Exploring Lora Merging Techniques in Virtual Try-On

Joseph McCoy
Stanford University
jmccoy03@stanford.edu

Songqi Pu
Stanford University
songqipu@stanford.edu

Abstract

With the rising interest in high-quality and customizable text-to-image generation, it's apparent that existing models encounter challenges—particularly due to token constraints that make restrict our ability to create hyper-specific prompts. This makes it difficult to to control fine details in image generation, which is essential for achieving the desired visual outputs. Here, we dive into exploring Low-Rank Adaptation (LoRA) merging techniques, using the SDXL latent diffusion model as our foundation. To effectively streamline the merging process and tailor it to our specific needs, we curated our own datasets. This approach allowed us to control critical parameters during training and explore the applicability of fine-tuning and merging techniques in real-world scenarios. After gathering these self-curated datasets, we trained multiple LoRA modules specifically designed for this purpose. We then employed various merging strategies as detailed in our methods section, which included common techniques like weighted averages and more advanced approaches such as ZipLoRA. Following the merging process, we evaluated the performance of the models using a series of different prompts to assess their ability to generate diverse and creative images. We calculated the similarity of the generated outputs to the original parent datasets, offering insight into how well the merged models preserved the unique characteristics of the individual LoRAs. Our findings show that the Mixed L2 Norm Weighted Average Merge technique had the highest similarity scores among the tested methods, doing a better job preserving the characteristics of each LoRA. In contrast, the other merging techniques exhibited more interference, which manifested in lower similarity scores. Moving forward, we plan to refine our algorithms and explore additional strategies, such as ZipLoRA, to further enhance the performance of text-to-image generation models.

1. Introduction

As deep learning models grow in complexity, there is increasing demand for more realistic and customizable text-to-image generation technologies. SDXL, a 2.6 billion parameter latent diffusion model, represents the current state-of-the-art in this space [1]. While SDXL is capable of generating high-quality images that reflect the user's prompt, it is constrained by the 77-token limit imposed by CLIP [2]. The initial tokens are weighted more heavily, and any tokens beyond the limit are processed in a separate chunk, potentially diminishing their influence. As a result, precise image generation depends heavily on prompt engineering, requiring users to concisely capture both broad concepts and fine details.

To reduce this dependence on highly engineered prompts, researchers have developed Low-Rank Adaptation (LoRA), which fine-tunes key cross-attention layers—where prompts and image features interact. LoRA has achieved results comparable to full model fine-tuning, while being more efficient in terms of time and computational resources [3]. These modules are lightweight, requiring only 10–20 images to train, and are typically task-specific, improving model performance for particular classes of images. This makes LoRA a powerful tool for adapting pre-trained diffusion models without the need for costly retraining. Though LoRA has proven highly effective for task-specific fine-tuning, the broader question of how to combine multiple LoRA modules remains an open area of research. Early efforts have revealed challenges such as parameter interference and inconsistent performance, prompting new approaches to more effectively integrate LoRA modules.

Developing more flexible and modular approaches to LoRA merging could expand the capabilities of models like SDXL, enabling more expressive and customizable image generation without the need for extensive retraining. Here, we explore how existing LoRA merging strategies perform when applied to modules trained on our own curated datasets, while also experimenting with tweaks and adaptations to improve compositional control and reduce interference during image generation. Our inputs are the images we curate for our LoRA modules and the merging methodologies, and our outputs are the SDXL images generated with the LoRA modules applied. Our results show that while some merging techniques do a good job of keep-

ing the unique traits of individual LoRAs, there are still challenges with interference that need to be addressed for better performance. Overall, these findings will help us improve methods that can boost the performance and customizability of text-to-image generation technologies.

2. Related Work

2.1. Fine-Tuning Diffusion Models

In the growing field of text-to-image generation, there is also a growing desire for image stylization based on the textual inputs. Existing techniques include Textual Inversion which learn text embeddings [4]; DreamBooth fine-tuning which adjusts the weights of the entire model [5], Low Rank Adaptations (LoRAs) [6], Custom Diffusion, which attempts to learn multiple concepts through expensive joint training from scratch [7] in addition to many other methods.

2.2. Parameter Efficient Fine-Tuning (PEFT)

In order to reduce the amount of storage and compute required to fine-tune stable diffusion models or LLMs, work has been done to limit the number of parameters that need retraining. One type of PEFT, the Low-Rank Adaptation (LoRA) is the state-of-the-art in this field [6]. Instead of fine-tuning the entire model, a LoRA aims to fine-tune the "residual" of it (by training ΔW).

$$W' = W + \Delta W \tag{1}$$

We can decompose the update matrix ΔW into low rank matrices such that $\Delta W = AB^T$ where $A \in \mathbb{R}^{n \times d}$, $B \in \mathbb{R}^{m \times d}$, and $d << n$. By fine-tuning A and B instead of W, our adjustment module becomes extremely small, which is helpful for memory storage and compute [8].

2.3. LCM-LoRA

Luo et al. proposed LCM-LoRA, an acceleration module for stable diffusion models. They train a Latent Consistency Model (LCM) using a one-stage guided distillation method and solve an augmented Probability Flow ODE (PF-ODE). They then distill this LCM to get the parameters for a LoRA module. The result is quality image generation that requires fewer sampling steps and reduced memory. In addition to the performance benefits, they demonstrate the merging of this LCM-LoRA with a specific style LoRA using a weighted linear combination of LoRA weights. They show that this approach could significantly enhance SDXL performance [9]. This study did great work at using a LoRA to speed up inference with the model, but it didn't really innovate when it came to using multiple LoRAs.

2.4. Mixure of LoRA Experts (MoLE)

Wu et al. sought to overcome the interference that comes with a (normalized) linear combination and the expensive nature of the reference tuning approach with manually crafted mask information. They developed the MoLE which incorporates a gating function for multiple LoRAs that learns the optimal composition weights based on a specific objective. This helps enhance the desirable characteristics while lessening the unfavorable ones. They found that this method outperforms their other benchmarks [10]. This method has only really been tested in the NLP domain and seems to work well for multiple LoRA modules. However, it does require training the gating function which could be expensive.

2.5. ZipLoRA

Shah et al. developed ZipLoRA, a method to cheaply and effectively merge independently trained style and subject LoRAs. They made 2 important observations when developing this idea. First, most LoRA weights are sparse, and have little effect on image generation. Second, The columns of the weight matrices of two independently trained LoRAs have different levels of alignment (cosine similarity), and summing highly aligned columns degrades performance of the merged model. Thus, they aimed at reducing this alignment interference while preserving the distinct functionality of each module, which they optimized for in the loss function. Their results showed that ZipLoRA outperformed earlier merging strategies [11]. This work made significant progress on the merging pipeline for a single subject and style LoRA, but has not extended to merging multiple subjects.

2.6. CLoRA

Building on these efforts, Han Salih Meral et al. introduced CLoRA, a method that, using contrastive learning, avoids merging weights altogether. Instead, CLoRA adjusts attention maps at inference time to spatially direct each LoRA's contribution to specific subjects or regions in the image. This selective activation enables greater compositional control and was shown to outperform both ZipLoRA and conventional merging techniques [12]. This work is probably the strongest candidate for LoRA merging thus far, but we were not able to access the codebase for this project.

3. Dataset and Features

For our project, we are working with a few self-curated datasets of 10-15 images each. Each image was self-labeled. We have collected the following sets:

1. Profile
2. FC Barcelona jersey
3. White shirt
4. Stanford sweater
5. Disney sweater

The barca and the white shirt datasets were collected from publically available images from the internet, and the others were collected by taking the images ourselves around campus. Figure 1 shows the shirt datasets and Figure 2 shows the profile dataset.

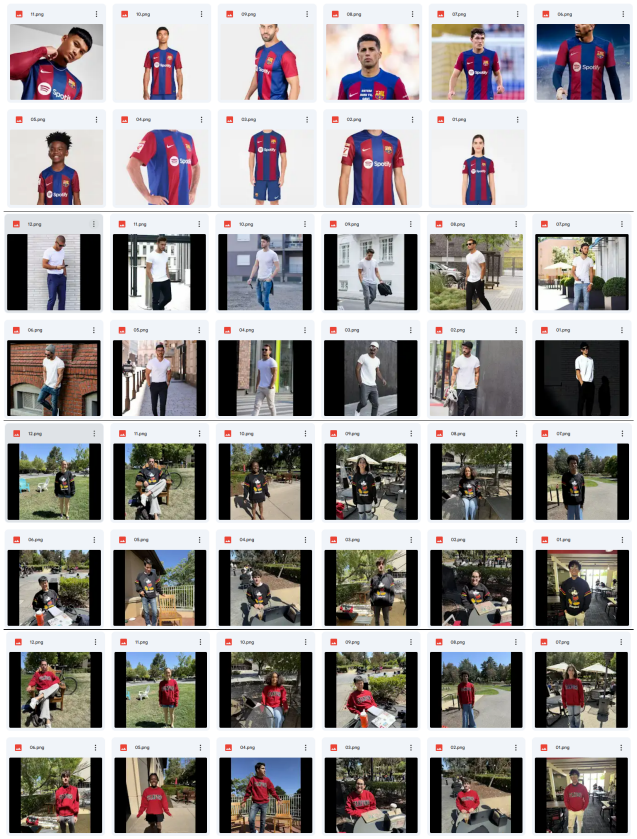

Figure 1: The images included in the datasets of different shirts that we curated. We have FC Barcelona jerseys (top), white shirts (second from top), Disney sweaters (3rd from top), and Stanford sweaters (bottom).

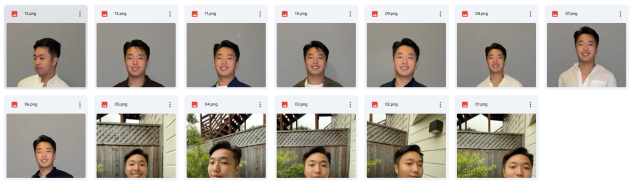

Figure 2: The images comprising our profile dataset that we curated.

We will use these datasets to train the LoRA modules that we will eventually merge.

4. Methods

4.1. Model Selection

For this project we will be working with a pretrained SDXL model ("stabilityai/stable-diffusion-xl-base -1.0"), a 2.6 billion parameter latent diffusion model which we will fine-tune with our LoRA modules. We chose this model because, while it is no longer the best stable diffusion model out there, it was cutting edge when it was released. Moreover, it's frequent use has led to a streamlined interface in Huggingface as well as ample research into fine-tuning it for specialized image generation.

4.2. LoRA Training

In order to control the parameters of our LoRA modules, like the rank, number of images, and the number of epochs, we trained our own using the train_text_to_image_sdxl.py script from the Huggingface stable diffusion library [8]. The parameters we used are:

- pretrained_model_name_or_path="stabilityai/stable-diffusion-xl-base-1.0"
- resolution=512 –random_flip
- train_batch_size=1
- max_train_steps 400
- rank 8
- num_validation_images 0
- num_train_epocs=1
- checkpointing_steps=100
- learning_rate=$1e-04$
- lr_scheduler="constant"
- lr_warmup_steps=0
- seed=42

The trained LoRA modules were then uploaded to Huggingface for easy use with SDXL.

Using this pipeline, we trained 4 LoRA modules for our project. The first two were pure, only using the dataset we curated for them. The next two were trained on a mixed dataset, containing the original images plus three from the opposite dataset.

4.3. LoRA Merging

4.3.1 Weighted Average

The weighted average, as shown in Figure 3, is one of the more common approaches for merging LoRA modules.

It involves taking the weighted average of the LoRA weights according to Eq.2,

$$\Delta\hat{W} = \sum_{i=1}^{N} w_i \cdot \Delta W_i, \tag{2}$$

where $\Delta\hat{W}$ are the parameters of the merged LoRA, ΔW_i are the parameters of each individual LoRA, and w_i are the

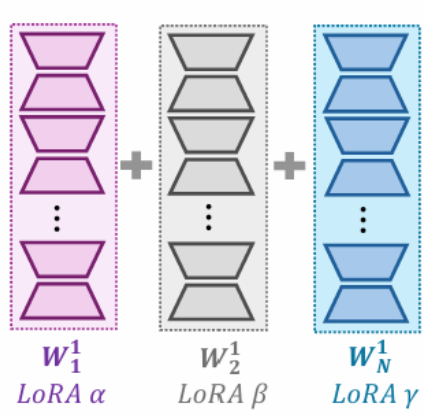

Figure 3: Weighted average, which commonly applies the same composition weight W_i to all layers of the ith LoRA [10].

composing weights for each LoRA such that $\sum_{i=1}^{N} w_i = 1$. It is common practice to weight the LoRA contributions as such to preserve the model's general capabilities, but it should be observed that a LoRA will begin to lose it's individual characteristics as the composing weight is reduced.

We wrote a script for this operation, and for our experiments we use $N = 2$ and $w_i = .5$ for all $i \leq N$.

4.3.2 L2 Norm-based Weighted Average

The L2 norm-based weighting average is the same as the method in the previous section with one key difference: we weight each set of parameters proportionally by the L2 norm. The result is a slightly different value for the composing weights, as seen in Eq 3. This has the effect of making the LoRAs with a larger magnitude contribute more heavily to the merged result.

$$w_i = \frac{||\Delta W_i||_2}{\sum_{k=1}^{N} ||\Delta W_k||_2} \tag{3}$$

We wrote a script for this operation, and for our experiments we also use $N = 2$.

4.3.3 ZipLoRA

This method works by learning the composing weights for each column of ΔW_i for both LoRas. it does this by (1) minimizing the difference between the images generated with the merged LoRA and the those from the original LoRA models and (2) minimizing the cosine similarity between the columns of the two LoRA modules. See Figure 4 for a pictorial overview. The goal of this objective is to conserve the properties of each LoRA while minimizing signal interference [11].

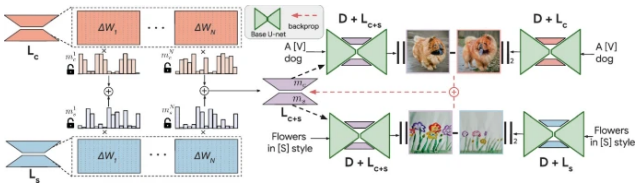

Figure 4: Overview of ZipLoRA [11]

In order to complete this operation for our project, we utilized an implementation of the algorithm that was written by user mkshing [13]. While being a great starting place, this implementation is a bit depreciated, so we've been working to make it functional with current libraries.

4.4. Evaluation

For evaluation, we compare the outputs of the base model (without fine-tuning) and the pure LoRA fine-tuned model against the outputs from each of the four merged LoRA fine-tuned models. We will generate images for 5 different conditions using each LoRA. The prompts are:

1. "A portrait of a psq_person smiling at the camera, wearing a barca24_jersey."
2. "A portrait of a psq_person smiling at the camera, wearing a barca24_jersey and facing to the side."
3. "A portrait of a psq_person smiling in a selfie, wearing a barca24_jersey with Nike shoes."
4. "A portrait of a psq_person smiling at the camera, wearing a barca24_jersey and standing in front of a wall."
5. "A portrait of a psq_person smiling at the camera, wearing a barca24_jersey and holding a soccer ball on a field."

Using the images we generate with these prompts, we will calculate DINO similarity scores using Facebook's DINO-v2. To properly evaluate our LoRA's performance, we will first compute the average DINO embedding for the images in our datasets. Then we will embed each image using DINO-v2 and calculate the cosine similarity between the output image and both of the average dataset embeddings. Finally we will compute an average score for each LoRA module.

5. Results

5.1. LoRA Training

Using the 5 datasets we curated above, we successfully created a LoRA module for each of them. The profile and the barca LoRAs worked pretty well (see Figure 8.), but the LoRAs for the white shirt, the disney shirt, and the Stanford shirt did not produce a high quality result, as shown in Figure 5.

We hypothesized that this was because the images in those datasets had too much variablity, which made the LoRA module harder to train using our pipeline. As a result, we only continued on with the Profile and Barca LoRAs, which are accessible below:

1. profile [link]
2. barca [link]
3. profile_mixed [link]
4. barca_mixed [link]

Figure 5: Images of the Failed LoRAs, white shirt, Disney sweater, and Stanford sweater.

5.2. LoRA Merging

Subsequently, we merged the Profile and the Barca LoRAs according to the methods we outlined in Section 4.3. We obtained the following merged LoRA modules:

1. Weighted Average [link]
2. L2 Norm Weighted Average [link]
3. Mixed Weighted Average [link]
4. Mixed L2 Norm Weighted Average [link]

5.3. Image Generation

With these LoRA modules we have trained and merged, it was time to use them for image generation. Our collection resulted in the following 7 settings we've used:

1. SDXL with no LoRA fine-tuning
2. SDXL with Barca fine-tuning
3. SDXL with Profile fine-tuning
4. SDXL with merged LoRA via Weighted Avg fine-tuning
5. SDXL with merged LoRA via L2 Norm Weighted Avg fine-tuning
6. SDXL with merged LoRA with data swap via Weighted Avg fine-tuing
7. SDXL with merged LoRA with data swap via L2 Norm Weighted Avg fine-tuning

For each of these settings, we ran the model on 5 different prompts, which are shown in Figure 8, (refer to section 4.4 for the prompts). Qualitatively, we decided that the mixed dataset LoRAs had the best images.

5.4. Evaluation

After collecting all of the images, we computed average similarity scores for each LoRA module using DINO-v2 embeddings from the generated images and the source datasets (see Section 4.4 for more detail). This means that we have two output numbers for each LoRA module that indicates its performance.

See Figure 6 for the exact metrics. We see that the mixed dataset L2 norm weighted average LoRA has the highest similarity of all the ones we merged.

	Profile Avg Similarity	Barca Avg Similarity
None	0.5334	0.6445
Profile	0.6823	0.6293
Barca	0.4036	0.7599
Weighted Avg	0.5703	0.7168
L2 Norm Weighted Avg	0.5986	0.6840
Mixed Weighted Avg	0.6053	0.7087
Mixed L2 Norm Weighted Avg	0.6154	0.7179

Figure 6: Table of our average similarity scores compared to each dataset for each LoRA

5.5. Data Visualization

To help us analyze our results better, we plotted the different similarity metrics on a few graphs. In Figure 7 we have the similarity scores for the images generated from each LoRA against the average embedding from the Profile dataset. We see that the LoRA trained only on the profiles performed the best, with all of the merged LoRAs having a lower similarity score. Among the merged LoRAs, the Mixed Norm-based weighted average had the best similarity.

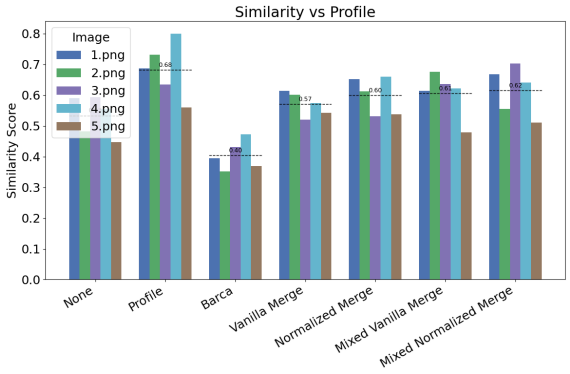

Figure 7: DINO similarity vs. Profile

Figure 8: Images generated with each of our 5 prompts and each of our 7 model configurations.

In Figure 9 we have the similarity scores for the images generated from each LoRA against the average embedding from the Barca dataset. From these metrics, the Barca LoRA has the highest similarity, but the merged LoRAs performed pretty well; the decrease is less significant than with the Profile similarities. Out of the merged LoRAs, the normal weighted average and the mixed dataset L2 norm weighted average had the same average similarity.

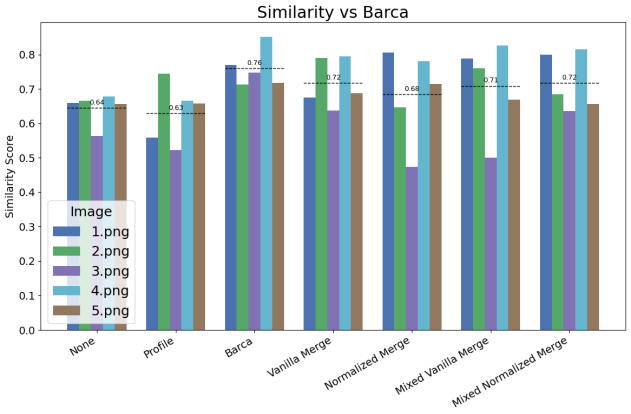

Figure 9: DINO similarity vs. Barca

In Figure 10 We have the DINO similarity scores for each LoRA module with each dataset separated by the prompt (or the image name). This shows us that the prompt we use also seems to have an impact on the similarity scores. Prompt 3 seemed to generate a pretty equal score for both sets, but the others all seemed to slightly favor the Barca set.

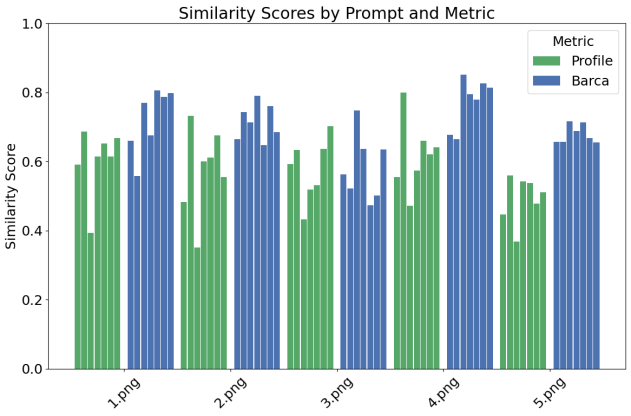

Figure 10: DINO similarity by Prompt

In figure 11 we have the average similarity score for each LoRA module, split by the dataset it was compared to. We see that aside from the Profile LoRA, every other module, including SDXL without any fine-tuning had higher similarity to the Barca dataset. This makes sense, as it results from the difficulty of recreating a specific face and the ease of recreating patterns on a shirt.

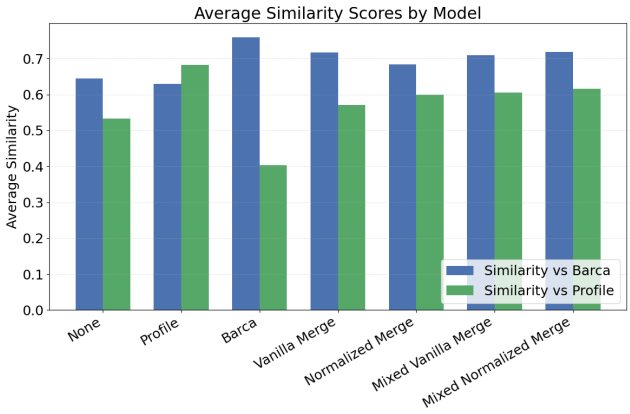

Figure 11: Average DINO similarity by model

5.6. Missing ZipLoRA

In our methods section, we detailed ZipLoRA as a technique for merging two LoRAs together. As we mentioned earlier, the codebase we were working with was a bit depreciated. We were unfortunately unable to run the merging algorithm, which explains the lack of a ZipLoRA example above. More specifically, we found out that there were a couple of big refactoring from key libraries including huggingface and diffusers that the ZipLora implementation relies on. These libraries had initial incompressibilities we had to resove. After resolving them, we were able to kick off the training loop, however, we then found out that the way ZipLora is implemented is not compatible with the LoRAs we've trained using the latest libraries due to the mismatch in the naming schemes of keys. We regret that there was not enough time to explore this further.

6. Discussion

In this work, we explored different ways of merging LoRAs trained on self curated datasets and observed the impacts of the different techniques on model performance. Our primary evaluation metric was similarity, which we obtained through the DINO-v2 embeddings for the images. We computed the cosine similarity scores between the generated images and their corresponding dataset averages for two target groups: the Barca dataset and the profile dataset. The higher these scores, the more similar the generated images are to the training images, and this would represent an improved ability to retain the characteristics of the training dataset when generating new images.

Quantitatively, we are able to see that each of the individual LoRAs has a higher similarity to our targets than that of the plain stable diffusion. In addition, across all of our

merged LoRAs, the similarity scores are generally higher than that of the plain stable diffusion as well. However, their individual categorical result is lower than that of the individual Profile and Barca LoRAs, as expected. Within the merged LoRAs, the Mixed Normalized Merge performs the best.

Qualitatively, we're able to see clearly that the Profile and Barca LoRAs each learned the concepts of the face and jersey. In Profile LoRA, we can see that the features of the face is learned really well. In the Barca LoRA, we can see that the overall contours of the jersey is learned well although the letters and the details of the jersey still is not perfect. However, we can still easily see a substantial difference between the results from these 2 LoRAs against the plain stable diffusion.

Across all of our merged LoRAs, we're able to see that although the Barca jersey has maintained it's overall shape and colors, the facial features from the Profile LoRA is generally not as well preserved. Out of all the merged LoRAs, the facial features are preserved best in the Mixed L2 Norm Weighted Average LoRA. This is also indicated through the Dino score as mentioned above.

We regret that we were not able to fully replicate ZipLoRA's functionality due to the nested incompatibility of packages and LoRA standards from late 2023 to now. However, we also learned a valuable lesson that in future work, we should be careful to have the versions of software and libraries explicitly written out, since this is a fast moving industry, and even the most popular libraries may have refactored code or drastic changes on a monthly basis.

7. Conclusion

Here, we explored the nuances of building your own small image datasets, training LoRA modules on those datasets, and trying different ways of merging them to allow for more complex fine-tuning of a popular stable diffusion model SDXL. We found that all of our merged datasets experience some amount of interference that reduced the performance of the model. Out of our techniques, the L2 norm-based weighted average exhibited the best similarity to the training sets, though still performed worse than the pure LoRAs themselves. This indicates that simply averaging the LoRA parameters together could be a bit naive, and potential demands something with a bit more compute. If we had more time and more compute we would pursue some of these more complete methodologies, like ZipLoRA (if we could get it to work) and CLoRA, which innovatively leverages contrastive learning to merge the LoRA modules together. In addition, it could be interesting to explore how merging more than to LoRAs impacts the dynamics of this performance. Overall, though our work helps understand this problem fairly well, there is much work to be done in the realm of parameter efficient fine-tuning of stable diffusion models, especially since the technologies are rapidly evolving.

8. Contributions and Acknowledgments

For this we utilized the following codebases:

1. Huggingface
2. mkshing/ziplora
3. pytorch
4. Facebook's DINO-v2

Songqi's contributions: Collected and labeled datasets, Implemented the training of LoRAs, Implemented the merging techniques of the LoRAs. Worked on getting the ZipLoRA to work. Ran DINO evaluations. Did parts of the writing (Section 6).

Joey's contributions: Collected and labeled datasets, Worked with trying to make the ZipLoRA implementation functional, Did most of the report writing (Abstract through Results and then Conclusion). Processed the DINO data into graphs and made/sourced the figures.

References

[1] D. Podell, Z. English, K. Lacey, A. Blattmann, T. Dockhorn, J. Müller, J. Penna, and R. Rombach, "Sdxl: Improving latent diffusion models for high-resolution image synthesis," https://arxiv.org/abs/2307.01952, Jul. 2023, arXiv preprint arXiv:2307.01952. 1

[2] S. Mukherjee, "Decoding long-clip: Understand the power of zero-shot classification," https://www.digitalocean.com/community/tutorials/long-clip-zero-shot-classification-text-analysis, Dec. 2024, digitalOcean. 1

[3] L. Tsaban, "Lora training scripts of the world, unite!" https://huggingface.co/blog/sdxl_lora_advanced_script, Jan. 2024, hugging Face. 1

[4] R. Gal, Y. Alaluf, Y. Atzmon, O. Patashnik, A. H. Bermano, G. Chechik, and D. Cohen-Or, "An image is worth one word: Personalizing text-to-image generation using textual inversion," https://arxiv.org/abs/2208.01618, Aug. 2022, arXiv preprint arXiv:2208.01618. 2

[5] N. Ruiz, Y. Li, V. Jampani, Y. Pritch, M. Rubinstein, and K. Aberman, "Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation," https://arxiv.org/abs/2208.12242, Mar. 2023, arXiv preprint arXiv:2208.12242. 2

[6] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," https://

openreview.net/forum?id=nZeVKeeFYf9, Oct. 2021, openReview. 2

[7] N. Kumari, B. Zhang, R. Zhang, E. Shechtman, and J.-Y. Zhu, "Multi-concept customization of text-to-image diffusion," https://arxiv.org/abs/2212.04488, Jun. 2023, arXiv preprint arXiv:2212.04488. 2

[8] Cloneofsimo, "Cloneofsimo/lora: Using low-rank adaptation to quickly fine-tune diffusion models," https://github.com/cloneofsimo/lora, 2024, gitHub. 2, 3

[9] S. Luo, Y. Tan, S. Patil, D. Gu, P. von Platen, A. Passos, L. Huang, J. Li, and H. Zhao, "Lcm-lora: A universal stable-diffusion acceleration module," https://arxiv.org/abs/2311.05556, Nov. 2023, arXiv preprint arXiv:2311.05556. 2

[10] X. Wu, S. Huang, and F. Wei, "Mixture of lora experts," https://openreview.net/forum?id=uWvKBCYh4S, Oct. 2023, openReview. 2, 4

[11] V. Shah, N. Ruiz, F. Cole, E. Lu, S. Lazebnik, Y. Li, and V. Jampani, "Ziplora: Any subject in any style by effectively merging loras," in *Computer Vision – ECCV 2024*. Springer Nature Switzerland, 2024, springerLink. 2, 4

[12] T. H. S. Meral, E. Simsar, F. Tombari, and P. Yanardag, "Clora: A contrastive approach to compose multiple lora models," https://arxiv.org/abs/2403.19776, Mar. 2024, arXiv preprint arXiv:2403.19776. 2

[13] Mkshing, "mkshing/ziplora-pytorch: Implementation of "ziplora: Any subject in any style by effectively merging loras"," https://github.com/mkshing/ziplora-pytorch/tree/main, 2023, gitHub. 4