha_{d+1}\|_F^2$. We can similarly obtain the upper bound for 5

This concludes the proof. □

Theorem A.1 provides an upper bound on the Frobenius norm of the difference between W_{d+1} or $W_{\mathcal{E}}$ and the optimal solution ${W^*}_{d+1}$. 5 provides a tighter upper bound on the norm of the difference when task t_{d+1} majorly lies in the shared principal subspace. The extent to which task t_{d+1} lies in the shared principle subspace is captured by the second term involving the sum of squared truncated singular values $\hat{W}$. Hence, if the task completely or majorly lie in the shared principal subspace, then the first term (sqrt(rank)) will dominate the upper bound. Hence, if rank($W_{d+1} \geq K$), then we can see that the upper bound in eq. 5 will be tighter than in eq. 4 where the task lies majorly in the shared principal subspace. Similarly, when $m \leq K$, the upper bound on the difference norm will be tighter for W_{d+1} than $W_{\mathcal{E}}$. When ${W^*}_{d+1}$ has a significant alignment or projection along the singular vectors orthogonal to the ones with top K singular values, then the second term in 4 comes into picture and it becomes difficult to directly compare the bounds in 4 and 5. However, if majority of the variance of ${W^*}_{d+1}$ is along the singular vectors orthogonal to the top K components, it follows that $W_{\mathcal{E}}$ will never be able to achieve convergence

while W_{d+1}. In contrast, W_{d+1} could perform significantly better, as it is not restricted to learning only along the top K principal components of $\hat{W}$. While the assumption that ${W^*}_{d+1}$ is spanned by the principal components of the shared principal subspace might appear to be very strong, we empirically observe in table 4 that such an assumption is not impractically far from reality. Particularly, we observe in table 2 that for GLUE benchmark, LoRA adapters trained on 5 diverse tasks shared a principal subspace. We see that EigenLoRAx was able to leverage the principal components of this shared subspace with just 12K training parameters learned for a new 6th task and achieve competitive performance compared to fine-tuning full rank weights with 125M parameters or individual LoRA adaptors with 1.2M parameters, even outperforming them in certain tasks. Similarly, table 4 demonstrates zeroshot performance using only top K principal components of the shared subspace obtained through 500 LoRA adaptors trained on diverse tasks. This further suggests that increasing the number of LoRA adapters enables a richer set of top principal components, effectively spanning the shared subspace and providing broader coverage for new tasks.

A.2. Experiments

For VeRA, LoRA and PiSSA, we experimented with a range of learning rates, from higher to lower, along with three different scheduling approaches: ReduceLROnPlateau, Linear, and Cosine. The hyperparameters that yielded the best average performance were selected for further experimentation. The observed discrepancies with EigenLoRAx hyperparameters are attributable to these methodological choices. Comprehensive hyperparameter tuning for EigenLoRAx was not pursued extensively, as the initially selected hyperparameters, notably a high learning rate paired with ReduceLROnPlateau or Linear, demonstrated satisfactory performance, thereby conserving computational resources.

A.2.1. Image Classification

Trainable parameters for EigenLoRAx The base model is vit-base-patch16-224. The following are the trainable parameters in ViT [12] that are trained for EigenLoRAx. We ignore the last linear layer for simplicity since it is trained for all models and baselines and is constant. The loading parameter has the shape of $[\text{number of EigenLoRAx PC}, 1]$ (we only have 2 in each EigenLoRAx PC for this experiment). Therefore, the total number of trainable parameters (for the number of components$= 2$) is 12 (layers) $\times$ 4 (set of parameters per layers) $\times$ 2 (number of trainable parameter per coefficient) $= 96$ trainable parameters.

Hyperparameters LoRA [18] and VeRA [24] implementations are taken from the HuggingFace PEFT [35] library with hyperparameters of the default method. For Food101 [4] experiment, we randomly remove 1 class for ease of compute. Experimental hyperparameters are reported in Table 5 and Table 6.

Table 5. Hyperparameters for LoRA [18] and VeRA [24] for the Image Classification Experiment

	CIFAR100	**Flowers102**	**Food101**
Learning Rate	1e−4	1e−4	1e−4
Weight Decay	0.1	0.1	0.1
Warmup ratio	0.06	0.06	0.06
Epochs	10	10	10
Number of Subsets	5	6	5
Categories/Subset	20	17	20
Seed	42	42	42
Batch Size	128	64	128

Experimental Results The experiments were conducted 5 times utilizing randomly generated dataset splits. The mean accuracy values are reported in Table 1. Empirical analysis indicates that without control and annealing of learning rates, the loss for both LoRA and VeRA may diverge or plateau, particularly with high learning rates. Even with the lower learning rate, Full training or LoRA can overfit to the training data without proper regularization. In contrast, no such instability was observed during EigenLoRAx training, where a relatively higher learning rate proved advantageous for rapid convergence.

Table 6. Hyperparameters for EigenLoRAx for the Image Classification Experiment

	CIFAR100	Flowers102	Food101
Learning Rate	1e−2	1e−2	1e−2
Weight Decay	0.1	0.1	0.1
Warmup ratio	0.06	0.06	0.06
Epochs	10	10	10
Number of Subsets	5	6	5
Categories/Subset	20	17	20
Seed	42	42	42
Batch Size	128	64	128

Table 7. Image Classification Accuracy results on CIFAR100 [25]

Model	Trainable Params	subset1	subset2	subset3	subset4	subset5	Avg.
FT	86389248	98.8	97.95	95.55	96.05	96.3	96.93
LoRA ($r = 1$)	36864	97.6	93.95	93.75	91.75	85.2	92.45
LoRA ($r = 4$)	147456	98.15	95.2	93.5	92.85	89.25	93.79
VeRA ($r = 2$)	18480	93.65	89.7	89.5	89.95	91.55	90.87
EigenLoRAx ($K = 2$)	96	97.25	95.05	94.55	93	94.15	94.8

Table 8. Image Classification Accuracy results on Food101 [4]

Model	Trainable Params	subset1	subset2	subset3	subset4	subset5	Avg.
FT	86389248	98.64	97	97.36	94.28	95.92	96.64
LoRA ($r = 1$)	36864	93.36	88.44	94.28	89.4	89.9	91.076
LoRA ($r = 4$)	147456	98.2	96.96	96.08	92.88	94.52	95.728
VeRA ($r = 2$)	18480	91.22	88.42	94.42	91.88	92.82	91.752
EigenLoRAx ($K = 2$)	96	97.24	95.96	96	91.88	94.6	95.136

Table 9. Image Classification Accuracy results on Flowers102 [37]

Model	subset1	subset2	subset3	subset4	subset5	subset6	Avg.
FT	99.7	99.3	98.01	98.22	99.7	98.01	98.82
LoRA ($r = 1$)	85.9	88.47	92.69	91.02	91.7	91.01	90.13
LoRA ($r = 4$)	96.23	92.76	97.22	95.01	98.24	90.73	95.03
VeRA ($r = 2$)	99.2	95.4	97.7	94.7	90.9	95	95.48
EigenLoRAx ($K = 2$)	99.686	97.905	97.689	98.291	99.344	97.718	98.43

A.3. Natural Language Processing - GLUE benchmark

Hyperparameters LoRA [18], VeRA [24] and PISSA [36] implementations are taken from the HuggingFace PEFT [35] library. Refer to Table 10 and Table 11 for hyperparameter details. For LoRA [18], we use the ranks $\in \{8, 16\}$. For VeRA [24], we use rank$= 256$, and for EigenLoRAx, we use $K \in \{16, 32\}$ and $r = 8$. Here, r refers to the dimensionality of the trainable coefficients and not the rank. For both PISSA [36] and LoRA, all the parameters of the low rank matrix are trainable. For the EigenLoRAx initialization experiment, we train both the components and coefficients for a fair comparison with PISSA. In

practice, however, we do not need to do so - we can tune only the sparse coefficients and after the loss converges, finetune the components for a few training steps.

Table 10. Hyperparameters for LoRA [18], VeRA [24] and PiSSA [36] for the GLUE benchmark. [46]

	CoLA	MRPC	QNLI	RTE	SST-2	STSB
Learning Rate	4e−4	4e−4	4e−4	5e−4	5e−4	4e−4
Weight Decay	0.1	0.1	0.1	0.1	0.1	0.1
Warmup ratio	0.06	0.06	0.06	0.06	0.06	0.06
Epochs	80	30	25	80	60	40
Scheduler	Linear	Linear	Linear	Linear	Linear	Linear
Seed	0	0	0	0	0	0
Batch Size	64	64	64	64	64	64

Table 11. Hyperparameters for EigenLoRAx for the GLUE benchmark. [46]. (RLrP - ReduceLROnPlateau)

	CoLA	MRPC	QNLI	RTE	SST-2	STSB
Learning Rate	4e−3	4e−3	4e−3	5e−3	5e−3	4e−3
Weight Decay	0.1	0.1	0.1	0.1	0.1	0.1
Warmup ratio	0.06	0.06	0.06	0.06	0.06	0.06
Epochs	80	30	25	80	60	40
Scheduler	RLrP	RLrP	RLrP	RLrP	RLrP	RLrP
Seed	0	0	0	0	0	0
Batch Size	64	64	64	64	64	64

A.4. Text-to-Image Generation (Stable Diffusion Models)

Figure 5 show more examples of a text-to-image stable diffusion model finetuned using EigenLoRAx. Note that not only there is no publicly available code for VeRA that allows its usage in complex text-to-image generation tasks, but our VeRA implementation also did not work well in this task.

Figure 5. A single EigenLoRAx (identical components, varying loadings) was employed to produce these images utilizing the Stable Diffusion-XL [38] model. A comparison between our results and those obtained from multiple LoRAs does not show a noticeable degradation in visual quality.

Figure 6. Failure Case: EigenLoRAx may fail if an important component is missing from the initialized subspace i.e. the shared subspace is incomplete, which may happen due to inadequacy in the number of initial adapters or due to the majority of the adapters being of bad quality. E.g., the model may have lost the essential "mosaic" property when generating an image for the prompt: "mosaic picture of a dog."

B. Method Analysis and Ablation

Through a rigorous comparative analysis of EigenLoRAxs and their target LoRAs, we identified that the most pronounced reconstruction discrepancies manifest in the initial and terminal layers of the neural network, as depicted in Figure 7. Allowing the EigenLoRAx PCs in these layers to undergo fine-tuning alongwith the coefficients can alleviate failure scenarios, thereby alleviating the need for comprehensive model fine-tuning.

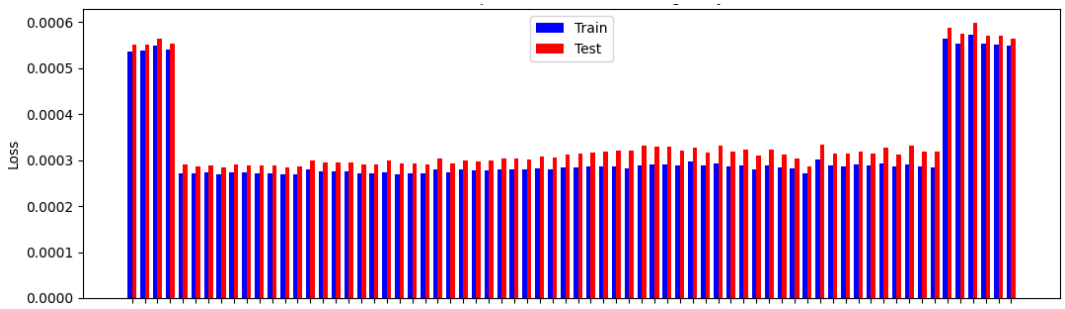

Figure 7. Average reconstruction error between EigenLoRAx and a set of LoRAs for all UNet layers in a stable diffusion model.

B.1. How to Choose K Principal Components and r for EigenLoRAx

We perform an ablation study on the selection of EigenLoRAx principal components (K). Our analysis concentrates on one experiment as shown in Figure 10, specifically pertaining to the MRPC task within the GLUE [46] benchmark. The analysis in Figure 8 shows the training loss in relation to increasing number of EigenLoRAx principal components K, as well as the explained variance of the LoRAs used to initialize the EigenLoRAx in Figure 9. We find, empirically, that choosing EigenLoRAx PCs for the explained variance of $50 - 80\%$ of the LoRAs used to initialize EigenLoRAx is sufficient for a robust initialization. This is shown in Figure 9 where we choose $K = 8$ which roughly corresponds to the explained variance of $55 - 60\%$. We further ablate this choice in Figure 8, where although substantial improvements are evident up to $K = 8$, an increase in the number of K thereafter yields only marginal gains, demonstrating diminishing returns as the number of components increases. The parameter r in EigenLoRAx does not equate the *rank* parameter in LoRA and its variants. It reflects the dimensionality of the EigenLoRAx coefficients. Although $r = 1$ works well, we observe slight performance improvements as we increase this value as shown in Figure 11. Increasing this value corresponds to a small amount of parameter increase. We observe no finetuning instability by changing this value and recommend that it can be set to anywhere between 1 and the rank of the LoRAs used to initialize EigenLoRAx.

Figure 8. Training Loss Convergence for different numbers of Eigen-LoRAx PCs

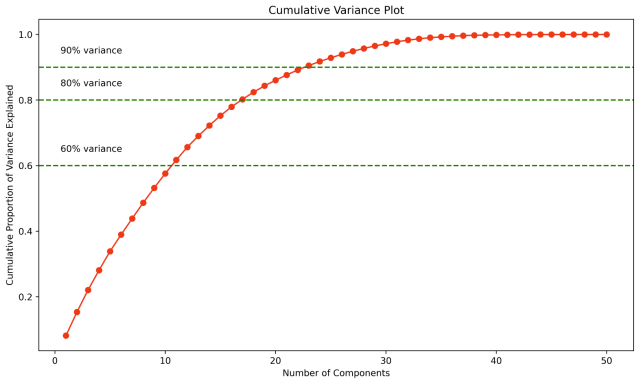

Figure 9. Explained Variance for increasing number of PCs

Figure 10. Ablation of Number of EigenLoRAx Principal Components

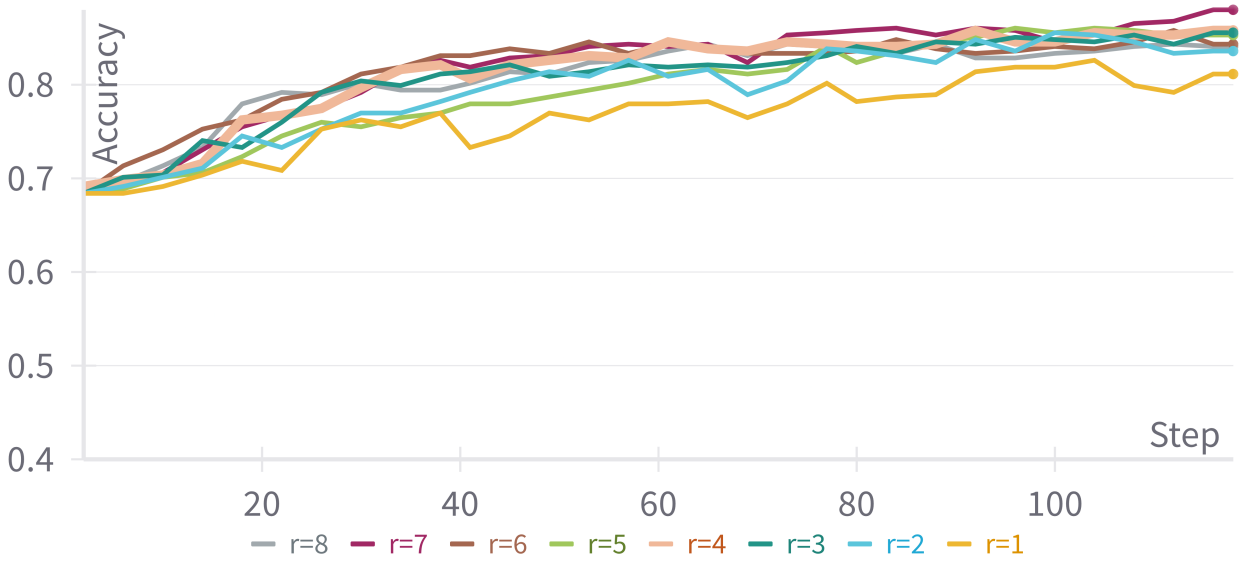

Figure 11. Ablation for the EigenLoRAx's r hyperparameter. This experiment was done for the MRPC task in the GLUE benchmark.

B.2. Failure Cases

Figure 6 illustrates a potential failure case of EigenLoRAx, where the incorrect number of principal components (PCs) was selected. In this instance, the "mosaic style" information was excluded from the principal subspace identified by EigenLoRAx due to an insufficient number of PCs. However, this issue can be resolved by selecting a larger number of PCs, as the extended principal subspace contains the necessary information for the task.

Another hypothetical failure scenario arises if the domain gap between the low-rank adapters used to initialize EigenLoRAx and the downstream task is significantly large. Although we do not observe such a case in our experiments, it is plausible that under such conditions, EigenLoRAx might underperform. This issue could potentially be mitigated by allowing only a subset of PCs to remain trainable, enabling the model to adapt more effectively to the target domain.

A further observed limitation of EigenLoRAx occurs in complex tasks like Text-to-Image generation, which may extend to other tasks as well. If the majority of LoRAs used to initialize EigenLoRAx encode biases (e.g., related to gender, race, or context), these biases tend to propagate into EigenLoRAx outputs. While such biases are a common issue in deep learning models trained using stochastic gradient descent or similar methods, addressing them remains a critical area of future work.

We consider this an important avenue for improvement and discuss the broader implications in Appendix C.

B.3. Impact of LoRA adapter quality on EigenLoRAx PC initialization

To evaluate EigenLoRAx's robustness to adapter quality and its resistance to noise, we conducted an ablation study on a subset of tasks of the NLU experiment specified in Section 4.2. Specifically, we generated EigenLoRAx adapters using LoRA matrices with varying levels of random noise added. The results are shown in Table 12

Table 12. EigenLoRAx performance on subset of GLUE task using noisy LoRA adapters for initialization

Noise Level	CoLA	MRPC	RTE	STS-B	Avg
5%	60.51	85.45	74.73	89.9	77.65
15%	57.53	83.09	72.92	89.9	75.86
30%	55.23	76.47	71.84	89.8	73.34

The results show that EigenLoRAx exhibits only minor performance changes even as noise levels increase significantly, indicating some robustness to adapter quality. This suggests that EigenLoRAx can still perform effectively without high quality adapters. However, there is a limit to this robustness. If the signal-to-noise ratio (SNR) in the initial LoRA matrices becomes extremely low—where the LoRAs primarily encode noise rather than meaningful information—the effectiveness of EigenLoRAx diminishes. In such cases, the principal components (PCs) extracted by EigenLoRAx would correspond to random directions in the parameter space. Consequently, EigenLoRAx's performance would resemble that of random matrix methods, such as VeRA and NoLA. These methods rely on a large number of random components or bases to approximate meaningful results. While they can achieve reasonable performance, they require fine-tuning a substantially larger number of weights associated with these large number of random components, leading to less efficient learning compared to EigenLoRAx. This highlights an important consideration: for EigenLoRAx to maintain its efficiency and effectiveness, the initial LoRA matrices must contain at least a minimal level of meaningful signal. This requirement ensures that EigenLoRAx can leverage the structured information encoded in the LoRAs while avoiding the inefficiencies of purely random approaches.

B.4. Forward pass and backward pass FLOPs

While it is obvious that EigenLoRAx utilized significantly less number of model parameters as the number of tasks in a domain increase, we show that even in terms of floating point operations on a single task, EigenLoRAx is more efficient than LoRA for our experiments. Even for a single task, the number of floating point operations or multiply-accumulate operations in a forward pass for EigenLoRAx is lower than LoRA for all our experiments. Here are the comparisons of the floating point operations (FLOPs) for the forward (fwd FLOPs) and including backward pass (fwd+bwd FLOPs) for each of the Image Classification and GLUE benchmark (batch size = 1) (MFLOPs - MegaFlops):

Table 13. Floating Point Operation calculations for GLUE Benchmark experiment

Method	Training Parameters	fwd FLOPs	fwd+bwd FLOPs
LoRA	1.2M	97,930 MFLOPS	293,800 MFLOPS
VeRA	25K	106,390 MFLOPS	319,170 MFLOPS
EigenLoRAx	12K	97,030 MFLOPS	291,080 MFLOPS

Table 14. Floating Point Operation calculations for Image Classification experiment

Method	Training Parameters	fwd FLOPs	fwd+bwd FLOPs
LoRA	36K	33,773.8 MFLOPS	101,322 MFLOPS
VeRA	18K	33,744.8 MFLOPS	101.234 MFLOPS
EigenLoRAx	96	33,730.2 MFLOPS	101,191 MFLOPS

C. Broader Impact and Implications

This work presents a novel parameter-efficient method for deep learning methods utilizing open source, pretrained Low-Rank Adaptation (LoRA) models. By substantially reducing the computational and memory demands of training and inference, our approach creates a more sustainable and environmentally friendly deep learning paradigm. Our method democratizes accessibility to larger models, making them accessible to researchers and practitioners with limited resources. Furthermore, by harnessing pretrained models, our method can accelerate development and diminish the need for extensive data collection. However, we recognize the inherent risks associated with the use of pretrained models. These include potential biases (racial, gender, etc.), explicit content, since there is no guarantee of the data or method used in training the model, and the potential presence of malicious code. Appropriate caution is advised when using unverified, open-source models.