Fashion Synthesis Assistant Tool based on Stable Diffusion Model

Amrita Moturi[1]
UC Berkeley, CS 182
amoturi@berkeley.edu

Mariana Vasquez[1]
UC Berkeley, CS 182
marianavasquez@berkeley.edu

Maya Zheng[1]
UC Berkeley, CS 182
mayaz@berkeley.edu

Abstract

This project presents a fashion synthesis assistant tool that allows users to generate fashion images by simply entering a text prompt. The tool is developed based on the Stable Diffusion model, which is effective for text-to-image generation. By fine tuning the diffusion model on a product fashion dataset using low-rank adaptation (LoRA), our fine-tuned model can synthesize visually appealing images of clothing. The experiments demonstrate that fine-tuning a diffusion model on a fashion data can improve the quality of text-to-image generation tailored to our specific task and LoRA is a robust technique for the fine-tuning process. Our code is available at https://github.com/amritamo/fashion_stable_diffusion_finetuned.

1. Introduction

Fashion synthesis is a technique that leverages computer algorithms to generate, manipulate, or enhance fashion-related images [5]. It is very useful in practice, especially in the fashion industry, for example, in terms of virtual try-on applications, styling, fashion recommendations and personalization. However, fashion synthesis is a challenging task since it requires placing a reference garment on a body model at an arbitrary pose. With the great success of AI and deep learning in image processing and computer vision, people started to delve into the use of AI, especially generative models, in fashion synthesis and have proposed various AI-enabled (such as GAN-based or diffusion model based) techniques for fashion synthesis [3,7,12].

This project aims to implement a stable diffusion technique based fashion synthesis assistant tool that automatically generates clothing design based on user input through the use of a text prompt. This is inspired by the power of stable diffusion that is a deep generative artificial neural network generating high quality images conditioned on text descriptions [9,11,12]. Text-to-image fashion synthesis is relatively unexplored compared to other fashion synthesis approaches [12]. The ability to generate detailed images from scratch through the use of a text prompt provides a convenient interface for users to express their intention, meanwhile the diffusion techniques may creatively generate something new and "surprising" to the users. Thus our developed design assistant can provide fashion students or designers with new ideas and design fashion images based on the input preferences and design attributes. It may also be used by costume designers when they seek inspiration for year or season specific designs.

Our approach is to use a pre-trained KerasCV model for high-performance image generation using Stable Diffusion [6] as our baseline to generate images from text prompts. In order for stable diffusion to generate meaningful and relevant images, we fine-tune the model on a fashion product image dataset [4] with LoRA for our specific task related to fashion design [1,2,6,8,9]. We further examined some variables in the fine tuned model, and studied the trends by varying their values.

2. Methodology

2.1 Stable Diffusion Model

Diffusion models consist of two processes [11, 12]. The first process is a *forward diffusion* process that iteratively adds small amount of Gaussian noise to the sample in T steps given a data point, sampled from a real data distribution $x_0 \sim q(x)$, producing a sequence of noisy samples $x_1, x_2, \cdots, x_T$ satisfying

$$q(x_t|x_{t-1}) \sim N(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I), \quad q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$

where $\beta_t \in (0, 1)$ are a variance schedule. For an arbitrary timestep t, x_t can be sampled directly from x_0:

$$q(x_t|x_0) \sim N(x_t; \sqrt{\alpha_t}x_0, (1-\alpha_t)I)$$

where $\alpha_t = \prod_{i=1}^{t}(1-\beta_i)$. Hence the data sample x_0 gradually loses its distinguishable features as timestep t becomes larger. When $T \to \infty$, x_T is diffused to an isotropic Gaussian distribution $N(0,1)$.

The second process is the *reverse diffusion* process, which starts with a Gaussian noise and iteratively refines it towards a realistic sample by learning the conditional probabilities

$$p_\theta(x_{t-1}|x_t) = N(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

where μ_θ and Σ_θ are the learnt mean and variance of the reversed Gaussian distribution.

Stable Diffusion is a latent text-to-image diffusion model capable of generating photo-realistic images given a text prompt [9]. It consists of three components: the variational autoencoder (VAE), U-Net, and a text encoder. The VAE encoder compresses the image from a pixel space to a smaller dimensional latent space, capturing fundamental semantics of the image. Gaussian noise is iteratively applied to the latent representation during the forward diffusion process. The U-Net denoises the output from forward diffusion backwards to obtain a latent representation.

The VAE decoder generates the final image by converting the representation back into the pixel space.

2.2 LoRA

LoRa, standing for Low-Rank-Adaptation, is a fine-tuning technique that freezes the pre-trained model weights and injects trainable rank decomposition matrices into the layer of the Transformer architecture, reducing the number of trainable parameters for downstream tasks [6]. Specifically, for a pre-trained weight matrix $W \in R^{d\times k}$, we constrain its update with a low-rank decomposition $\Delta W = BA$, where $B \in R^{d\times r}, A \in R^{r\times k}$ and rank r is much smaller than d and k. Thus $h = Wx + \frac{\alpha}{r}\Delta Wx = Wx + \frac{\alpha}{r}BAx$ where α is a hyperparameter and it is usually within [0,r]. See Fig.1 for illustration. During the training, W is frozen and does not receive gradient updates, while A and B contain trainable parameters. Matrix A is usually initialized using Gaussian distribution and B is initialized to be zero.

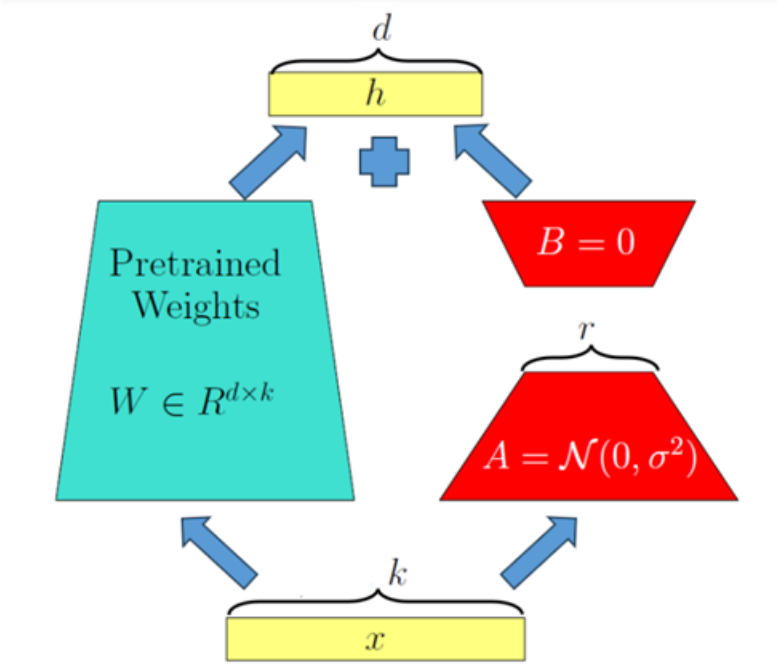

Fig 1: Illustration of LoRA architecture

2.3 Baseline Model

We used KerasCV's implementation of a pre-trained stable diffusion model [10] to generate novel images based on a text prompt. KerasCV's implementation includes XLA compilation and mixed precision support, which are key features that contribute to the performance optimization

of machine learning models and achieve significant performance boosts. This serves as our baseline, which can produce fashion images by clothing specific prompts.

We tested it to observe how it performs when used by our target audience. Some testing results are shown below.

Fig 2. Prompt: “maroon strapless top”

Fig 3. Prompt: “photograph of a 90s style top appropriate for winter”

From these results, we can see that the model does generate clothing that fulfills the requirements. However, there are limitations, for example, in Fig 2 and Fig 3, we can see that many of the images generated are wrong types of clothing or are not clothing at all. We can also see that some of the clothing pieces generated are out of frame.

Therefore, we are fine tuning the model on a fashion and clothing specific dataset.

2.4 Preprocessing the dataset

The dataset we have chosen is composed of images of clothing marked with an ID and categorized with descriptors such as gender, primary fashion category, and secondary fashion category. The breakdown of the categories are as follows: gender is broken up into 50% men and 42% women, the primary fashion category is divided into 48% apparel and 25% accessories and

26% other, the secondary category is broken up into 35% topwear and 17% shoes and 49% other. There are various other categories such as type of clothing, color, season, year, usage, and brand name. The year span is from 2007 to 2019. There are 10 categories total. Ideally, there would be a correlation between clothes that belong to the same season, same year, and other such categories that could display a correlation in the output.

The labels of the images were all separated by categories, therefore, the labels needed to be combined together to form a cohesive caption in a sentence format for better prompting. We utilized the generic format of "an image of a *year season color gender articleType*" for example, "an image of a 2011 Fall Blue Man's Top". However, in the given dataset, all the articleTypes were labeled in the plural form, in order to not affect the model's fine tuning and interpretation of the prompt, the labels were changed to fit the number of articles in the corresponding image.

The data in the code is split for training, testing, and validation. We utilized randomization to ensure that each part of the split is a representative sample of the dataset. We have a seed that is fixed in order to reproduce our results. A requirements.txt file with the necessary packages are included in the codebase.

3. Experiments

We used Google Colab Pro to fine-tune the KerasCV pre-trained model, utilizing both A100 and V100 GPUs. The A100 and V100 GPUs offered by Google Colab Pro provided us with accelerated processing capabilities, significantly reducing training times and allowing for more efficient experimentation and model refinement. By incorporating LoRA and enabling mixed-precision, we successfully managed to mitigate memory constraints during execution.

3.1 Variables tested

In our performance analysis, we systematically investigated the impact of three crucial variables on the overall performance of our model. Specifically, we scrutinized the influence of LoRa rank, batch size and learning rate.

We initially adopted hyperparameters suggested in a HuggingFace script found at [13], incorporating a LoRA rank of 4 and a learning rate of 1e-4, and then iteratively adjusted them based on our findings. We tested several different learning rates ranging from 1e-6 to 1e-4, adjusted the batch size between 1 and 6, and a rank between 2 and 10.

During our experimentation, we observed that varying the LoRa rank significantly affected model outcomes. Interestingly, we found that an increase in rank yielded improved results, with rank 2 demonstrating the least favorable performance, while rank 4 emerged as the most

effective among the configurations tested. We kept incrementing the rank until we reached a point where our images were not visually more appealing.

lora_rank	lr	batch_size
2	5e-05	3
4	5e-05	1
4	5e-05	3
4	5e-05	6
4	5e-05	3
4	0.0001	3
4	1e-06	3
6	0.0001	1
10	5e-05	3

Fig 4. Summary of Some Variables Tested

4. Results and Evaluation

Fréchet Inception Distance (FID) is a metric used to evaluate the quality of generated images in machine learning. It quantifies the similarity between the distributions of real and generated images by measuring the distance between their feature representations extracted from a pre-trained Inception model. Lower FID values indicate better image generation.

Our implementation for calculating FID is based on the tools provided by Hugging Face. This approach allows us to leverage their advanced machine learning models and frameworks to conduct our evaluation effectively.

For our analysis, we utilized smaller sets of images for comparison, rather than the large ones typically employed in FID calculations. The 3 images in Fig 5. were randomly selected from the test dataset and the prompts were used to generate images from the various models. The generated images and the known images from the dataset were then used to calculate the FID values.

Prompt: “an image of a 2011 Casual Fall White Men's Sweater”

Prompt: “an image of a 2012 Casual Summer Grey Women's Dress”

Prompt: “an image of a 2011 Sports Fall White Men's T-Shirt”

Figure 5: Raw Test Dataset images with Prompt (selected for FID evaluation)

For KerasCV’s stable diffusion base model, we derived a FID value of around 478.6. For visualization the images generated by the model are:

Prompt: “an image of a 2011 Casual Fall White Men's Sweater”

Prompt: “an image of a 2012 Casual Summer Grey Women's Dress”

Prompt: “an image of a 2011 Sports Fall White Men's T-Shirt”

Figure 6: Images generated by Base Model with test dataset prompts

Fine Tuned Model results

After fine-tuning the baseline model, we found that the model that performed the best and produced the highest quality images, which we confirmed with the FID score, had the parameters batch size 3, LoRA rank 4, and a learning rate of 5e-05.

From our results as seen in Fig 7 and 8, we determined that the optimal rank for this dataset is rank 4 and the optimal batch size is 3. Although decreasing learning rates seemed to improve the FID scores from Fig 9, from testing with more epochs, we concluded that the optimal value is 5e-05.

In Fig 10, we have the Early, Mid and Late stage training results of our best model for the 3 prompts from the test dataset as mentioned in Fig 5. Fig 11 has the Early and Mid stage training results of another model with batch size 6 for comparison. From the 3 stages, we can see that as training continues, the images generated increase in quality and fit more accurately to the prompt given. We can see that the results of our best model significantly exceeds the results of the other model, both in terms of the FID scores at each stage, and the aesthetic appeal, quality and accuracy of the images.

Fig 12. showcases several of the training and validation plots for models that were trained on a learning rate of 5e-5. Our best model is shown in the bottom left corner, Fig 12c, with a LoRA rank of 4 and batch size of 3. For all plots, the training loss tends to have a decreasing trend, whereas the validation loss is unstable and varies greatly. This is likely due to the very small dataset we used for training and the even more limited validation dataset used for tracking the validation loss. The small validation dataset may not adequately represent the data in the training set. This lack of generalization can lead to high variance in the validation loss, as the model might perform very differently on the limited set of examples available in the validation set. As such, we decided to choose our best model of batch size 3, LoRa rank 4 based on the FID scores calculated after training the model instead of the training and validation curves.

Our analysis revealed that the fine-tuned model exhibited more stable outputs compared to the original model. Although there is room for further aesthetic enhancements in the results, the discernible effects of fine-tuning underscore its positive impact on the model's performance for the purpose of fashion synthesis.

Rank	lr	Batch size	FID
2	1.00E+06	3	387.286438
4	1.00E+06	3	325.385376
10	1.00E+06	3	421.9648743

Fig 7: Results of changing LoRA rank

Rank	LR	Batch size	FID
4	5.00E-05	1	542.612854
4	5.00E-05	3	425.1402893
4	5.00E-05	6	460.951416

Fig 8: Results of changing Batch size

Rank	lr	Batch size	FID
4	1.00E-04	3	390.9792786
4	1.00E-06	3	370.3812561
4	1.00E-03	3	420.7935486

Fig 9: Results of changing Learning Rate

(a) Early Stage Training (after 1 epoch, FID = 452.5)

(b) Mid Stage Training (after 11 epochs, FID = 293.9)

(c) Late Stage Training (after 20 epochs, FID = 284.6)

Fig 10: Best Model of batch size 3, LoRa rank 4, leaning rate 5e-05

(a) Early Stage Training (after 2 epochs, FID = 461.0)

(b) Mid Stage Training (after 10 epochs, FID = 356.8)

Fig 11: Other model of batch size 3, LoRa rank 6, leaning rate 5e-05

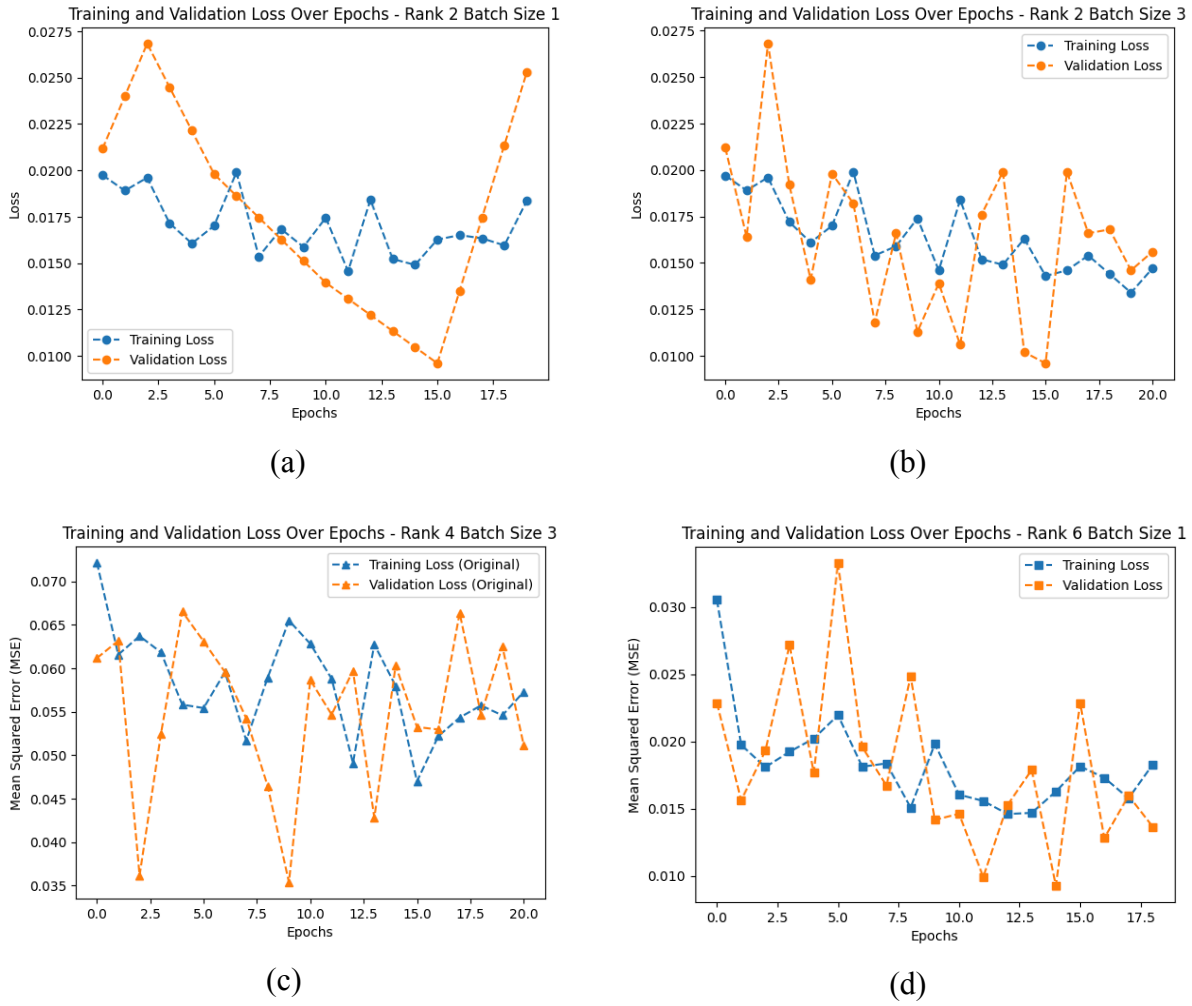

Fig 12. Training and Validation Curves using LoRA

Limitations

Our current study is significantly constrained by two major limitations. The first limitation is resource limitation. Due to limitations in compute resources, budget constraints, and access to proper GPUs, our investigation heavily relied on Google Colab, which offered only limited computational units and memory. The use of KerasCV for fine-tuning posed additional challenges, necessitating restrictions on various factors and parameters. These constraints included time limitations for training, a small dataset size (850 images sampled from a pool of 44,000 due to GPU disconnection issues), a restricted number of epochs (limited to 20), and a constrained number of images used for FID testing.

The second limitation is dataset limitations. The dataset itself presented challenges, featuring noise, missing labels, and miscellaneous, non-clothing data that proved difficult to filter out completely during preprocessing. Some labels were missing or filled as "None," leading to the generation of captions with inaccurate or nonsensical descriptions during training, for example, “an image of a 2014 Casual Summer White Women's None". Furthermore, the dataset, initially believed to contain only images of clothing without models, included instances of people wearing multiple clothing items. This unexpected presence of models affected the generated results, as clothing pieces often appeared with accompanying models in the fine-tuned model's outputs. These limitations and challenges collectively impacted the quality of our results in this investigation.

5. Conclusions and Future work

We have presented a fashion synthesis assistant tool that is developed based on a Stable Diffusion model. Our approach is based on the KerasCV implementation of the Stable Diffusion model[10] and fine tune the model on a fashion product image dataset. The tool allows the user to enter a text prompt as input, based on which a fashion image is automatically generated. While the current results are far from aesthetically pleasing, we believe that they will improve if the model is further tuned. For example, we can further preprocess the dataset to remove the noise that we mentioned in the limitations section. We can use a better loss function, increase the number of epochs and dataset size, which can be thought of as our future work. We also want to experiment with using the KerasTuner which is available at https://keras.io/keras_tuner/ to see how the results differ. In the future, we would also like to train on a larger dataset with more detailed captions, which would be more useful for designers with very specific prompts.

6. References

1. Bui, B. Fine-tuning stable diffusion using Keras. URL: https://github.com/Elvenson/stable-diffusion-keras-ft/blob/main/README.md, accessed on 28 Nov 2023.
2. Chollet,F., Wood,L. and Gupta, D. High-performance image generation using Stable Diffusion in KerasCV, URL: https://keras.io/guides/keras_cv/generate_images_with_stable_diffusion/, accessed on 28 Nov 2023.
3. Cui,Y.R., Liu,Q., Gao,C.Y., and Su, Z. FashionGAN: Display your fashion design using Conditional Generative Adversarial Nets. Computer Graphics Forum, Vol.37, No.7, pp.109-119, 2018.
4. Fashion Product Image Dataset. URL: https://www.kaggle.com/datasets/paramaggarwal/fashion-product-images-small, accessed on 28 Nov 2023.
5. Han,X., Wu,Z., Wu,Z., Yu,R., and Davis,L.S.. VITON: An Image-based Virtual Try-on Network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp.7543-7552, 2018.
6. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L. and Chen, W. Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685, 2021.
7. Jetchev,N., and Bergmann,U. The Conditional Analogy GAN: Swapping Fashion Articles on People Images. In Proceedings of the IEEE International Conference on Computer Vision, pp.2287-2292, 2017.
8. Paul, S. and Chansung Park, C. Fine-tuning Stable Diffusion, URL: https://keras.io/examples/generative/finetune_stable_diffusion/, accessed on 28 Nov 2023.
9. Rombach, R., Blattmann, A., Lorenz, D., Esser, P. and Ommer, B. High-resolution image synthesis with latent diffusion models. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 10684-10695, 2022.
10. Keras. 2022. Generate Images with Stable Diffusion. Keras Documentation. https://keras.io/guides/keras_cv/generate_images_with_stable_diffusion/
11. Sohl-Dickstein,J., Weiss,E., Maheswaranathan, N. and Ganguli, S.. Deep unsupervised learning using nonequilibrium thermodynamics. CoRR, abs/1503.03585, 2015.
12. Sun, Z., Zhou, Y., He, H. and Mok, P.Y. SGDiff: A Style Guided Diffusion Model for Fashion Synthesis. In Proceedings of the 31st ACM International Conference on Multimedia, pp. 8433-8442, Oct 2023.
13. Von Platen, P. URL: https://github.com/huggingface/diffusers/blob/main/examples/text_to_image/train_text_to_image.py. Accessed on 1 Dec 2023.