Scene Creation and Understanding Using Diffusion Models - ECE 285

Vatsalya Chaubey
Electrical and Computer Engineering
PID: A59012865

Priyansh Bhatnagar
Electrical and Computer Engineering
PID : A59019463

Abstract

We intend to study stable diffusion models and evaluate them. Furthermore, to demonstrate a practical use of the model, we employ it for the task of image outpainting. We can use the diffusion model to generate an entire scene by making use of reference pixels generated in the original image. This can help us control minute details in the scene, as we would be able to specify what particular area of the image is to be modified.

1 Problem Definition

Diffusion Models have become quite popular in recent years for generative modelling. The problem we plan to tackle for achieving an in-depth understanding of generative models is image outpainting. Taking inspiration from some of the contemporary technologies like DALL.E and Getimg.ai, we intend to exploit Stable Diffusion models and generate possible visual representations of surrounding information/ context for already known data in the form of plain text. Image outpainting has a lot of real world applications nowadays like - content creation/generation for social media, data augmentation for tackling the problem of class imbalance, mitigating data losses while data transfers, building accurate image editing tools and many more.

Below is an example of the final representations that we want to achieve.

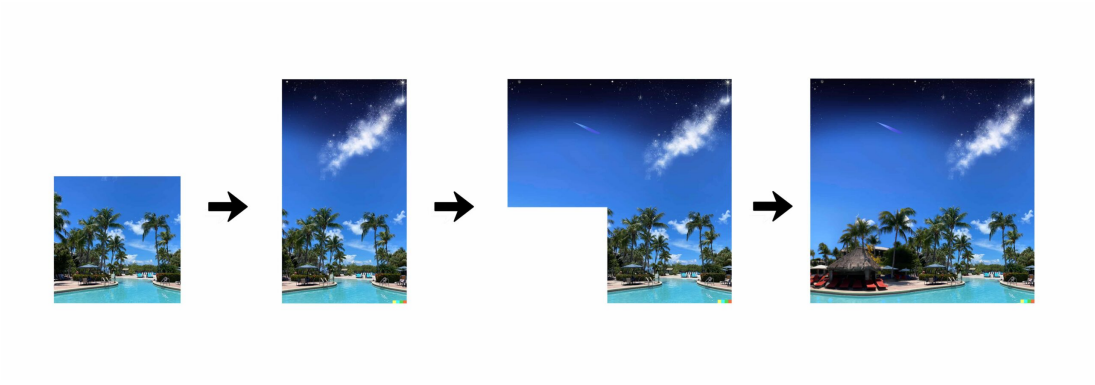

Figure 1: Expected results of Image Outpainting [8]

2 Related Works

Image outpainting has recently become one of the hot topics for text/ image to image generation. Earlier efforts were focused on solving this problem using generative models like GANs. In [14] the

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

author has used Generative Adversarial Networks to outpaint the surrounding pixels on all four sides. However, this extension is not guided by any image/ text prompt can not be customized. Moreover, every time the image has to be outpainted, a fixed portion can only be extended. Hence, we intend to address these two issues to generate customized (both in terms of content and size) outpainted images.

In a very recent paper[15] the authors are able to address these issues using Stable Diffusion to produce panoramic outpainted images that are coherent with each other.

Moreover, our biggest inspiration comes from OpenAI's DALL·E [16] who have just introduced image outpainting using Stable Diffusion.

In [17], the authors have tried to expand the aspect ratios of old content and use image outpainting using mutliple computer vision models including Stable Diffusion. They have adopted a novel method to build the masks, which inspired us to create masks in such a way, that without any other model or processing, Stable Diffusion can create coherent images.

3 Model Architecture and Methodology

Diffusion models try to estimate the data distribution $q(x)$ and generate samples from it to get new data. This is done as a two-step process:

3.1 Forward Diffusion

The forward diffusion process mimics a Markov Chain, where at each time step a random noise sampled from a Gaussian distribution is added to the data. This process is known as the "diffusion" of noise. The forward process tries to reach an isotropic normal distribution of data over a large number of timesteps T. At every timestep t the conditional distribution of the data given the data at the previous timestep can be modelled as a Gaussian as shown below. The step sizes are controlled by a variance schedule, $\{\beta_t \in (0,1)\}_{t=1}^{T}$. The variance parameter β_t can be a constant or can follow a linear or cosine schedule.

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

Over time as $t \to \infty$, the distribution $q(x_t)$ becomes an isotropic Gaussian.

We can further show that the distribution at any time step t can be written in the closed form by reparameterizing. This avoids computing the q sequentailly. Let $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$, then

$$\begin{aligned}
\mathbf{x}_t &= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1} && \text{;where } \boldsymbol{\epsilon}_{t-1}, \boldsymbol{\epsilon}_{t-2}, \cdots \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\bar{\boldsymbol{\epsilon}}_{t-2} && \text{;where } \bar{\boldsymbol{\epsilon}}_{t-2} \text{ merges two Gaussians (*).} \\
&= \dots \\
&= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}
\end{aligned}$$

This reparameterization allows us to write the distribution at any arbitrary timestep

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}) \tag{1}$$

We can then also write the joint distribution of all the timesteps as

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \tag{2}$$

This forward diffusion step in stable diffusion is pretty basic as for every image in our dataset, we add a random amount of noise over successive time steps and then use the Denoising UNet to predict the amount of noise added to the image. This process is shown below

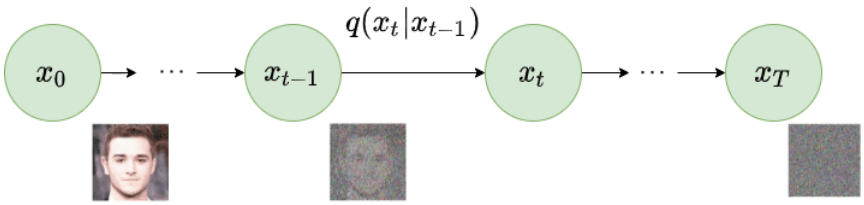

Figure 2: Forward Diffusion for one image [2]

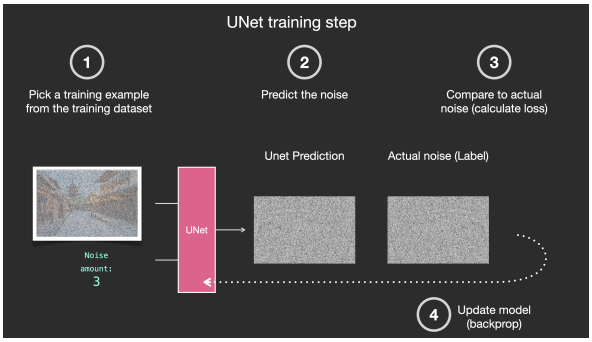

Figure 3: Training Loop for the Denoising UNet [13]

3.2 Denoising

Now, if we can learn the reverse distribution $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$, we can sample a data point $\mathbf{x}_T$ from $\mathcal{N}(0, \mathbf{I})$ and perform the steps in reverse to get a new sample from $q(\mathbf{x}_0)$. This iterative sampling process allows us to generate new data points from the original distribution. However, we don't accurately know the reverse distribution $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$. Thus, we learn a parameterized model p_θ using a neural network. The reverse distribution $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ will be Gaussian for a small enough β_t. Thus, we parameterize the mean and the standard deviation to learn the Gaussian

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \mathbf{\Sigma}_\theta(\mathbf{x}_t, t))$$

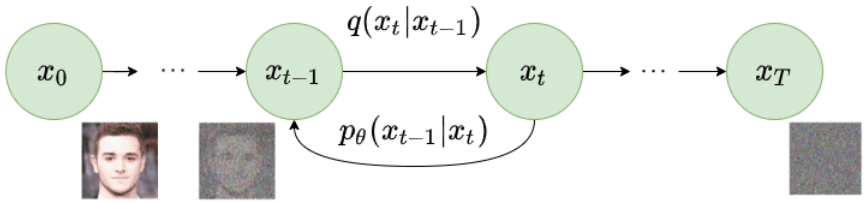

Figure 4: The Reverse Diffusion Process [2]

Thus, we can write the probability distribution for the entire trajectory as

$$p_\theta(\mathbf{x}_{0:T}) = p_\theta(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)) \tag{3}$$

As we can see Eq. 2 and Eq. 3 are essentially the same probability distributions and are reminiscent of the Variantional Autoencoders (VAE). Thus, to learn the parameters θ of the reverse diffusion model we can formulate a similar cost as that of VAEs and minimize it. The core idea is that we want to minimize the difference between the distributions q and p, and we use KL-Divergence as the distance metric between the distributions.

Thus, the loss function from [3] (as formulated for the case of VAE using ELBO) to be minimized can be written as

$$
\begin{aligned}
-\log p_\theta(\mathbf{x}_0) &\leq -\log p_\theta(\mathbf{x}_0) + D_{\mathrm{KL}}(q(\mathbf{x}_{1:T}|\mathbf{x}_0)\|p_\theta(\mathbf{x}_{1:T}|\mathbf{x}_0)) \\
&= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_{\mathbf{x}_{1:T}\sim q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})/p_\theta(\mathbf{x}_0)}\Big] \\
&= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_q\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} + \log p_\theta(\mathbf{x}_0)\Big] \\
&= \mathbb{E}_q\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\Big]
\end{aligned}
$$

$$
L = \mathbb{E}_{q(\mathbf{x}_{0:T})}\Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\Big] \geq -\mathbb{E}_{q(\mathbf{x}_0)} \log p_\theta(\mathbf{x}_0) \tag{4}
$$

It can also be shown from [2] that

$$
\tilde{\boldsymbol{\mu}}_t = \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_t\Big)
$$

We need to learn a neural network to approximate the probability distributions in the reverse diffusion process, $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$. We would like to train $\boldsymbol{\mu}_\theta$ to predict $\tilde{\boldsymbol{\mu}}_t$. Since $\tilde{\boldsymbol{\mu}}_t$ only depends on the noise $\boldsymbol{\epsilon}_t$, we can parameterize our model to predict the noise $\boldsymbol{\epsilon}_\theta(x_t, t)$.

As shown in [2], this reparameterization greatly simplflies our objective function

$$
\begin{aligned}
L_t^{\text{simple}} &= \mathbb{E}_{t\sim[1,T],\mathbf{x}_0,\boldsymbol{\epsilon}_t}\Big[\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\|^2\Big] \\
&= \mathbb{E}_{t\sim[1,T],\mathbf{x}_0,\boldsymbol{\epsilon}_t}\Big[\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t, t)\|^2\Big]
\end{aligned} \tag{5}
$$

The noise prediction model used is U-Net. U-Net is an architecture first created for image segmentation in medical imaging. In the U-Net architecture, the input and output are of the same shape. It consists of an encoder which downsamples the images followed by a decoder which upsamples the images to the original size. There are also skip connections between the encoder and the decoder to allow for faster training and alleviate the vanishing gradient problem. For stable diffusion, the U-Net uses the Resnet blocks for both downsampling and upsampling. At every time step the noisy image is fed to the neural network and the model tries to predict the noise.

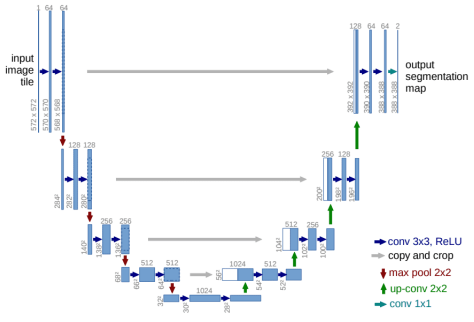

Figure 5: Basic U-Net as proposed in [9]. In stable diffusion each convolution block is replaced by a ResNet block.

In addition to having Resnet blocks the U-Net also contains cross-attention layers between the Resnet blocks. These layers look at any additional conditional information text, images, etc. and guide the reverse diffusion process by incorporating the conditional information in the latent space to generate specific types of images. Fig. 6 shows how the noisy image and text encodings are passed through

the denoising U-Net. Fig. 7 depicts the complete architecture of stable diffusion as described in [2]. We will talk about the other components in the next sections.

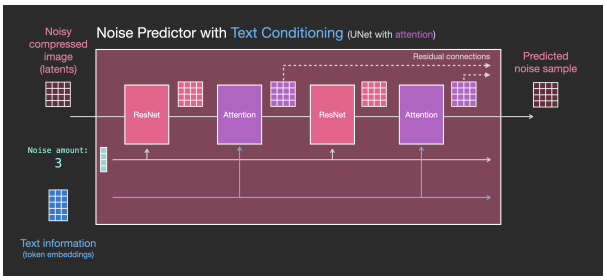

Figure 6: Denoising UNet with residual and attention blocks [13]

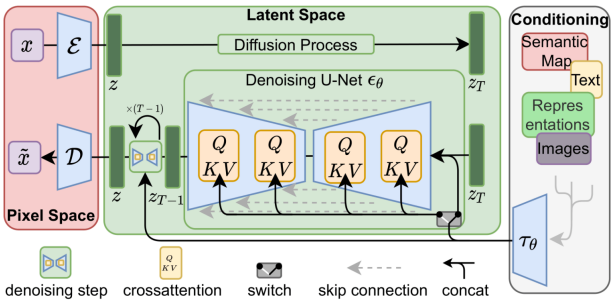

Figure 7: The Stable Diffusion Architecture. Each individual component is highlighted in a different color [5]

3.3 Latent Space

As we have seen in the previous sections under the working of the 'Diffusion' process, a random noise of a certain amount is added to the original image, and a new training example is created. This new training sample along with the amount of noise is then passed through the U-Net (which is nothing but our noise predictor). The U-Net is trained in a way such that the output of the U-Net is a noise that has minimal loss compared to the initial random noise added to the original image. This process is repeated 'N' times. 'N' is a parameter that used to be quite large when the model was just introduced. However, as time passed, newer and more efficient models take N to be 50 or 100. Even more recent models have limited this value to as low as N=4.

Something that is imperative and a bit obvious is that as this process of first adding noise (Forward Diffusion) to the original image and then denoising the image using U-Net (Reverse Diffusion) is not only time taking but also requires a huge amount of resources for the required computation.

Till now we have been working in the 'pixel space', that is, we are using operations directly on the original image with the same number of channels and dimensions (generally huge even for mid resolution images, let alone high resolution images, animations and videos). As the number of dimensions is large for the space that we working on, certainly the time taken and compute required is exponentially large.

To solve this problem, and make the whole process of Forward and Reverse Diffusion faster, the authors have moved/ shifted the whole training process into a new transformed space called the 'latent or hidden' space. This latent space has very few dimensions in comparison to the 'image/ pixel space'. Hence, the operations are much faster. The number of training steps can also be increased which we had to restrict earlier to save compute and time.

$$z^{d_l} = \epsilon(x^{d_i})$$

where ϵ is the encoder, z is the d_l dimensional latent representation of input x whose dimensions are d_i such that,

$$d_l << d_i$$

Now, we will dig in deep to see how this space is transformed into a low dimensional latent space. The idea is simple which is to pack the training setup by an encoder and a decoder (introduction of an Autoencoder) as shown below.

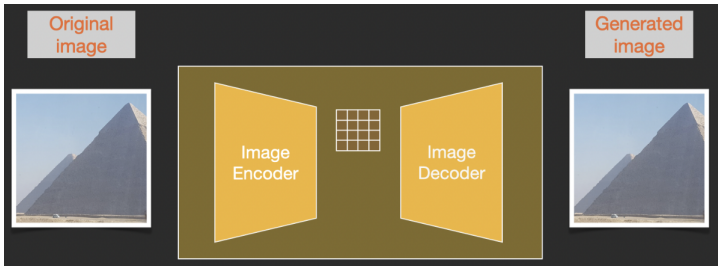

Figure 8: Introduction of Autoencoder [13]

Let us first discuss how this is achieved while training, and then we will discuss about how inference takes place.

While training, during forward diffusion, the original image is first passed through an encoder and is compressed to obtain a 'compressed latent representation' of the original image. Now the forward diffusion process is done on the compressed latents. The samples of noise are applied to those latents, not to the pixel image. And so the noise predictor (U-Net) is basically trained to predict noise in the compressed representation (the latent space) as shown below.

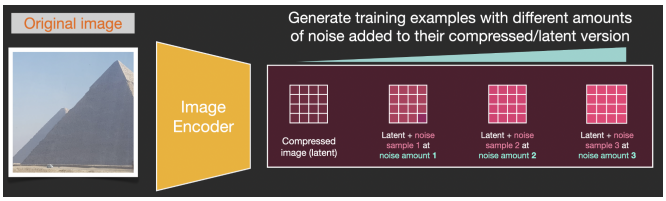

Figure 9: Forward Diffusion in Latent Space [13]

The forward process (using the autoencoder's encoder) is how we generate the data to train the noise predictor. Once it's trained, we can generate images of original dimensions by running the reverse process (using the autoencoder's decoder) using the latent space learnt as shown below.

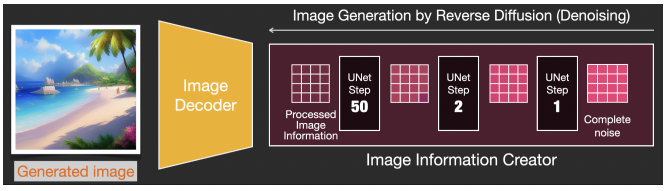

Figure 10: Reverse Diffusion in Latent Space [13]

3.4 Text Embeddings Generation

For the model to take text prompts as inputs along with images, a text encoder is used. This encoder builds a space where words can be represented as numbers and this representation of numbers is called a text embedding. Words which are related to each other or used together have higher similarity between their text embeddings.

The released Stable Diffusion model uses transformer based language model - ClipText (A GPT-based model), while the paper used BERT to generate these text embeddings. CLIP is composed of two encoders, one for processing images and the other for processing text.

To train CLIP, we can consider the process of taking an image and its corresponding caption, and encoding both of them separately using the image and text encoders. We won't be discussing the training process of CLIP in detail. However, we can give an overview of the whole process. The resulting embeddings after passing the images and text through encoders, are compared using cosine similarity.

$$\cos\theta = \frac{\overrightarrow{a} \cdot \overrightarrow{b}}{\|\overrightarrow{a}\|\|\overrightarrow{b}\|}$$

$$\cos\theta = \frac{\sum_i^n a_i b_i}{\sqrt{\sum_1^n a_i^2}\sqrt{\sum_1^n b_i^2}}$$

Here, $\overrightarrow{a}$ and $\overrightarrow{b}$ are the embedding vectors.

When the training process begins, the similarity is not very high, even if the text describes the image accurately. Then the two models are updated so that the next time we embed them, the resulting embeddings are similar. Below is the cosine similarity that is used.

This process is repeated for multiple images and captions and multiple datasets as well. Both 'positive' (embeddings for correct images and captions should have high similarity) and 'negative' (embeddings for incorrect images and captions should have low similarity) training is done in order to obtain better embeddings.

Below is a high level explanation of the training process for CLIP.

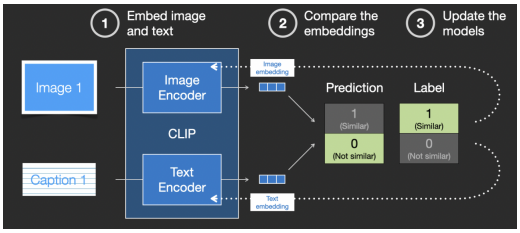

Figure 11: CLIP [13]

3.5 Text prompts + Image Generation

As we have already seen how images are generated with the help of the 'Diffusion process', we still have to include information from text prompts for the whole model to work with both images and text prompts. Using the CLIP Transformer Language Model, once we have generated the text embeddings, we use these as an additional input to our U-Net.

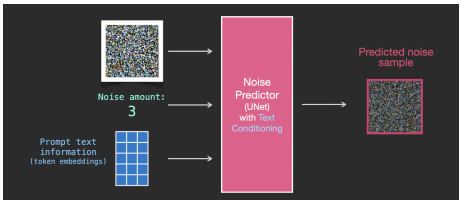

Figure 12: Text and Image (both as input) [13]

As shown below, once we have the compressed latents for the image, noise is added to the compressed image (forward diffusion) and then finally text embeddings, noisy image and noise label are fed into the model.

Figure 13: Noise added to Image (Text and image both as input) [13]

4 Experiments

To achieve our objective using Stable Diffusion, experiments were conducted in two parts

- Trained the text encoder to generate custom embeddings
- Outpainted generated images to create a user-instructed scene

4.1 Part I

To create customized scenes specific to an entity, it is required to train the model in such a way that for a given token, a set of desired visuals are learned through embeddings. For example, we have focused on creating a scene/ surrounding neighbourhood for a library (specifically Geisel Library, UC San Diego). Earlier, when text prompts like 'Geisel Library' or 'Library' were given, the following visuals were generated -

Figure 14: Generated of Images of Library (Without training)

As we can see in the above images, the generated photos are not specific to the 'Geisel Library'. In order to address this, we have trained the text encoder while keeping all the other components of 'Stable Diffusion' model frozen, so that it generates images of 'Geisel Library' when suitable prompts are given.

Dataset

We used a custom dataset comprising of 10 images. Some of them can be visualized as follows -

Figure 15: Dataset Visualization

Textual Inversion is a method used to extract new ideas from a limited set of example images. Although initially showcased with a latent diffusion model, this approach has now been extended to other model variations such as Stable Diffusion. The acquired concepts can enhance the control over

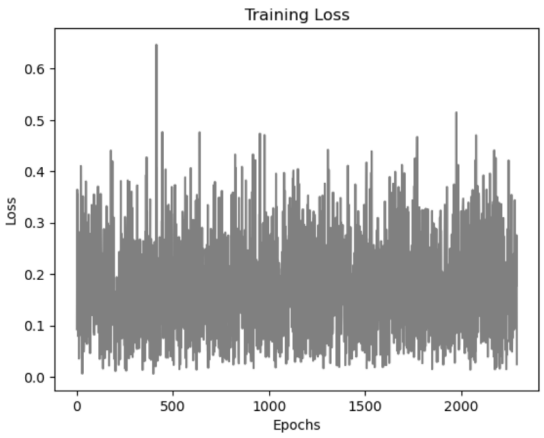

Figure 16: Training Loss

generated images in text-to-image systems. By acquiring new "words" within the embedding space of the text encoder, these tokens are incorporated into text prompts to personalize the generation of images.

For the training process, we created our own dataset of images of 'Geisel Library' along with corresponding text prompt templates like 'a photo of a ', 'a painting of ' etc. Once the dataset was created, the CLIP Tokenizer is loaded and the new token is added to it so that it is able to get associated with new features in the images provided. Finally the text encoder, U-Net and Variational Autoencoder are loaded. Since we are only training the newly added embedding, except the text encoder, the other two components are kept frozen. Also, the following hyperparameters were used for the training setup -

- Minimum noise variance: 0.00085
- Maximum noise variance: 0.012
- Noise schedule: scaled linearly
- Learning rate: 5e-04
- Max train steps: 2500
- Train batch size: 1
- Gradient accumulation steps: 4

The training loss is observed with every epoch and can be visualized in Fig. 16.

There are a lot of fluctuations in the loss which indicate the highly complex nature of our model and a small batch of inputs. Hence, there is significant variation in the values while training due to the existence of several local minima and the model is not able to converge to a single value. This nature of the loss helps us to generate new and fresh concepts around the ground truth.

4.2 Part II

Once we are able to generate customised images for Geisel, our next step is generate its surroundings guided by user given text prompts. Following steps are adopted to achieve the same -

- Resize the image to desired size
- Pad the image to achieve desired shape (height, width) of generated image (with mode as 'mean', 'edge' or 'linear ramp' for better results)
- Prepare a mask consisting of black and white pixels such that only white pixels are generated again and black pixels are retained
- Finally use the Inpaint pipleine from Stable Diffusion to generated the extended image
- Play with/fine tune the seed, number of inference steps, negative prompts and guidance scale to enhance the scene generated

5 Results

Finally, after adding the new token, and training the model, we could generate the images as shown in Fig. 16.

Figure 17: Generated Images for token - ‘Geisel Library’

In comparison to the images that we generated earlier without adding the new token, we are now able to mimic the architecture and design of Geisel.

Following is an example of a scene generated of size (500, 1100) from an initial image of (500,300).

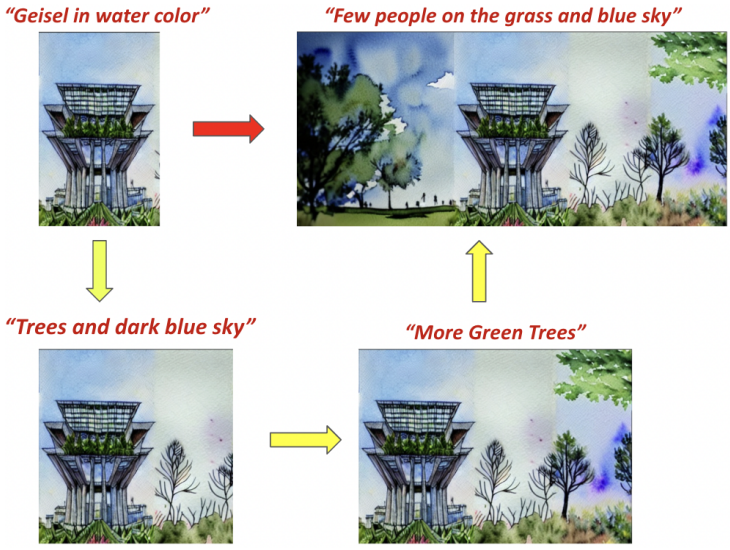

Figure 18: Scene Creation around ‘Geisel Library’

First, the previously learnt and newly added token ‘Geisel Library’ is used to generate images of Geisel guided by the text prompt ‘Geisel in water color’. Then the image is extended to (500,500) using the text prompt ‘Trees and dark blue sky’, followed by ‘More green trees’ to further extend the image on the right to (500,700). Finally, an extension of 400 pixels is guided by the text prompt ‘Few people on the grass and blue sky’ to achieve a scenery of size (500,1100).

Another example with a ‘Realistic’ background setting can be seen below.

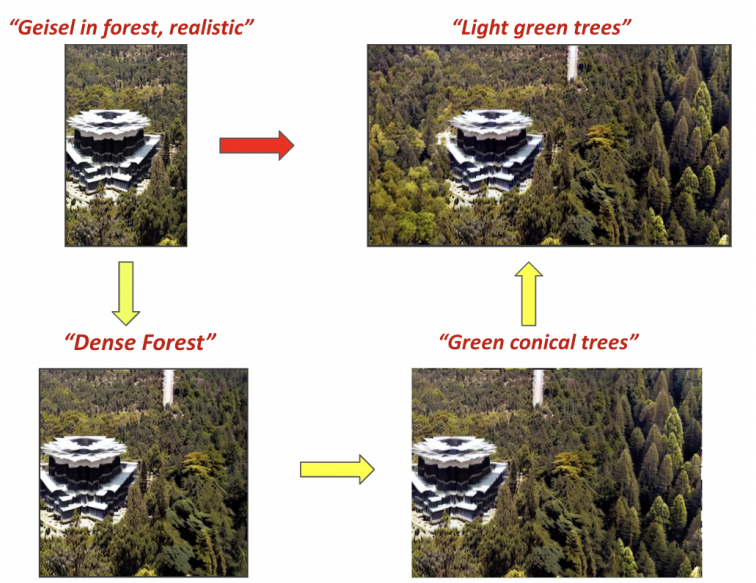

Figure 19: Scene Creation around 'Geisel Library' (Realistic)

6 Contributions

We modify vanilla Stable Diffusion to incorporate and understand specific images and vocabulary to build custom embeddings. This enables us to explore more personalized ideas and image generation. Secondly, we extend the modes of inputs which can be provided to Stable Diffusion to generate an image. Now, we provide multimodal inputs both in the form of text and images. Finally, Stable Diffusion has a hard time filling in a blank image (with all pixels at constant intensity) so we devise a novel to add noise or structure to the base image such that Stable Diffusion can fill in the details without losing the aesthetic sense of the image.

7 Conclusions

Stable diffusion has been the state of the art for text to image generation. It also allows for context to be incorporated into the generation process through various modalities like text, image, semantic map etc. In this work, we attempt to introduce multi-modal conditions for image generation in Stable Diffusion. Additionally, we are also fine tuning certain word-embeddings to generate images relevant to us, for eg. we would want to learn embeddings for the word "Geisel library" which would generate images related to UC San Diego. Finally, using both approaches in tandem we are able to generate a complete scene.

8 Code

The code for project is available at - `https://github.com/PriyanshBhatnagar/Scene-Creation-using-Image-Outpainting/tree/main`

References

[1] Sohl-Dickstein, Jascha, et al. Deep Unsupervised Learning Using Nonequilibrium Thermodynamics. arXiv:1503.03585, arXiv, 18 Nov. 2015

[2] Ho, Jonathan, et al. Denoising Diffusion Probabilistic Models. arXiv:2006.11239, arXiv, 16 Dec. 2020

[3] Weng, Lilian. What Are Diffusion Models? 11 July 2021

[4] Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., & Sutskever, I. (2021). Zero-Shot Text-to-Image Generation. ArXiv.

[5] Rombach, Robin, et al. High-Resolution Image Synthesis with Latent Diffusion Models. arXiv:2112.10752, arXiv, 13 Apr. 2022

[6] Karagiannakos, Sergios, Adaloglou, Nikolaos, "Diffusion models: toward state-of-the-art image generation", "https://theaisummer.com", 2022

[7] Justin. 'Design Your Masterpiece – Outpainting with DALLE-2'. January 17, 2023. https://goldpenguin.org/blog/design-your-masterpiece-outpainting-with-dalle-2/

[8] Steins. 'Stable Diffusion Clearly Explained!'. January 2, 2023. https://medium.com/@steinsfu/stable-diffusion-clearly-explained-ed008044e07e

[9] Olaf Ronneberger and Philipp Fischer and Thomas Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation", 2015

[10] Nichol, Alex, and Prafulla Dhariwal. Improved Denoising Diffusion Probabilistic Models. arXiv:2102.09672, arXiv, 18 Feb. 2021

[11] Dhariwal, Prafulla, and Alex Nichol. Diffusion Models Beat GANs on Image Synthesis. arXiv:2105.05233, arXiv, 1 June 2021

[12] Karagiannakos, Sergios, Adaloglou, Nikolaos, "Diffusion models: toward state-of-the-art image generation", https://theaisummer.com/, 2022

[13] Alammar, J, "The Illustrated Stable Diffusion", https://jalammar.github.io/illustrated-stable-diffusion/

[14] Van Hoorick B. Image outpainting and harmonization using generative adversarial networks. arXiv preprint arXiv:1912.10960. 2019 Dec 23.

[15] Li J, Bansal M. PanoGen: Text-Conditioned Panoramic Environment Generation for Vision-and-Language Navigation. arXiv preprint arXiv:2305.19195. 2023 May 30.

[16] OpenAI, 'DALL·E', https://openai.com/blog/dall-e-introducing-outpainting

[17] Paull N, Keshari S, Wong Y. Stable Remaster: Bridging the Gap Between Old Content and New Displays. arXiv preprint arXiv:2306.06803. 2023 Jun 11.