

---

# EMLoC: Emulator-based Memory-efficient Fine-tuning with LoRA Correction

---

## Supplementary Material

1 This supplementary document provides additional context, analysis, and results to support the main  
2 paper. We begin with a discussion of the limitations of our approach and potential directions for future  
3 work, followed by a brief examination of the broader social impact. We then present supplementary  
4 proofs and derivations to justify components of our method. Additional implementation details are  
5 included to facilitate reproducibility. Finally, we provide extended experimental results that further  
6 validate the effectiveness and versatility of EMLoC.

### 7 A Limitation and Future Work

8 While our emulator construction method enables significant memory savings and offers flexibility  
9 in placing LoRA modules, it relies on off-the-shelf SVD methods. These methods are primarily  
10 designed to preserve the inference-time behavior of the model, such as maintaining output logits,  
11 rather than the fine-tuning dynamics. As a result, the emulator may not fully capture the training  
12 characteristics of the original model, which could introduce suboptimalities during fine-tuning. This  
13 limitation is shared by all baselines. A promising direction for future work is to develop emulator  
14 construction strategies explicitly tailored to preserve fine-tuning behavior, such as gradient directions  
15 or parameter update trajectories. Such an approach could more faithfully mimic the training dynamics  
16 of the original model, potentially narrowing the performance gap between EMLoC and directing  
17 LoRA fine-tuning the original model.

### 18 B Social Impact

19 EMLoC lowers the memory barrier for fine-tuning large foundation models, broadening access for  
20 users with limited resources and promoting more inclusive participation in AI development. While  
21 this democratization has clear benefits, it may also increase the risk of misuse, such as facilitating  
22 the generation of harmful or biased content. Additionally, although EMLoC reduces memory usage,  
23 widespread fine-tuning of large models may still contribute to environmental concerns. We encourage  
24 responsible use and transparency in deployment to mitigate these risks.

### 25 C Supplementary Proofs and Derivations

#### 26 C.1 Verification of Corrected LoRA

27 In the main text, we proposed that the corrected LoRA module  $\Lambda^c = W_A^c W_B^c$  can be constructed by

$$W_A^c = W_A', \quad W_B^c = W_B' - \Delta, \quad (10)$$

28 where  $\Delta = W_A'^\top (W - W^\mathcal{E})$ . We now verify that substituting  $\Lambda^c$  into the original model indeed  
29 satisfies the desired condition

$$x^\top (W + \Lambda^c) = x^\top (W^\mathcal{E} + \Lambda) \quad \forall x \in \mathcal{V}_\Lambda. \quad (11)$$

30 Note that we can write any  $x \in \mathcal{V}_\Lambda$  as a linear combination of the columns of  $W_A'$

$$\exists \gamma \in \mathbb{R}^d \text{ such that } x = W_A' \gamma \quad \forall x \in \mathcal{V}_\Lambda. \quad (12)$$

31 We now substitute this into the left-hand side of Eq. (11) and simplify:

$$\begin{aligned}
x^\top (W + \Lambda^c) &= (W'_A \gamma)^\top (W + \Lambda^c) \\
&= (W'_A \gamma)^\top (W + W'_A W_B^c) \\
&= (W'_A \gamma)^\top (W + W'_A W_B' - W'_A \Delta) \\
&= (W'_A \gamma)^\top (W + \Lambda - W'_A W_A'^\top (W - W^\mathcal{E})) \\
&= \gamma^\top (W_A'^\top W + W_A'^\top \Lambda - (W_A'^\top W_A') W_A'^\top (W - W^\mathcal{E})) \quad (13) \\
&= \gamma^\top (W_A'^\top W + W_A'^\top \Lambda - I W_A'^\top (W - W^\mathcal{E})) \\
&= \gamma^\top (W_A'^\top W^\mathcal{E} + W_A'^\top \Lambda) \\
&= (W'_A \gamma)^\top (W^\mathcal{E} + \Lambda) \\
&= x^\top (W^\mathcal{E} + \Lambda) \quad \forall x \in \mathcal{V}_\Lambda,
\end{aligned}$$

32 which is exactly the right-hand side of Eq. (11).

## 33 C.2 Justification for Using SVD in LoRA Correction Preprocessing

34 In our LoRA correction algorithm, we introduce a preprocessing step where we apply SVD to the  
35 LoRA weights to obtain  $W'_A$  and  $W'_B$ . It is important to note that, in the subsequent derivation of  
36 the correction procedure, the only essential property of  $W'_A$  is that its columns form an orthonormal  
37 basis. Any other pair  $(W''_A, W''_B)$  that satisfies this orthonormality condition can also be used in the  
38 correction algorithm. In this section, we show that although SVD may appear to be an arbitrary  
39 choice, it is a valid and convenient one — and critically, the final corrected LoRA result remains the  
40 same for any orthonormal decomposition.

41 Formally, suppose we have

$$W'_A W'_B = W''_A W''_B = W_A W_B, \quad (14)$$

42 where both  $W'_A$  and  $W''_A$  have orthonormal columns. Our goal is to show that the resulting corrected  
43 LoRA modules are identical:

$$W'_A (W'_B - W_A'^\top (W - W^\mathcal{E})) = W''_A (W''_B - W_A''^\top (W - W^\mathcal{E})). \quad (15)$$

44 Given Eq. (14), this reduces to prove that

$$W'_A W_A'^\top = W''_A W_A''^\top. \quad (16)$$

45 Since the columns of both  $W'_A$  and  $W''_A$  are orthonormal and span the same subspace  $\mathcal{V}_\Lambda$ ,  $W'_A W_A'^\top$   
46 and  $W''_A W_A''^\top$  are identical—they both represent the projection onto  $\mathcal{V}_\Lambda$ . Therefore, the resulting  
47 corrected LoRA modules are equivalent, regardless of the specific orthonormal basis chosen. This  
48 justifies the use of SVD in our preprocessing step: while not strictly necessary, it provides one  
49 convenient and consistent way to obtain the required orthonormal basis.

## 50 D Additional Implementation Details

### 51 D.1 Usage of Existing Assets

52 In our experiments, we make use of publicly available datasets, models, and codebases in accordance  
53 with their respective licenses. Specifically, we use the InfoVQA and TextVQA datasets, which are  
54 released under the CC-BY license; the PMC-VQA and WC-VQA datasets, available under the CC  
55 BY-SA license; and the WebSRC and LAION-COCO datasets, released under the MIT license; and  
56 the ChartQA data set, release under the GPL-3.0 license. The model backbones and training code  
57 are based on InternVL2.5, which is licensed under MIT. All assets were used strictly for academic,  
58 non-commercial research in accordance with their intended terms. We thank the respective authors  
59 and communities for making these valuable resources publicly available.

### 60 D.2 Implementation Details

61 **Main Results.** Detailed information, including the number of base model parameters, LoRA ranks,  
62 and the number of trainable parameters, for the experiments in our main results (Table 1) can be  
63 found in Table 7. Note that we use different LoRA ranks for each baseline to ensure that the number  
64 of trainable parameters is approximately the same across methods.

Table 7: Information of experiments for main results.

Ratio	Method	Base Model Parameters	LoRA Rank	Trainable Parameters
100%	Original	8.1B	8	18.9M
50%	InternVL 4B	3.7B	10	18.7M
	Offsite	3.8B	64	18.9M
	UPop	3.8B	14	18.7M
	EMLoC	3.7B	8	18.9M
25%	InternVL 2B	2.2B	19	18.7M
	Offsite	2.2B	64	18.9M
	UPop	2.3B	19	18.9M
	EMLoC	2.3B	8	18.9M

Table 8: Information of the experiments for 26B and 38B large models.

	InternVL-26B	InternVL2.5-38B
Original Parameters	25.5B	38.4B
Emulator Parameters	5.0B	5.3B
LoRA Rank	6	4
Trainable Parameters	27.1M	33.5M

Table 9: **Performance comparison between EMLoC and zero-shot inference.** Note that fine-tuning with EMLoC requires less memory than performing inference. This enables users who can run inference to also benefit from fine-tuning, gaining additional performance improvements without increased memory cost.

Quantization	Method	ChartQA	DocVQA	InfoVQA	TextVQA	PMC-VQA	WebSRC	WC-VQA
$\times$	Zero-shot	83.8	92.0	69.6	78.5	48.6	76.5	43.1
	EMLoC 50%	<b>84.6</b>	<b>92.3</b>	69.9	78.8	<b>52.3</b>	<b>85.2</b>	<b>48.8</b>
	EMLoC 25%	84.2	<b>92.3</b>	<b>70.0</b>	<b>79.0</b>	51.6	79.6	46.2
$\checkmark$	Zero-shot	83.7	91.3	67.1	78.6	48.1	67.4	42.0
	EMLoC 25%	<b>84.2</b>	<b>91.9</b>	<b>67.5</b>	<b>79.0</b>	<b>50.9</b>	<b>78.1</b>	<b>44.8</b>

65 **Large Models.** Details of the experiments involving the 26B and 38B models (Table 2) are provided  
66 in Table 7. Note that for these experiments, we report performance on a subset of the test set due to  
67 limited computational resources, which made full-set inference infeasible within a reasonable time.

## 68 E Additional Results

### 69 E.1 Zero-shot Results

70 In Table 9, we report the zero-shot performance of the backbone models used in our experiments. As  
71 expected, fine-tuning with EMLoC leads to consistent improvements over the zero-shot baselines.  
72 While surpassing zero-shot performance is not surprising, it is important to highlight that EMLoC  
73 achieves these gains under the same memory requirements as inference. This means that users who  
74 are able to run inference on these models can also benefit from task-specific fine-tuning—without  
75 needing additional memory resources—making model adaptation more accessible in practice.

### 76 E.2 Computation Resource

77 We report the computational resource usage of our main experiments in Table 10, including memory  
78 consumption during both fine-tuning and inference. Additionally, we present the FLOPs per forward  
79 pass and the throughput, measured as the time taken for a full forward and backward pass on a single  
80 training sample. All metrics are evaluated on the same machine with a sequence length of 2048  
81 to ensure fair comparison. Notably, Offsite-tuning exhibits slightly higher throughput due to its  
82 emulator using significantly fewer layers than other baselines. Despite this, EMLoC achieves superior  
83 fine-tuning performance while maintaining comparable computational resource, demonstrating its  
84 efficiency and effectiveness.

### 85 E.3 Comparison with QLoRA

86 Although quantization-based methods like QLoRA are orthogonal to our focus—they do not re-  
87 duce fine-tuning memory relative to inference, we include a comparison to highlight the potential

Table 10: **Computational resource usage and performance comparisons of finetuning approaches.** Ratios follow the definitions in Table 1. Note that EMLoC achieves stronger performance than other baselines while maintaining comparable computation resource.

Ratio	Method	Fine-tuning memory (GB)	Inference memory (GB)	TFLOPs	Throughput (sample/sec)	PMC	WebSRC	WC-VQA
100%	Original	22.3	16.1	37.4	0.09	52.9	87.4	53.4
50%	InternVL 4B	15.4	8.1	18.9	0.17	50.6	84.8	48.6
	Offsite	13.8	16.1	16.1	0.20	51.0	76.1	45.9
	UPop	14.0	16.1	16.5	0.16	50.7	76.4	42.1
	EMLoC	14.2	16.1	17.8	0.16	52.3	85.2	48.8
25%	InternVL 2B	10.9	4.9	12.9	0.29	44.6	78.1	34.4
	Offsite	10.7	16.1	9.3	0.33	50.6	76.6	45.4
	UPop	11.3	16.1	10.3	0.22	50.9	76.6	44.1
	EMLoC	11.5	16.1	11.3	0.21	51.6	79.6	46.2

Table 11: **Comparison between EMLoC and QLoRA.** Quantization-based methods offer limited flexibility in memory reduction and can degrade inference performance, while EMLoC enables fine-tuning on consumer-grade 24GB GPUs without affecting the base model’s inference quality.

Method	InternVL2.5-26B		InternVL2.5-38B	
	Fine-tuning memory (GB)	WC VQA	Fine-tuning memory (GB)	WC VQA
Zero-shot	-	53.6	-	51
LoRA	72.9	-	94.8	-
QLoRA	38.1	39.0	43.4	43.6
EMLoC	<b>18.6</b>	<b>56.8</b>	<b>20.1</b>	<b>52.6</b>

advantages of our emulator-based approach. Results are presented in Table 11. Memory usage is measured in the same way as Table 10, with the exception that methods other than EMLoC are run across multiple GPUs without distributed data parallelism (DDP), and model weights are split across devices. Due to limited computing resources, we omit standard LoRA fine-tuning results for these large models.

By design, quantization methods offer limited flexibility and a fixed upper bound on memory savings. As shown in Table 11, even with 4-bit quantization, QLoRA cannot support fine-tuning of InternVL2.5-26B or InternVL2.5-38B on a 24GB consumer GPU. In contrast, EMLoC enables flexible memory reduction via emulator construction, allowing fine-tuning under memory budgets. Furthermore, while quantization performs well in most cases, it can degrade the original model’s performance—even during inference. In our experiments, quantizing InternVL2.5-26B and 38B significantly reduced accuracy, to the extent that QLoRA underperformed compared to the half-precision zero-shot baseline. These results underscore the practical benefits of emulator-based fine-tuning, offering greater memory flexibility without compromising inference performance.

#### E.4 Ablation Study on the Number of Calibration Data

EMLoC relies on a small calibration set  $\mathcal{D}_C$  to perform activation-aware SVD during emulator construction. In our main experiments, we use 64 samples as the default size for  $\mathcal{D}_C$ . To evaluate the sensitivity to this choice, we vary the number of calibration examples and report the results in Fig. 5. As shown, model performance plateaus after using 32–64 calibration samples, indicating that only a small amount of task-specific data is sufficient to construct an effective emulator. This highlights the practicality of EMLoC in low-resource scenarios.

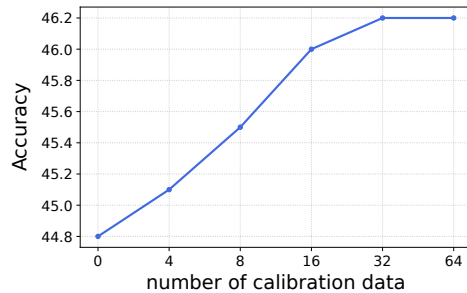


Figure 5: We plot performance on WC-VQA under different number of calibration data.

## E.5 Applying to Diffusion Model Personalization

To further assess the generalizability of EMLoC beyond text generation, we apply it to the task of personalizing a large text-to-image diffusion model. Specifically, we use DreamBooth [4]—a well-established personalization method—combined with LoRA to fine-tune FLUX.1-dev [2], a 12B state-of-the-art diffusion model. We conduct experiments on 8 subjects from the DreamBooth dataset, fine-tuning each for 500 steps without prior preservation. For evaluation, we report standard metrics used in prior works: DINO [1], CLIP-I [3], and CLIP-T scores. Due to the short fine-tuning duration in this experiment, we adopt standard SVD for emulator construction instead of activation-aware SVD, which would be less cost-effective in this setting.

Quantitative results are presented in Table 12. Note that normal fine-tuning requires over 35GB of memory, whereas EMLoC enables fine-tuning within a 24GB GPU budget. While EMLoC achieves slightly lower DINO and CLIP-I scores, it yields a higher CLIP-T score—likely due to slower convergence. We also show qualitative examples in Fig. 6. These results demonstrate that EMLoC can be effectively applied to image generation tasks, reinforcing its potential as a plug-and-play replacement for conventional LoRA fine-tuning with reduced memory requirements.

Table 12: **Quantitative results of applying EMLoC to diffusion model.** EMLoC achieves comparable, sometimes slightly lower, performance than direct fine-tuning while significantly reducing memory usage.

Method	Fine-tuning memory (GB)	DINO	CLIP-I	CLIP-T
w/o EMLoC	35.1	<b>0.652</b>	<b>0.851</b>	0.306
w/ EMLoC	22.9	0.615	0.831	<b>0.321</b>

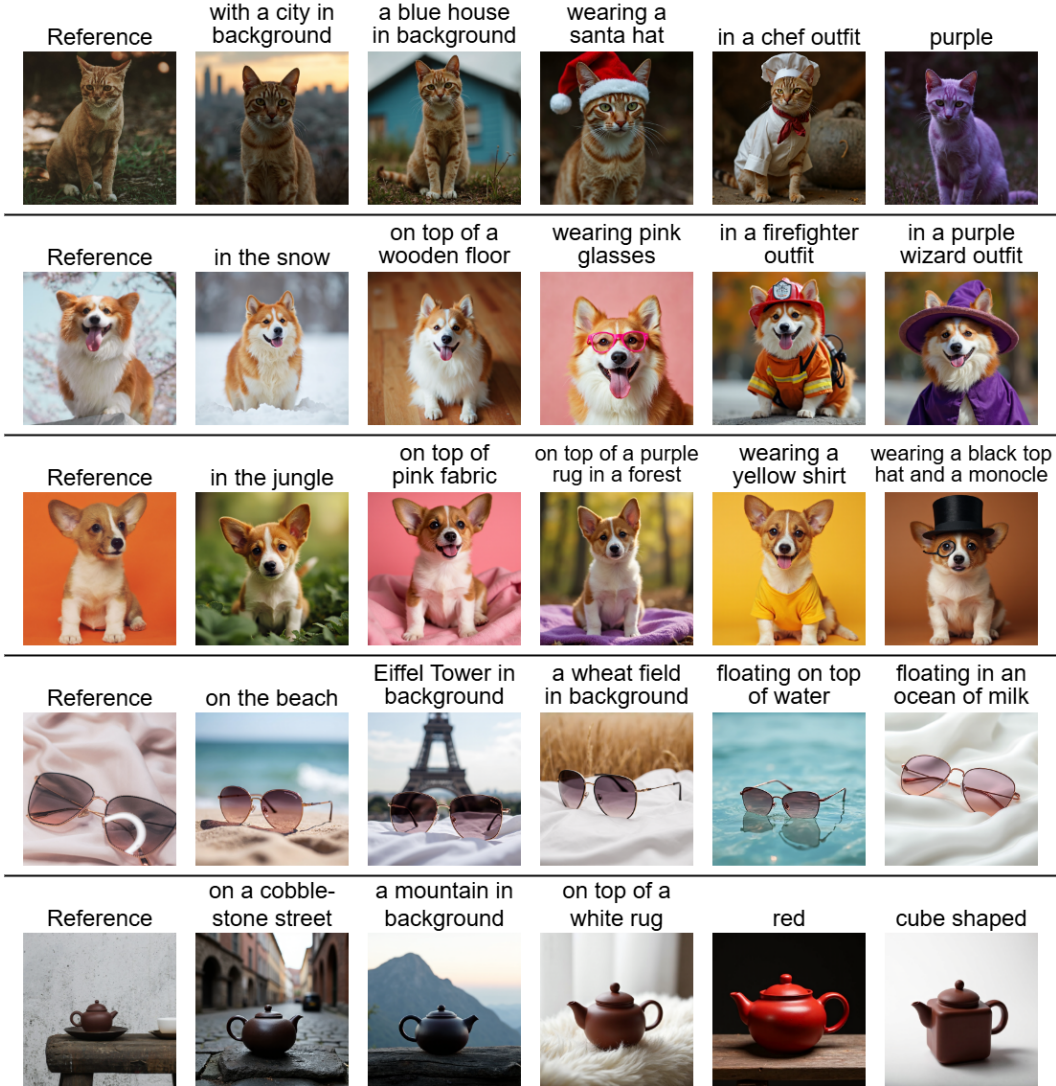


Figure 6: **Qualitative results of applying EMLoC to diffusion model personalization.** DreamBooth with LoRA is used to personalize the 12B FLUX.1-dev diffusion model, illustrating that EMLoC can be effectively extended to generative tasks beyond text.

## References

- [1] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.
- [2] Black Forest Labs. Flux. <https://github.com/black-forest-labs/flux>, 2024.
- [3] 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*, pages 8748–8763. PmLR, 2021.
- [4] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22500–22510, 2023.