Clothes Design utilizing Generative AI

Final Year Project Final Report

Nguyen Quang Phuoc

Nguyen Ngoc Minh

Instructor Luong Trung Kien

Bachelor of Artificial Intelligence

Hoa Lac campus - FPT University

2023

Acknowledgment

We would like to thank Mr. Hanh, our advisor, and future Doctor Luong Trung Kien for providing us with computer hardwares and researching places. Their unwavering support and trust in our project have been invaluable. Additionally, we acknowledge Mr. Luong Trung Kien from FPT University for his continuous guidance and support throughout our final year project. His valuable feedback has helped improve our research quality. We also thank FPT University for giving us a good learning environment to study and develop well. Finally, we appreciate the support of our friends and families who encouraged us during the project journey. Their motivation helped us overcome challenges and complete this research work.

Abstract

The dynamic domain of text-to-image generation has witnessed remarkable advancements in recent years, particularly in applications like fashion design and artistic creation, where AI algorithms have demonstrated their capabilities. In addressing the intrinsic challenge of acquiring well-labeled datasets for training, we embraced a framework outlined in [7]. This framework hinges on the collaboration of two pivotal models: the Prior Model and the Decoder. Through careful curation of publicly available datasets, we meticulously prepared datasets for both pre-training and fine-tuning phases, with a deliberate choice of the MS-COCO dataset owing to its suitability and content integrity. Our efforts lay in the development of a comprehensive pipeline for text-to-image generation, complemented by a demo for deploying a model API. This demonstrated the potential of our approach in establishing a user-friendly service for designing Print On Demand images. The efforts not only showcase the current state-of-the-art in AI-driven image generation but also position our framework as a practical and accessible solution for users seeking personalized and creative image design experiences.

Keywords: Text-to-image, Shifted Diffusion, Clothes design, Print On Demand, Shirts

Table of Contents

List of Tables

List of Figures

1. INTRODUCTION

1.1 Problem and related work

Text-to-image generation is increasingly achieving good results and has been widely applied in recent years as research results and products in this field are becoming more and more popular and bring surprising results. In 2017, Amazon claimed that they "may be poised to lead the way when it comes to replacing stylists and designers with ever-so-chic AI algorithms" [1]. DeepVogue, a fashion design AI created by DeepBlue Technology, won the runner-up prize, and "People's Choice Award" at China International Fashion Design Innovation Competition 2019 [2]. AI is demonstrating its strong ability in creating art and becoming an effective tool in helping people, even with poor designing backgrounds, create their own artworks. The emergence of image generative models such as GANs [3] (e.g. CGAN [4], AttnGAN [5]), then diffusion models (e.g. DDPMs [6]) has achieved the ability to generate high-quality images that can comply according to the user's text conditions, but also very diverse.

We, together with advisor Luong Trung Kien and his university friend To Nguyen Hanh, had an idea about applying the image generation model to the application of designing Print On Demand (POD) drawings for clothes, especially when printed shirts have been one of the most attractive products for young people today. A system for designing POD drawings, like the AI models Midjourney and DALL·E, is a text-to-image generation problem.

For this project, manually labeling image-caption pairs is a very time-consuming task for a small team. Therefore we decided to approach this problem using the method mentioned in the article Shifted Diffusion for Text-to-image Generation with Language-free Text-to-image Generation setting [7]. The result demonstrated in the paper had shown that the paper's method has achieved better results compared to other pioneering text-to-image generation models.

This innovative approach leverages advanced image generation techniques to create tailor-made visuals for a diverse array of shirts, catering to the unique preferences of consumers. By adopting diffusion models, which excel in capturing intricate details and nuanced styles, this methodology seeks to revolutionize the POD industry by offering high-quality, customizable designs. In this context, we explore the integration of diffusion models, the challenges encountered, and the solutions devised to enhance the stability and quality of image outputs. Through this exploration, we aim to present a compelling narrative on the potential and advancements in utilizing diffusion models for the dynamic and ever-evolving landscape of POD shirt image generation.

Evaluating the quality and performance of text-to-image generation models is crucial in improving the text-to-image generation task results, common metrics and datasets play a vital role in this evaluation process. Metrics such as Inception score (IS) [8] and Frechet Inception Distance (FID) [9] are widely used metrics that assess

the quality and diversity of generated images. IS measures the diversity and confidence of class predictions within generated images, while FID quantifies the similarity between generated and real images in feature space or the distance between their feature vectors. In the case of IS, higher values are preferable, whereas for FID, lower values are preferred. R-precision [5], on the other hand, evaluates the alignment of generated images with textual descriptions by measuring how many real images are ranked higher than the generated ones based on their relevance to the text. For benchmarking, datasets like CUB (Caltech-UCSD Birds-200-2011) [10] and COCO (Microsoft Common Objects in Context) [11] provide rich resources, with CUB specializing in bird images and COCO offering a diverse range of everyday scenes and objects. These metrics and datasets collectively empower researchers to assess text-to-image generation models in terms of image quality, diversity, and textual alignment, facilitating advancements in this exciting and challenging domain.

After the publication of the Generative Adversarial Network (GAN) paper [3], GAN demonstrated superior image generation capabilities compared to previous methods, leading to a series of ongoing efforts to enhance their performance. GAN is a type of deep learning model consisting of two neural networks, a generator, and a discriminator, that are trained simultaneously through adversarial training. The generator aims to create data, such as images, that is indistinguishable from real data, while the discriminator's role is to differentiate between real and generated data. This competitive process helps the generator improve its ability to generate increasingly realistic and high-quality data, making GANs a powerful tool for tasks like image generation, style transfer, and data augmentation. There are multiple of GAN's variations, including the development of Conditional Generative Adversarial Networks (cGAN) [10], which allow users to impose specific conditions on generated images, laying the foundation for text-to-image models using GANs. On the other hand, Attentional Generative Adversarial Networks (AttnGAN) [6], building upon the progress in text-to-image generation, improved image generation accuracy by incorporating an attention mechanism to focus on word-level features. In 2018, the year of its publication, AttnGAN surpassed other GAN models, achieving new state-of-the-art results on both CUB and COCO datasets in terms of Inception Score (IS). Particularly noteworthy was its remarkable 170.25% increase in IS performance on the COCO dataset.

In 2020, when the paper Denoising Diffusion Probabilistic Models [6] by Ho et al was published, diffusion modeling gained attention and has since grown rapidly. Diffusion models represent a noteworthy advancement in the realm of generative modeling. Their approach, which involves iteratively refining data by adding noise, offers a fresh perspective on generating high-quality samples. This technique stands out due to its training stability, overcoming challenges often associated with traditional Generative Adversarial Networks (GANs), such as mode collapse. A study by OpenAI in 2021 [12] showed that Diffusion Models outperformed state-of-the-art GANS at that time like BigGAN-deep and StyleGAN2 on LSUN [13] and ImageNet [14] datasets in terms of FID with similar computing costs, and could generate images with a higher level of diversity. OpenAI's DALL·E 2 [15] and Google's Imagen [16] - two of the most large-scale text-conditional image generation models - have leveraged

diffusion models and can be generalized with the validation set of MS-COCO, respectively achieving state-of-the-art Zero-shot FID results and even surpassing many GAN-based models trained directly on the MS-COCO set. Research by 2 papers has shown that helping the diffusion model grasp the semantics of input conditional text by taking advantage of the ability to link the relationship between images and text using the latent space of the pre-trained CLIP model [17] (DALL·E 2) and the ability to effectively encode text of large language models like T5 [18] (Imagen), diffusion models can work effectively on text-to-image generation tasks by generating high quality and diverse images but still close to the given conditions.

However, having a training dataset for text-to-image generation tasks is not always an easy task, labeling image-text pair data in large quantities requires a lot of manpower and resources. Therefore, there have been studies on training models using datasets with few or only images. CLIP model is trained on a large set of image-text pairs, its ability to perform zero-shot text snippet prediction given an image has been utilized as the basis for recent Language-Free Training Text-to-Image Generation research. CLIP-GEN [19] embeds training images with CLIP to get their textual semantics and uses the embedded image as a conditional input alternative to descriptive text in methods that use paired text-image data. Combined with VQGAN that had been trained in the pre-training stage, the method aims to optimize Conditional Autoregressive Transformer, specifically in this case GPT2 architecture was used, to restore the low-level image information that the CLIP model failed to extract. The findings indicate that due to its extensive training on a vast image dataset, the model can produce high-quality images and attain commendable Inception Score (IS) and Frechet Inception Distance (FID) scores on the MS-COCO dataset though it still falls short of outperforming certain GAN-based models like AttnGAN in terms of IS. Despite being trained without textual information, the model demonstrated an understanding of the semantic concepts found in the validation set sentence descriptions. However, there were instances of failure, particularly in comprehending numeric concepts, such as the count of objects in the picture. Much like CLIP-GEN, the approach presented in LAFITE's method [20] utilizes CLIP's multimodal feature space for generating pseudo text features that approximate the actual text feature. The authors used StyleGAN2 [21], a state-of-the-art model in the field of GANs for image generation, as the generator and reformed it from unconditional to conditional generative model. The proposed method has been shown to be effective through extensive experiments. It achieved state-of-the-art results in standard text-to-image generation tasks, outperformed most existing models trained with full image-text pairs at the time the paper was published, and achieved better results in terms of IS and FID on MS-COCO dataset in language-free settings compared to the CLIP-GEN model. The latest version of LAFITE, Lafite2 [22], has improved the results on both MS-COCO and CUB datasets by introducing a new method of generating pseudo text features using a two-step process of retrieval and optimization. The relevant pseudo text features are first identified given an image, then these features are then optimized for better alignment. This method for creating synthetic text features can be combined with text-to-image models such as conditional StyleGAN2, similar to their previous LAFITE paper, or LDM model, and has become one of the pioneering methods for

language-free text-to-image generation. The final method we researched and selected to approach the problem is presented in the article [7]. Although the method mentioned in the article does not focus on improving text feature extraction from training images like previous studies, this method achieved the best overall results on the MS-COCO and CUB datasets compared to the three methods mentioned previously and the results are currently state-of-the-art for the language-free text-to-image generation task (Table 1).

Methods	**IS ↑**	**FID ↓**
CLIP-GEN [19]	21.40	20.70
Lafite [20]	27.20	18.04
Lafite-2 [22]	31.16	10.26
Corgi [7]	34.14	10.33

Table 1: Language-free text-to-image performance comparison

1.2 Dataset

1.2.1 Data overview

To train and fine-tune the models, we meticulously curate two distinct datasets. One comprises image-caption pairs, the other consists of images alone. We will use common and publicly available datasets for the text-to-image generation task for the image-caption pairs dataset. As for the image-only dataset, we will crawl POD images to help guide the model to generate images with this style. During the training process, the training image will be transformed using the CLIP's preprocess to align with text embedding. The transforming process consists of Resize the image to 224x224 pixels using BICUBIC interpolation, CenterCrop, Normalize and then convert the images to tensors.

1.2.2 Image-caption pairs data

For the first dataset, we have considered using several datasets that are publicly available to the community and also commonly used (Table 1). "k" represents the unit of "thousand", and "M" represents the unit of "million".

Dataset	Size (pixels)	Category	#train/val/test	Total
MS-COCO [11]	640x480	Humans and objects	118k/5k/0k	123k
CUB-200-2011[10]	500x500	Birds	6k/0/6k	12k
CC3M [23]	>400x400	All that pass the filters	3.3M/16k/13k	3.3M
LAION 400M [24]	256x256	All except NSFW		413M

Table 2: Review dataset for Text-to-Image task

After carefully considering the hardware conditions of the computer, we decided to choose the MS-COCO dataset. This dataset has image categories suitable for our training purposes, and also reduces exposure to NSFW content compared to two datasets CC3M and LAION 400M collected in large quantities from the web.

Figure 1: Example of MS-COCO image-text pairs

We use the train split of the MS-COCO 2017 as data for pre-training the Prior Model. The average image resolution of the dataset is 640×480 pixels. The dataset provides 118 thousands images with 5 different captions for each image to train an Image Captioning model, our purpose was to train a Text-to-Image one so we only need 1 caption for 1 image respectively.

We begin by downloading the dataset and initiating the preprocessing phase. Subsequently, we partition the processed text into six smaller segments. Each segment is associated with a metadata.jsonl file adhering to the format: {file_name, text}. This format effectively maps each image to its corresponding caption, aligning with the 'imagefolder' format used in HuggingFace datasets.

1.2.3 Image-only Data

The image-only dataset sourced from Amazon comprises meticulously captured representations of designed shirts, skillfully clipped to isolate the garment's visual features. The dataset exhibits a diversity of image formats, predominantly consisting of PNG, JPG, and PSD file types. Notably, the images showcase a variance in dimensions, with an average width ranging from 2800 to 6500 pixels and an average length spanning from 3000 to 7000 pixels. This dataset's composition reflects a comprehensive collection of visual assets, encapsulating the intricate details and design nuances of shirts, thereby providing a robust foundation for endeavors in computer vision and image processing.

Figure 2: Example of Amazon Clothes Design images

Despite assistance from Mr. Hanh, we encountered challenges in crawling a substantial amount of data. The Amazon.com website detected an abnormal number of GET requests, leading to the collection of only approximately 200 images. We attempted to utilize these images as a dataset for fine-tuning our Decoder. To prevent overfitting and preserve the model's pretrained knowledge [25], we employed a small number of training steps and a low learning rate. Unfortunately, this approach yielded negligible effects, as the fine-tuning process had minimal impact on the overall performance of the model. Therefore we will use the public Decoder checkpoint of the authors to do the evaluation.

1.3 Objective

Our goal for this project will be to evaluate the framework mentioned in [7]'s ability to generate images. We will evaluate whether the model is suitable for POD image generation, and from there will find a way to apply it to create products.

1.4 Scope

In this project, we will fine-tune the model and possibly retrain it if necessary. This practice helps us better understand how hyperparameters affect the model's training process and helps us develop the ability to optimize hardware within our budget capabilities.

Along with that, instead of just applying what is already available on GitHub, we will refine and design a text-to-image pipeline that meets both Mr. Hanh's POD images generation requirements and the hardware constraints of 12GB VRAM GPU and 32GB RAM that Mr. Hanh provided at the office.

Then, to demonstrate the high applicability of this pipeline, we will deploy the model's API and combine it with a User Interface (UI) to demo a shirt design website that we plan to develop in the future.

2. LITERATURE REVIEW OF METHODOLOGY

Utilizing the framework outlined in [7, Fig. 2c], the architecture will comprise two primary models: the Prior Model (Shifted Diffusion Model) responsible for producing image embeddings based on the text captions provided by users, and the Decoder (Stable Diffusion 2) which use the image embeddings to generate output images for the user.

2.1 Prior Model (Shifted Diffusion)

Prior Model is a decoder-only Transformer used to predict CLIP image embedding corresponding to the user's text input. As mentioned in [7], the model involved utilizing a sequence of multiple inputs, including encoded text, CLIP text embedding, an embedding representing the diffusion timestep, an embedding representing the index of corresponding Gaussian, and a noised image embedding. Instead of predicting the noise, the model is trained to predict the unnoised image embedding directly

The reason this model is called Shifted Diffusion is because of comparison with the sampling process mentioned in DALL·E 2 [15] which the authors call "vanilla sampling process", the authors present a sampling process that is said to better approximate CLIP image embeddings. As illustrated in [7, Fig. 3], the goal of Shifted Diffusion is to shift the starting point of the sampling process into the effective output space of the CLIP image encoder. The diffusion process normally uses a standard Gaussian noise distribution N(0, I), the starting point of the sampling process will be random noise, which may cause it to be at a far location from the target embedding. Therefore, the author changed the starting point of the sampling process by changing the noise distribution from standard Gaussian to N(μ, Σ), with μ and Σ being the mean and standard deviation of the ground-truth images in the training set respectively. Hence, the altered diffusion process will convert a ground-truth image embedding into a random image embedding rather than a random Gaussian noise. This modification assists in bringing the initial point of the image-denoising process closer to the target embeddings, consequently able to reduce the number of steps of the sampling process to approximate the target embeddings. The forward diffusion process moves from:

$$q(\mathbf{z}_t \mid \mathbf{z}_{t-1}) := \mathcal{N}(\mathbf{z}_t; \sqrt{1 - \beta t}\ \mathbf{z}_{t-1}, \beta_t \mathbf{I})$$

to:

$$q(\mathbf{z}_t \mid \mathbf{z}_{t-1}) := \mathcal{N}(\mathbf{z}_t; \sqrt{1 - \beta t}\ \mathbf{z}_{t-1} + \mathbf{s}_t, \beta_t \Sigma)$$

where $\mathbf{z}_t$ is image embedding at timestep t for t = T, ..., 1, β is the variance of Gaussian noise that controls the step sizes, and $q(\cdot|\cdot)$ is the forward diffusion process. A new term $\mathbf{s}_t$ is added at every timestep t indicates the shift term, and is chosen to be equal to $(1 - \sqrt{1 - \beta})\mu$. The reason for this choice is so that $q(z_T \mid z_0)$ can approximate the distribution of the image embeddings $N(\mathbf{z}_t;\mu,\Sigma)$. Mathematical proofs have been mentioned in the article [7].

2.2 Decoder Model (Stable Diffusion 2)

The methodology behind Stable Diffusion, a groundbreaking generative AI model for photorealistic image synthesis, involves a unique approach to image generation, differentiating it from conventional models. Launched in 2022, Stable Diffusion operates as a diffusion model, employing Gaussian noise to encode an image and subsequently using a noise predictor and reverse diffusion process to recreate the original image. Notably, it deviates from traditional pixel space utilization,

opting for a reduced-definition latent space to significantly reduce processing requirements while maintaining image quality.

Key architectural components include a variational autoencoder for compression and decompression, forward diffusion to add noise, reverse diffusion to undo the noise, a noise predictor (U-Net), and text conditioning. The variational autoencoder compresses images into a smaller, more manageable latent space, and forward diffusion progressively adds noise. Reverse diffusion undoes this process, enabling the generation of diverse and unique images.

The noise predictor, employing a U-Net model, is crucial for denoising images by estimating and subtracting noise from the latent space. Text conditioning, a common form of conditioning, involves using CLIP tokenizer and a text transformer to embed textual prompts into the model. This conditioning, combined with a random seed, allows for the generation of distinct images within the latent space.

3. IMPLEMENT

3.1 Implementation

- Data processing:

The MS COCO 2017 train split dataset undergoes a systematic treatment to facilitate effective utilization in training the Prior Model. The processed text is meticulously partitioned into six smaller segments, and for each segment, metadata.jsonl files are created, adhering to a standardized format denoted as {file_name, text}. This segmentation process ensures a structured organization of textual data, aligning with the 'imagefolder' format commonly employed in HuggingFace datasets. Simultaneously, for the Amazon clothes design dataset, a distinct preprocessing approach is adopted. Leveraging CLIP's preprocess functionality, the images are transformed systematically. This transformation involves a sequence of operations, starting with resizing the images to a uniform dimension of 224x224 pixels using BICUBIC interpolation for optimal quality. Subsequent steps include applying a CenterCrop operation, normalizing the images to ensure consistent features, and ultimately converting the processed images into tensors. This rigorous preprocessing methodology ensures that both datasets are suitably formatted and prepared for subsequent stages of model training and evaluation.

- Model Training:

The Decoder checkpoints represent Stable Diffusion 2, fine-tuned for 10,000 to 30,000 steps on various datasets including MS-COCO [11], Localized Narratives [26], CelebA-HQ [27], and CUB [10]. The Prior Model checkpoints encompass both a small model with 16 decoder layers and a large model with 20 layers; however, the specific number of training steps for each model is unspecified.

To optimize cost while achieving effective results, we opt to continue the fine-tuning process using these checkpoints. This approach allows us to leverage the prior knowledge encoded in the checkpoints and further enhance the model's performance.

In selecting the Decoder checkpoint, we chose the one trained on the MS-COCO dataset as the broader range of images in MS-COCO is deemed more effective for the scope of clothing design. The checkpoint trained on Localized Narratives dataset is also trained on the MS-COCO split of the dataset, and since the Decoder is only trained on image, both of the MS-COCO dataset checkpoint and Localized Narratives dataset checkpoint are trained on the same image set.

Conversely, the Prior Model checkpoints are not directly applicable since they were trained with the text embeddings of T5-11B [18], the 11 billion parameters checkpoint of the T5 model developed by the Google Research Team. Due to our system limitations, we are unable to meet the requirements for the T5-11B model. Consequently, we find it necessary to retrain the Prior Model from the ground up, utilizing the Flan-T5-Large [28] model – the largest T5 model within our capacity. The Flan-T5-Large model has the approximately same number of parameters with the T5-Large (780M parameters to 770 parameters) while performing better than all older T5 versions, even outperforming the 11 billion T5 checkpoint [28, Tab. 5].

Following dataset preparation, the model is trained using a single Q RTX 8000 GPU rented from the Vast.ai[1] server, equipped with 45 GB VRAM. The training process is conducted on 118,286 examples from the MS-COCO 2017 train split. Given the lack of detailed hyperparameter information in the referenced paper [7] and constraints on hardware resources, we make adjustments to select hyperparameters such as "train_batch_size," "num_train_epochs," "gradient_accumulation_steps," and "t5_model" to optimize resource utilization. Other hyperparameters adhere to the default values provided by the authors.

To expedite model convergence and reduce computational costs during the inference phase, we halve the number of layers compared to the original model. Our final important hyperparameters are shown in Table 3. The training process concludes after 14.25 hours, encompassing 9250 optimization steps.

[1] https://vast.ai

Parameter	Value
train_batch_size	128
num_train_epochs	10
gradient_accumulation_steps	1
t5_model	google/flan-t5-large
model_layers	8

Table 3: Hyperparameters on training the Prior Model

3.2 Text-to-image generation pipeline

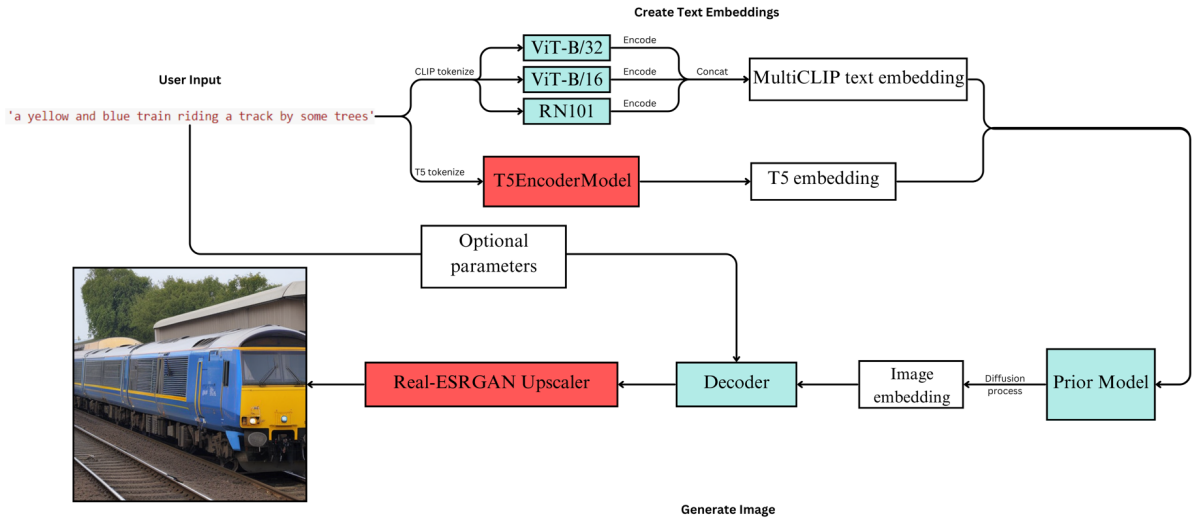

Figure 3: Text-to-image generation process

Figure 3 illustrates the process of our text-to-image generation, employing a pipeline akin to the StableDiffusionPipeline in the diffusers [29] library. Our approach begins by receiving a text input prompt from users. In addition to the input prompt, users can adjust several optional parameters to tailor the generated images to their preferences. These parameters include "guidance_scale," "num_inference_steps," "height," "width," and "negative_prompts."

The "height" and "width" parameters determine the size of the output image, compulsory to be divisible by 8, the default values for "height" and "width" are 512 and 512, and we also recommend using these values for optimal and stable results. "guidance_scale" enhances the connection between the input prompt and the output image, albeit with a trade-off in image quality. Increasing the "num_inference_steps" generally improves the quality of the output image, albeit at the cost of slower image generation speed. The "negative_prompts," a widely adopted feature in image generation prompts, assists in guiding what should be excluded from the image generation process, thereby enhancing image accuracy. For more detailed information, please refer to the diffusers GitHub repository.

Upon obtaining the user's input prompt, we create both the CLIP text embeddings and the T5 embeddings for the prompt. The code provided in paper [7] by the authors employs a method called MultiCLIP, utilizing three CLIP models for encoding text and images. To generate MultiCLIP embeddings, the input prompt is first tokenized and subsequently processed through three CLIP models: ViT-B/32, ViT-B/16, and RN101. All three CLIP models share a common base configuration, featuring a 12-layer 512-wide model with 8 attention heads for the Text Transformer. They also share identical image resolution (214) and embedding dimension (512). The outputs from these three models are concatenated to form the final MultiCLIP embeddings. Regarding T5 embeddings, the input prompt undergoes tokenization using T5's AutoTokenizer and is subsequently encoded by the T5EncoderModel. These embeddings are then input into the diffusion process of the Prior Model to derive the image embedding. The resulting image embedding is utilized in generating an image through a Stable Diffusion 2 Decoder, which has been trained to produce images based on their embeddings. Finally, the images will be used as the Real-ESRGAN [34] model’s input to go through the upscale process to make the photo resolution larger, thereby being able to be applied to many purposes, especially POD. The specific reasons for applying the Real-ESRGAN model will be presented in more detail in section 5.2.

Additionally, other parameters and the embedding for the negative prompt are integrated into the pipeline. This comprehensive set of inputs helps tailor the image generation process to meet the specific demands of the users.

Upon finalizing our pipeline, the next step involves deploying the model API for practical use by users. We employ the FastAPI [30] framework in Python and utilize the pyngrok [31] library to establish an ngrok tunnel, facilitating the deployment of the model for demonstration purposes.

3.3 Languages and libraries

The implementation described above primarily utilizes the Python programming language along with several libraries and frameworks that have been mentioned where they were first presented.

4. EXPERIMENTAL RESULTS

4.1 Environment

The training of both the Stable Diffusion and Shifted Diffusion models necessitates a minimum of 24GB VRAM. Our attempts to train on Kaggle and Google Colab were hindered by insufficient hardware capabilities. While Kaggle provides two T4 GPUs, almost meeting the requirements with slight VRAM deficiency of 10MB VRAM with train_batch_size = 4 and gradient_accumulation_steps = 4, the free GPU resources on both platforms were insufficient. Consequently, we opted to rent hardware from Vast.ai for enhanced training capabilities, utilizing a single Q RTX 8000 GPU with 45 GB VRAM.

For the image generation pipeline, the hardware requirements with all models loaded on GPUs amount to 7GB of RAM and 11 GPU VRAMs. To compute the FID score for evaluating model performance, a P100 GPU with 16GB VRAM and a CPU with 32GB RAM from Kaggle were employed for image generation and comparison. This strategic choice of hardware rental ensures efficient training and evaluation processes, overcoming limitations posed by free GPU resources on other platforms.

4.2 Result

Regarding evaluation, we will gauge the model's performance based on the FID score, utilizing the MS-COCO 2017 validation split. This split comprises 5,000 distinct images, each accompanied by its captions. Similar to the train split, the validation split offers 5 different captions for each image. To establish pairs for evaluation, we will retain the first caption found in the annotation file of the validation

split and pair it with its corresponding image. However, our system resources did not meet the requirement to calculate FID score on the whole 5,000 images of the validation split, therefore we randomly chose a different number of examples from the split to evaluate our dataset, and the maximum number of samples that the hardware can handle is 1000. The results are displayed in Table 4.

Number of samples	FID
50	280.66
200	197.04
500	149.70
1000	107.87

Table 4: FID results of our model on MS-COCO

While FID exhibits greater stability against noise compared to IS [9], as indicated in the article [32], it is essential to note that FID carries a substantial bias. The score finds it hard to converge to its believed true value, with different numbers of samples used to calculate, the score returns different values. As can be seen in Table 4, with the increasing number of samples used for calculation, the FID of the model also gradually decreases. It is therefore unfair to compare the FID we obtained with published state-of-the-art results because those results were calculated with 30 thousand samples.

Therefore, we will compare the FID results achieved with 2 models: stable-diffusion-2-base checkpoint is provided by Stability AI on huggingface, Lafite checkpoint is supervised training on MS-COCO set that achieves 8.12 FID when evaluated on 30000 samples on MS-COCO. We use both of these checkpoints to recalculate the FID results on the same 1000 samples of MS-COCO validation split, the results are demonstrated in Table 5.

Models	**FID_{1000} ↓**
Lafite [20]	141.40
stabilityai/stable-diffusion-2-base[2]	116.98
Corgi [7]	107.87

Table 5: FID results comparison on MS-COCO calculated on 1000 samples

Compare to the stable-diffusion-2-base checkpoint, we achieved slightly better results as the Decoder we are using is also utilizing the pre-trained knowledge of the stable-diffusion-2-base checkpoint, and also the Prior Model and the Decoder were respectively retrained and fine-tuned focus on the MS-COCO set, therefore improved the result achieved. And for the Lafite checkpoint, both of our training process and the checkpoint are mainly focus on using MS-COCO as the training data, although our training process has an advantage in terms of data quantity because we use MS-COCO's 2017 train split with 118k samples compare to 83k samples of 2014 train split used to train the Lafite checkpoint. However, it can be seen that at this time our model has shown a more robust performance on generating common objects compared to a checkpoint which had achieved 8.12 FID. Therefore, this framework has shown its capability of being applied to real world's problems, specifically POD image generation.

Because we used Stable Diffusion 2 to fine-tune into our Decoder with default setting config, the height, and width of the output image during the fine-tuning process were: unet_sample_size * vae_scale_factor = 512x512 (the Stable Diffusion 2's default values of unet_sample_size and vae_scale_factor are 64 and 8 respectively). Consequently, in text-to-image generation tasks, our model performs better in quality and stability when constructing 512x512 images. We generated images in some of the popular shirt's Print On Demand image sizes to compare them. The image size referenced is based on the article [33]. We will use the prompt: "a yellow and blue train riding a track by some trees" and the negative prompt: "no tree, wavy rail, low resolution" as inputs for this image generation test.

[2] https://huggingface.co/stabilityai/stable-diffusion-2-base

Figure 4: Sleeve and Pocket POD size image generation

When generating images for printing on Sleeves and Pockets, it is observed that photos with dimensions of 384x288, suitable for Adult Short Sleeve or Pocket, exhibit the highest quality. These images appear well-balanced, fully meeting the conditions specified in the input prompt. In the case of 288x192 images intended for printing on Youth Short Sleeve or Pocket, there is a noticeable decline in detail and quality. Finally, for photos intended for Long Sleeve printing, the images capture the specified details in the prompt and maintain good quality. However, they may start to incorporate redundant or illogical details.

Figure 5: Full Shirt Design POD size image generation

When generating images for Full Shirt Design, we have specified sizes for Adult Men's, Adult Women's, and Youth shirts as 1056x1200, 864x1056, and 864x1008, respectively. Nevertheless, with increasing resolution, the image begins to exhibit signs of deterioration. Objects start merging into one another, and the details gradually become less coherent. While the image maintains a certain level of quality, a significant portion of the details in the image loses logical consistency.

We can see that due to being trained to create images with a resolution of 512x512, a resolution that is too low will reduce image quality, and a resolution that is too high will cause the image to have irregular details.

To address this issue, our approach involves utilizing the Real-ESRGAN model, as presented in [34], to enhance the resolution of the generated image post the image generation process. After experimentation, we determined that the model achieves optimal performance with a resolution of 512x512. Consequently, we will employ this resolution during the image generation to ensure stability and quality in the outputs. The total number of pixels for this optimal result will be proportionally divided into height and width based on the user's original aspect ratio. This adjustment will be applied to both higher and lower resolutions. The newly derived height and width will serve as parameters for the text-to-image pipeline, replacing the previous ones. Subsequently, once the image is generated, we will resize it back to its original resolution.

In cases where the user's resolution exceeds ours, we will use the RealESRGAN_x4 model to upscale the generated image. By default, the model upscales the image four times compared to its original resolution. Therefore, we will first upscale the image and then resize it back to its original resolution. If the original image resolution’s height and width total pixels exceeds 4096 (the height and width’ sum of the resolution after upscaling from the optimal 512x512), the image will remain in the upscaled resolution post the upscaling step. We are not currently applying a larger resolution upscaler due to the possibility of anomalous details. If we have a GPU with larger resources, we can switch to using diffusion upscaler to increase the image quality after upscaling and also the stability of the output. On the other hand, if the user's resolution is lower than ours, we will straightforwardly resize the image back to its original resolution. Given that resizing the image to a lower resolution has a minimal impact on image quality, our solution ensures stable image quality for both high and low resolutions in our model pipeline.

Figure 6: Full Shirt Design POD size new image generation process

The aforementioned processing enhancements not only substantially improved the quality of the generated image but also accelerated the diffusion speed, given the reduced image resolution required for generation. Previously, generating an image with Full Shirt Design resolution on a Colab T4 GPU took approximately 1 minute; now, the process completes in less than 20 seconds. As depicted in Figure 6, the results indicate that, although the new images may still exhibit flaws, their quality has markedly improved compared to the previous versions and has become more stable. The reasons for inaccurate generated detail will be discussed at the end of this part.

To demo the application of text-to-image generation on POD design, we generate an image of sunset with the following parameters: prompt = ‘romantic sunset on the beach’, negative_prompt='low resolution', height=512, width=512, guidance_rescale=2.0, num_inference_steps=50. The shirt demonstration will be in Figure 7. This demo does not include all possible demos in shirt printing as we are limited in UI design. We refer to the UI from the repo Github [35].

Figure 7: Printed shirt demo

Lastly, we would like to discuss how Prior Model affects the overall image generation quality. We have tried some example prompts from the blog post [36] to evaluate our model. Due to the lack of training image style, our model is limited to generating normal life images, similar to the data from MS-COCO that was used to train the model, therefore we will test on common objects image generation of the model.

Figure 8: Comparing our model's out image (on the left) versus the sample images generated by Midjourney (on the right)

It can be seen in Figure 8 that for the first prompt "A cat on the sofa", our model resulted in an image that is precisely described by the text. Although the image quality is not the same as the sample ones, however both the model output image and the sample images can be best described using the same caption. Moving on to the second prompt, the model can only catch some ideas about the prompt: a child, beach. The "Nikon D6 high shutter speed action shot, soft yellow lighting" is a sign that the model is trained on a diverse set of data crawled from the internet. This part of text cannot be recognized by our model, therefore we will remove it and re-generate the image. The prompt becomes "A child playing on a sunny happy beach, their laughter as they build a simple sandcastle".

A child playing on a sunny happy beach, their laughter as they build a simple sandcastle

A laughing child building a simple sandcastle on a sunny happy beach

Figure 9: Model's output after changing the prompt

In Figure 9, after removing the redundant part of the prompt that the model is not able to understand, the model output has been more focusing on the child and the surrounding beach, the image just lacks "laughter" and "sandcastle". We then go further to summarize the prompt, which then helps the model constrain the output image to make the child actually build something. It can be inferred from this example that while the model is able to understand the ideas of the prompts, longer prompts have shown to become less effective than the shorter ones, and also distract the model from focusing on some details. There are 2 main reasons for the issue: 1. The MS-COCO only consists of short descriptions about the picture, which is less in detail compared with large web-crawling-based datasets such as LAION-5B, LAION-400M. These large datasets contain detailed descriptions of the image, therefore the models trained on these datasets are able to generate more diverse objects and styles. And 2. Our Prior Model was trained from scratch with nearly 10k steps, it cannot yet provide the same efficient textual embeddings as commercial AI text-to-image service. By further training the model with a more diverse training dataset, the performance of the model should improve significantly.

5. CONCLUSION

Our project showcases a pipeline that enables fine-tuning on an image-only dataset, alleviating the need for extensive image labeling while still delivering commendable results. By implementing the model architecture outlined in [7], individuals can fine-tune a model on their own dataset to attain a desired image style - in this instance, POD images. Moreover, the pipeline is designed for deployment and application with a UI, offering users a user-friendly service that caters to their specific demands.

Nevertheless, there exist limitations in the current models that could be enhanced in future work. In the context of a POD service, customers often have a significant demand for printing shirts with textual quotes, which remains a notable limitation for diffusion models. The model is trained to generate images that correspond to text descriptions rather than generating the text itself. Consequently, the challenge arises in producing "readable text" as part of the generated images. DALL·E 2, using the same architecture of Prior Model and Decoder, is also reported in [37] to struggle with text generation and even seemed to have developed a hidden vocabulary internally. To address this challenge, the use of combining Diffusion models with ControlNet [38] can help to preserve edges and lines condition, therefore maintaining the original wording requirements.

In this project, we've conducted a relatively brief training of the Prior Model, spanning approximately 10,000 steps—a modest iteration compared to other commercial text-to-image generation AI services. Although the results have not met our desire, they have shown that slight training can provide us with a pipeline Through additional fine-tuning of the model, we anticipate a significant enhancement in the quality of image generation.

In future works, our aim is to leverage Large Language Models (LLMs) for processing input prompts. The idea is inspired by the paper [39]. This includes tasks such as translation to facilitate easier use without the necessity of knowing English to compose a suitable prompt. Additionally, we plan to explore pre-processing tasks for input prompts, aiming to enhance the effectiveness of the image generation process. The LLMs will be fine-tuned to predict a detailed version of a prompt, specify elements that can improve the image quality. For example, a prompt “An old fisherman” can be processed into “Portrait of an old fisherman at sea, using natural light to highlight weathered textures.” (example prompt taken from the website https://mspoweruser.com/prompts-ai-art/). To do this a large dataset of effective prompts should be prepared, the LLMs will be used to summarize the effective prompts into its shorter version, and then the LLMs will be fine-tuned to do the reverse job: predict effective prompts from the brief one.

References

[1] "Amazon Has Developed an AI Fashion Designer | MIT Technology Review." Accessed: Sep. 30, 2023. [Online]. Available: https://www.technologyreview.com/2017/08/24/149518/amazon-has-developed-an-ai-fashion-designer/

[2] "AI designer takes applause on Shanghai catwalk - SHINE News." Accessed: Sep. 30, 2023. [Online]. Available: https://www.shine.cn/biz/tech/1904243580/

[3] I. Goodfellow *et al.*, "Generative Adversarial Networks," *Commun ACM*, vol. 63, no. 11, pp. 139–144, Jun. 2014, doi: 10.1145/3422622.

[4] M. Mirza and S. Osindero, "Conditional Generative Adversarial Nets," Nov. 2014, Accessed: Sep. 30, 2023. [Online]. Available: https://arxiv.org/abs/1411.1784v1

[5] T. Xu *et al.*, "AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1316–1324, Nov. 2017, doi: 10.1109/CVPR.2018.00143.

[6] J. Ho, A. Jain, and P. Abbeel, "Denoising Diffusion Probabilistic Models," *Adv Neural Inf Process Syst*, vol. 2020-December, Jun. 2020, Accessed: Oct. 03, 2023. [Online]. Available: https://arxiv.org/abs/2006.11239v2

[7] Y. Zhou, B. Liu, Y. Zhu, X. Yang, C. Chen, and J. Xu, "Shifted Diffusion for Text-to-image Generation," *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10157–10166, Jun. 2023, doi: 10.1109/CVPR52729.2023.00979.

[8] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved Techniques for Training GANs," *Adv Neural Inf Process Syst*, pp. 2234–2242, Jun. 2016, Accessed: Sep. 30, 2023. [Online]. Available: https://arxiv.org/abs/1606.03498v1

[9] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium," *Adv Neural Inf Process Syst*, vol. 2017-December, pp. 6627–6638, Jun. 2017, doi: 10.18034/ajase.v8i1.9.

[10] C. Wah, S. Branson, P. Welinder, P. Perona, and S. J. Belongie, "The Caltech-UCSD Birds-200-2011 Dataset," 2011.

[11] T. Y. Lin *et al.*, "Microsoft COCO: Common Objects in Context," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8693 LNCS, no. PART 5, pp. 740–755, May 2014, doi: 10.1007/978-3-319-10602-1_48.

[12] P. Dhariwal and A. Nichol, “Diffusion Models Beat GANs on Image Synthesis,” *Adv Neural Inf Process Syst*, vol. 11, pp. 8780–8794, May 2021, Accessed: Oct. 03, 2023. [Online]. Available: https://arxiv.org/abs/2105.05233v4

[13] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao, “LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop,” Jun. 2015, Accessed: Oct. 04, 2023. [Online]. Available: https://arxiv.org/abs/1506.03365v3

[14] O. Russakovsky *et al.*, “ImageNet Large Scale Visual Recognition Challenge,” *Int J Comput Vis*, vol. 115, no. 3, pp. 211–252, Sep. 2014, doi: 10.1007/s11263-015-0816-y.

[15] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, “Hierarchical Text-Conditional Image Generation with CLIP Latents,” Apr. 2022, Accessed: Oct. 04, 2023. [Online]. Available: https://arxiv.org/abs/2204.06125v1

[16] C. Saharia *et al.*, “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding,” *Adv Neural Inf Process Syst*, vol. 35, May 2022, Accessed: Oct. 04, 2023. [Online]. Available: https://arxiv.org/abs/2205.11487v1

[17] A. Radford *et al.*, “Learning Transferable Visual Models From Natural Language Supervision,” *Proc Mach Learn Res*, vol. 139, pp. 8748–8763, Feb. 2021, Accessed: Oct. 04, 2023. [Online]. Available: https://arxiv.org/abs/2103.00020v1

[18] C. Raffel *et al.*, “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer,” *Journal of Machine Learning Research*, vol. 21, pp. 1–67, Oct. 2019, Accessed: Oct. 04, 2023. [Online]. Available: https://arxiv.org/abs/1910.10683v4

[19] Z. Wang, W. Liu, Q. He, X. Wu, and Z. Yi, “CLIP-GEN: Language-Free Training of a Text-to-Image Generator with CLIP,” Mar. 2022, Accessed: Oct. 04, 2023. [Online]. Available: https://arxiv.org/abs/2203.00386v1

[20] Y. Zhou *et al.*, “LAFITE: Towards Language-Free Training for Text-to-Image Generation,” Nov. 2021, Accessed: Oct. 04, 2023. [Online]. Available: https://arxiv.org/abs/2111.13792v3

[21] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, “Analyzing and Improving the Image Quality of StyleGAN,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 8107–8116, Dec. 2019, doi: 10.1109/CVPR42600.2020.00813.

[22] Y. Zhou, C. Li, C. Chen, J. Gao, and J. Xu, “Lafite2: Few-shot Text-to-Image Generation,” Oct. 2022, Accessed: Oct. 04, 2023. [Online]. Available: https://arxiv.org/abs/2210.14124v1

[23] P. Sharma, N. Ding, S. Goodman, and R. Soricut, “Conceptual Captions: A Cleaned, Hypernymed, Image Alt-text Dataset For Automatic Image Captioning.” 2018. Accessed: Dec. 01, 2023. [Online]. Available: https://research.google/pubs/pub47380/

[24] C. Schuhmann *et al.*, “LAION-400M: Open Dataset of CLIP-Filtered 400 Million Image-Text Pairs,” Nov. 2021, Accessed: Dec. 01, 2023. [Online]. Available: https://arxiv.org/abs/2111.02114v1

[25] Y. Lu, S. Singhal, F. Strub, O. Pietquin, and A. Courville, “Countering Language Drift with Seeded Iterated Learning,” *37th International Conference on Machine Learning, ICML 2020*, vol. PartF168147-9, pp. 6393–6403, Mar. 2020, Accessed: Dec. 13, 2023. [Online]. Available: https://arxiv.org/abs/2003.12694v3

[26] J. Pont-Tuset, J. Uijlings, S. Changpinyo, R. Soricut, and V. Ferrari, “Connecting Vision and Language with Localized Narratives,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12350 LNCS, pp. 647–664, Dec. 2019, doi: 10.1007/978-3-030-58558-7_38.

[27] Z. Liu, P. Luo, X. Wang, and X. Tang, “Deep Learning Face Attributes in the Wild,” pp. 3730–3738, Nov. 2014, Accessed: Dec. 14, 2023. [Online]. Available: http://arxiv.org/abs/1411.7766

[28] H. W. Chung *et al.*, “Scaling Instruction-Finetuned Language Models,” Oct. 2022, Accessed: Dec. 12, 2023. [Online]. Available: https://arxiv.org/abs/2210.11416v5

[29] Patrick von Platen *et al.*, “Diffusers: State-of-the-art diffusion models,” GitHub repository. Accessed: Dec. 14, 2023. [Online]. Available: https://github.com/huggingface/diffusers

[30] Sebastián Ramírez, “fastapi: FastAPI framework, high performance, easy to learn, fast to code, ready for production,” GitHub repository. Accessed: Dec. 14, 2023. [Online]. Available: https://github.com/tiangolo/fastapi

[31] Alex Laird, “pyngrok: A Python wrapper for ngrok.,” GitHub repository. Accessed: Dec. 14, 2023. [Online]. Available: https://github.com/alexdlaird/pyngrok

[32] M. Binkowski, D. J. Sutherland, M. Arbel, and A. Gretton, “Demystifying MMD GANs,” *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, Jan. 2018, Accessed: Dec. 28, 2023. [Online]. Available: https://arxiv.org/abs/1801.01401v5

[33] Alex Clem, “T-Shirt Design Guide | PicMonkey.” Accessed: Dec. 13, 2023. [Online]. Available: https://www.picmonkey.com/blog/tshirt-design

[34] X. Wang, L. Xie, C. Dong, and Y. Shan, “Real-ESRGAN: Training Real-World Blind Super-Resolution with Pure Synthetic Data,” *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2021-October, pp. 1905–1914, Jul. 2021, doi: 10.1109/ICCVW54120.2021.00217.

[35] “GitHub - accodes21/tshirt3d: Custom 3D TShirt designer. Design your own T-shirts from scratch, customize its colors, logo etc. Made using ThreeJS, ReactJS and Custom Hooks.” Accessed: Dec. 14, 2023. [Online]. Available: https://github.com/accodes21/tshirt3d

[36] Miguel Rebelo, “How to write effective AI art prompts | Zapier.” Accessed: Dec. 14, 2023. [Online]. Available: https://zapier.com/blog/ai-art-prompts/

[37] G. Daras and A. G. Dimakis, “Discovering the Hidden Vocabulary of DALLE-2,” Jun. 2022, Accessed: Dec. 13, 2023. [Online]. Available: https://arxiv.org/abs/2206.00169v1

[38] L. Zhang, A. Rao, and M. Agrawala, “Adding Conditional Control to Text-to-Image Diffusion Models,” Feb. 2023, Accessed: Dec. 13, 2023. [Online]. Available: https://arxiv.org/abs/2302.05543v3

[39] Y. Hao, Z. Chi, L. Dong, and F. Wei, “Optimizing Prompts for Text-to-Image Generation,” Dec. 2022, Accessed: Dec. 13, 2023. [Online]. Available: https://arxiv.org/abs/2212.09611v1