In-Depth Analysis of Stable Diffusion Model
CS 7180: Advanced Perception

Prateek Gulati[1]
Northeastern University
gulati.p@northeastern.edu

Darshil Patel[2]
Northeastern University
patel.darshi@northeastern.edu

Vikram Bharadwaj[3]
Northeastern University
bharadwaj.vi@northeastern.edu

Abstract

Image synthesis is the process of artificially generating images that contain specifically desired attributes. There have been many models created for this purpose with roots in likelihood-based models, such as Generative Adversarial Networks (GANs) [5]. However, they have proven to be fairly limited in variability and difficult to scale. Diffusion models (DMs) [2] are relatively new models that have achieved state-of-the-art results in image synthesis. Stable diffusion model [11] is able to achieve competitive scores in various tasks while significantly reducing computational requirements. However, because using DMs for image synthesis is a recent study, there is very limited study on their interpretation.

In this project, we study and analyze DMs. We evaluate Stable diffusion model on DrawBench [12]. We navigate through the text embedding space using interpolation and arithmetic operations and investigate the generated images . To control the layout, structure and artifacts in an image, we use cross attention map injection [6]. This allows local and global modification in the scene by replacing a word or controlling the extent to which a word is reflected in the image. This research project a step in the direction of dissecting, analyzing and visualizing Stable diffusion models, with the goal of understanding the mechanisms underlying relations as well as concepts. Through this project, we hope to provide a systematic analysis to understand the internal representations of DM.

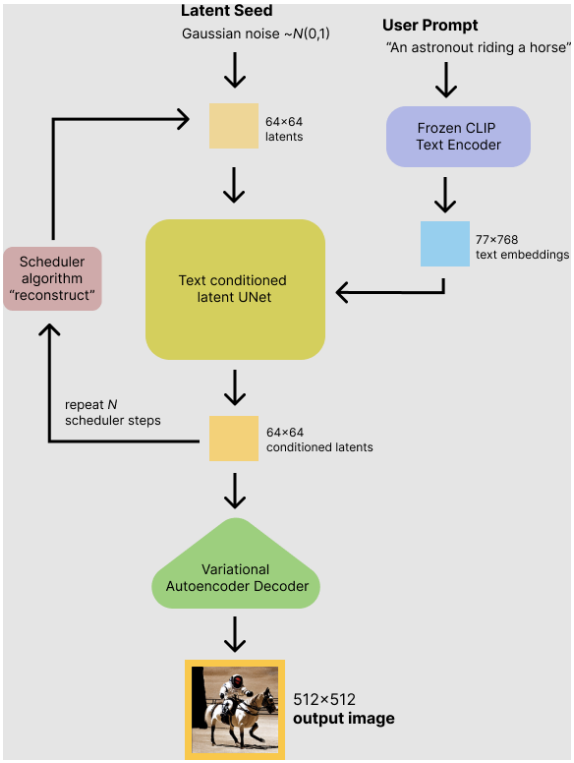


Figure 1. The stable diffusion model takes both a latent seed and a text prompt as an input. The latent seed is then used to generate random latent image representations of size 64×64 where as the text prompt is transformed to text embeddings of size 77×768 via CLIP's [10] text encoder

1. Introduction

Image synthesis has been one of the fields in Computer Vision that has witnessed spectacular development in recent times. GANs [5] played an important role in this expansion by allowing efficient sampling of high-resolution images with good perceptual quality. However, their results have been revealed to be confined to data with comparably limited variability. The encoding and decoding of input im-

ages produce semantically similar output images in a GAN.

Introduced in Dickstein *et al*. [3] work, Diffusion Models have become the talk of the hour. Large scale DMs such as OpenAI DALL-E 2, Google Imagen, Craiyon, and Stable Diffusion are able to create realistic images corresponding to novel concepts and compositions unseen in the training data. Its process of sequentially denoising autoencoders achieves extremely high scores while being extremely versatile in terms of application. Unconditional DMs are even able to perform tasks such as inpainting and colorization or stroke-based synthesis.

Architecturally, the diffusion models are U-Nets that incorporate cross-attention from text, but how this simple architecture is capable of creating such high-quality output is not well understood. Words such as numbers, relations, and positions influence the output is not understood.

The aim of the project is to develop methods that can use these insights to control these models in new ways. This paper shows how to identify new interpretable control directions for existing Diffusion Models, without requiring supervision or expensive optimization. Using these methods we can navigate the image synthesis process in a direction of our choice by controlling the local artefacts and global semantics of the scene. We try to create our exploratory approach that will make it easy to browse through the concepts that the DM has learned.

2. Related Works

2.1. Generative Models for Image Synthesis

The high dimensional nature of images presents distinct challenges to generative modeling. Ian Generative Adversarial Networks (GAN) [5] allow for efficient sampling of high resolution images with good perceptual quality, but are difficult to optimize and struggle to capture the full data distribution. In contrast, likelihood-based methods emphasize good density estimation which renders optimization more well-behaved. Variational autoencoders (VAE) and flow-based models enable efficient synthesis of high resolution images, but sample quality is not on par with GANs. While autoregressive models (ARM) achieve strong performance in density estimation, computationally demanding architectures and a sequential sampling process limit them to low resolution images. Because pixel based representations of images contain barely perceptible, high-frequency details [16,73], maximum-likelihood training spends a disproportionate amount of capacity on modeling them, resulting in long training times. To scale to higher resolutions, several two-stage approaches use ARMs to model a compressed latent image space instead of raw pixels

2.2. Diffusion Probabilistic Models

Recently, Diffusion Probabilistic Models (DM) [3], have achieved state-of-the-art results in density estimation as well as in sample quality. The generative power of these models stems from a natural fit to the inductive biases of image-like data when their underlying neural backbone is implemented as a UNet [9]. The best synthesis quality is usually achieved when a re-weighted objective is used for training. In this case, the DM corresponds to a lossy compressor and allow to trade image quality for compression capabilities. Evaluating and optimizing these models in pixel space, however, has the downside of low inference speed and very high training costs. While the former can be partially addressed by advanced sampling strategies and hierarchical approaches, training on high-resolution image data always requires to calculate expensive gradients. The Latent diffusion Models address both drawbacks, which work on a compressed latent space of lower dimensionality. This renders training computationally cheaper and speeds up inference with almost no reduction in synthesis quality.

These models can be interpreted as an equally weighted sequence of denoising autoencoders $\epsilon_\theta(x_t, t)$; $t = 1...T$, which are trained to predict a denoised variant of their input x_t, where x_t is a noisy version of the input x. The corresponding objective can be simplified to

$$L_{DM} = \mathbb{E}_{x,\epsilon\,\mathcal{N}(0,1),t}\left[||\epsilon - \epsilon_\theta(x_t, t)||_2^2\right] \tag{1}$$

2.3. Stable Diffusion

The version that we use here in this paper, Stable Diffusion is built on Rombach *et al*. [11] prior work Latent High-Resolution Image Synthesis with Latent Diffusion Models. Thanks to Stability AI and LAION, CompVis were able to train a Latent Diffusion Model on 512x512 images from a subset of the LAION-5B database. The v1 that we use, refers to a specific configuration of the model architecture that uses a downsampling-factor 8 autoencoder with an 860M UNet and CLIP ViT-L/14 text encoder for the diffusion model. The model was pretrained on 256x256 images and then finetuned on 512x512 images. This is a latent diffusion model conditioned on the (non-pooled) text embeddings of a CLIP ViT-L/14 text encoder.

3. Methods

3.1. Latent Space Interpolation

Interpolation is used to traverse between two known locations in latent space. Research on generative models often uses latent-space interpolation as a way of demonstrating that a generative model has not simply memorized the training examples. It is also used to qualitatively examine how well the model generalizes. It forms a trajectory between two latent encodings z_1 and z_2, which are indepen-

dently sampled from the prior distribution. Following the notations from Lesinak *et al*. [8], I formulate interpolation algorithms as a function f:

$$f : \mathbb{R}^D \times \mathbb{R}^D \times [0, 1] \ni (z_1, z_2, \lambda) \mapsto z \in \mathbb{R}^D \quad (2)$$

where λ is referred to as the interpolation weight.

Linear Interpolation This is most widely used interpolation algorithm, which simply forms a straight line between $z1$ and $z2$

$$f^{linear}(z_1, z_2, \lambda) = (1 - \lambda)z_1 + \lambda z_2 \quad (3)$$

Frequently used, this is often inappropriate as the latent spaces of most generative models are high dimensional with a Gaussian or uniform prior. In such a space, linear interpolation traverses locations that are extremely unlikely given the prior.

Spherical Interpolation Based on the formula introduced by Shoemake *et al*. [13], it treats the interpolation as a great circle path on a D-dimensional hypersphere.

$$f^{slerp}(z_1, z_2, \lambda) = \frac{\sin[(1 - \lambda)\Omega]}{\sin(\Omega)} z_1 + \frac{\sin[\lambda\Omega]}{\sin(\Omega)} z_2 \quad (4)$$

where Ω is the angle between z_1 and z_2. This method treats the interpolation as a great circle path on an n-dimensional hypersphere. Prior work by White *et al*. [15] using this technique has shown promising results on generative models like VAE and GAN with both uniform and Gaussian priors.

Normalised interpolation From Agustsson *et al*. [1], this methods adapts the latent-space interpolation operations, so that the resulting interpolated points would match the prior distribution. Based on linear interpolation with Gaussian prior, the formulation can be derived as

$$f^{norm}(z_1, z_2, \lambda) = \frac{(1 - \lambda)z_1 + \lambda z_2}{\sqrt{(1 - \lambda)^2 + \lambda^2}} \quad (5)$$

3.2. Latent Embedding Arithmetic Operations

Multiple prior studies have demonstrated an unexpected side effect of deriving word embeddings via neural networks, that is these embeddings appear to have additive composition. The typical example for this analogy popularised by word2vec is king $-$ man $+$ woman $=$ queen. In this section we try to explore this interest in using algebraic operations on word embeddings from clip to carry out semantic operations.

Feature subtraction Given two latent embedding z_1 and z_2, such that $z2$

$$f^{subtract}(z_1, z_2, \lambda) = z_1 - \lambda z_2 \quad (6)$$

Intuitively, embedding z_2 should be a subset of z_1, for example, if z_1 is *a scene of forest*, then z_2 can be *a tree*. If such a criteria is not adhered to, the resulting embedding would not match the prior distribution semantically and it will be hard to see any impact of $f^{subtract}$ in the generated image.

Feature substitution Another way we can utilize additive nature of latent space for editing the image is by substituting a feature. In the subtraction Eq. (6) of one feature vector from another, we can simultaneously add a third feature in the same ratio.

$$f^{substitute}(z_1, z_2, z_3\lambda) = z_1 - \lambda z_2 + \lambda z_3 \quad (7)$$

The same ratio λ ensures that the modified embedding stays close to the prior distribution and avoids divergence. From the previous example, z_1 is *a scene of forest*, z_2 is *a tree*, we can add a z_3 as *river*.

3.3. Navigating the cross attention

The cross attention maps define the spatial layout and geometry of a generated image using a text conditioned diffusion model. Each pixel in the generated image $\mathcal{I}$ gives more *attention* to the corresponding word. These attention maps $\mathcal{M}$ generated from original prompt $\mathcal{P}$ can also be used for a modified prompt $\mathcal{P}^*$. The synthesis of new image $\mathcal{I}^*$ will not only be generated using the new prompt $\mathcal{P}^*$, but also retain the structure and layout of original image $\mathcal{I}$. In this paper, we discuss two methods proposed in Hertz *et al*. [6] work and build on top of it.

3.3.1 Word swap

The goal here is to swap word(s) from the original prompt and generate a new image with a good alignment to the new prompt while preserving the original layout. To address this, the attention map from the original prompt can be used to condition the generation of the modified prompt. However, this injected attention map can over-constrain the layout, especially when there is a need of large structural change e.g. "a tree" to "a house". As proposed in Hertz *et al*. [6], we use a softer attention mechanism

$$Edit(\mathcal{M}_t, \mathcal{M}_t^*, t) := \begin{cases} \mathcal{M}_t^* & \text{if } t < \tau, \\ \mathcal{M}_t & \text{otherwise} \end{cases} \quad (8)$$

where t is the timestep until the injection is applied. By restricting the number of injection steps, the composition can be guided while allowing the freedom for adapting to the new prompt.

3.3.2 Attention re-weighting

Another aspect for modifying the prompt can be to strengthen or weaken the extent to which a token affects the resulting image. For example, a prompt that says "A

(a) Linear interpolation between text prompt *Desert* and *River*

(b) Linear interpolation between text prompts *Mountain* and *Grass*

(c) Spherical interpolation between text prompt *Ocean* and *Desert*

Figure 2. Comparision of transformed embeddings obtained from interpolation on two specific text prompts from Clip. In Fig. 2a, images 2-4 display beach sand and 5-6 have reluctance over the sand creating a water effect. In Fig. 2b, we can see a gradual appearance of grass on rock mountains. In Fig. 2c, the water dries up gradually with the same background before transforming into a desert.

blossomed tree", in this we want to change the intensity of the 'blossomness'.

$$Edit(\mathcal{M}_t, \mathcal{M}_t^*, t)_{i,j} := \begin{cases} c.(\mathcal{M}_t)_{i,j} & \text{if } j = j^*, \\ (M_t)_{i,j} & \text{otherwise} \end{cases} \quad (9)$$

where j^* is the token for which the attention is scaled.

3.4. Observing Style with Pixel-Level Attribution Maps

The work of [14] introduced DAAMs (Diffusion Attentive Attribution Maps) which are pixel-level attribution maps for visualizing the strength of association between words in prompts and the pixels generated by diffusion modelS. In diffusion models, attention mechanisms cross-contextualize text embeddings with coordinate-aware latent representations for an image. This latent space mapping gives us attention scores for each token image patch pair. These attention scores are then aggregated in the spatiotemporal dimension using interpolation. The final output is a heatmap that depicts the influence of a word over the pixels in the generated image. We extend this method to observe the attribution map when introducing a new style to a diffusion model. This gives us a way of evaluating the results of the newly trained style along with insights into how the conception of the style evolves during training.

To teach a new style to the network we apply the textual inversion method of [4]. Using 3-5 training images of a style/object this method learns to map a token to the visual characteristics of the training data. The model is trained on prompts containing a token representing the new concept. This enables the manipulation of the text embedding space to map the token to the image generation. The text encoder for the diffusion model is optimized to inject this new association into the model. The UNet and Variational Auto-Encoder (VAE) parameters are not modified in this process. We simply modify the text encoder to learn the new concept associated with a token while conditioning the image generation on the 3-5 images that capture the new concept.

4. Experiments and Results

4.1. Stable Diffusion on Drawbench

As introduced in Saharia *et al*. [12], DrawBench, is a structured suite of text prompts for text-to-image evaluation. It attempts to enable deeper insights through the evaluation of text-to-image models, with text prompts. These prompts are designed to probe different semantic properties of models. Some of these categories include compositionality, cardinality, spatial relations, complex text prompts, rare words. The goal is to push the limits of models' ability to generate highly implausible scenes well beyond the scope of the training data.

The model performs well in most of the easy-medium categories like color, DALL-E, descriptions, conflicting, etc., the results are mediocre on harder categories like text,

(a) Prompt: *A green valley*, the word *green* is swapped with L-R i) original, ii) dry, iii) wet, iv) flower, v) rocky, vi) snow, vii) desert

(b) Prompt: *A wall painting depicting a lake*, the word *wall* is swapped with L-R i) original ii) glass, iii) oil, iv) watercolor, v) pastel, vi) crayon, vii) children, viii) desert

Figure 3. Scene preservation using cross attention injection: By swapping a word of the initial prompt, we can make global modifications to the scene while preserving all the artifacts.

(a) Prompt: *Sunset in a mountain valley with a river* where the attention weight of the word *river* is increased from L - R

(b) Prompt: *Snowfall in a street with trees* where the attention weight of the word *trees* is increased from L - R

Figure 4. Text based editing by attention re-weighting: By changing the attention of a word of the initial prompt, we make local modifications to the scene while preserving the scene.

rare-words, and misspellings. Some of these are depicted in Fig. 5. A plausible reasoning for that can be the use of CLIP over a large transformer like T5-XXL in Imagen [12]. CLIP is trained on text-image pairings, so it might not have a good understanding of misspelled words, text intensive prompts and rare occurring words. As the prompts deviate from real-life instances, the image fidelity reduces. In addition, numerous prompts in DrawBench are fairly arduous even for an average human to understand. As these prompts deviate from real-life to abstract instances, it becomes hard to evaluate them, since it relies on an individual's perception. There is no quantitative measure to evaluate such a benchmark, this becomes subjective to human raters.

4.2. Interpolate in Text Embedding Space

Contrastive models like CLIP have been shown to learn robust representations of images that capture both semantics and style. the joint embedding space of CLIP enables language-guided image manipulations in a zero-shot fashion. The generative stack in Stable Diffusion uses conditioning from captions using a prior $P(z|y)$ that produces CLIP embedding z conditioned on captions y.

I compare different interpolation algorithms for two text embedding (z_1, z_2) from CLIP models. The embedding are extracted for two generic classes like *Grassland* and *Mountain* which is interpolated to obtain a new embedding z'. The transformed embedding z' is used to condition the diffusion model. In Fig. 2 we can see a gradual translation from one class to the other while preserving the image fidelity producing natural images. In Fig. 2b, initially the grass appears only on those parts of the mountain which gets more sunlight and then it spreads gradually.

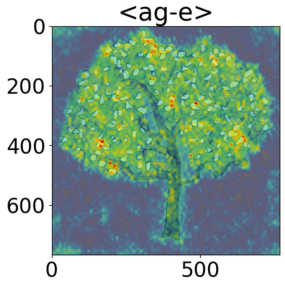
We see that transitions from linear and spherical interpolation are very similar to each other. The images generated using spherical interpolation of text embeddings are slightly more detailed, but in most of the tests, they are very close to

(a) Category - Colors

(b) Category - Conflicting

Figure 5. Stable Diffusion model evaluated on DrawBench introduced in [12]. Since the results are not cherry-picked, it seems to perform well in both the categories depicted here with a good text-image alignment. The image fidelity is dropped in the conflicting category.

(a) Training Images for Textual Inversion



(b) DAAM for style with prompt: "a tree in the style of ..."

Figure 6. Training data and results from adding a new style to a Stable Diffusion model. We can see the heatmap generated by querying the style in the generated image,

each other. A plausible explanation for this can be - if you select two samples, $z1$ and $z2$ from a Gaussian distribution in a high-dimensional space, they will have a similar Euclidean norm, and they will be roughly orthogonal. Theoretically, linear interpolation results in 'fainter' interpolants, but for the text conditioning, this difference is insignificant in our results. On the other hand, normalized interpolation alters the text embeddings such that they are not in the same bounds as training data. This leads to very unrealistic results. A comparison of three methods can be seen in Fig. 9. For most of our experiments, we use linear interpolation, since it's computationally cheap and produces good results.

4.3. Arithmetic Operations on Text Embeddings

We experiment with different prompts to test the effect of feature subtraction Eq. (6). We see that as the value of $\lambda \rightarrow 1$, the generated image diverges to produce noise. This is likely due to the train-test data mismatch arising from higher values of λ. At each sampling step, the text embeddings must be within the same bounds as training data. We see that high values of λ cause the text embedding values to exceed these bounds. To counter this problem, we make a minor change to out Eq. (6).

$$f^{subtract}(z_1, z_2, \lambda) = (1 + \lambda)z_1 - \lambda z_2 \tag{10}$$

By adding an additional term of $\lambda \times z_1$, we are able to restrict the values somewhat within the bounds.

Next, we experiment with feature substitute Eq. (7). The values of λ in Eq. (7) are set between the range of $[0, 1]$. As we see in Fig. 10, the substitution results seem to have a higher alignment to the text. The results are good when we attempt to substitute the color or texture. But as value of $\lambda \rightarrow 1$, the fidelity drops slightly, some odd artefacts appear in the image. A general consensus for these arithmetic operations is that they need to be restricted within their original bounds. That is a challenge we notice in our studies.

4.4. Cross Attention Guidance

In our experiments till now, we do see some interesting results, most of them have a high alignment and fidelity.

One common observation is that these attempt to edit the images are not confined to local artefacts. Most of these changes are global which doesn't retain the semantic layout and structure of the original image.

Using cross attention guidance Fig. 12, we demonstrate a more controlled editing by modifying the prompts. This method allows retaining the spatial layout, geometry, and semantics when replacing the word in the prompt. In the initial setup we swap a given word from an original prompt using the interaction between the pixels to text embedding using cross attention layers.

Global modification: We can preserve the composition of a scene and the artifacts while doing a global editing. The goal is to retain the location and identify of each artifact while changing the global environment. We can see in Fig. 3a, we are able to change the terrain of the valley while preserving the structure of valley and the hills. In Fig. 3b, we try to change the style of painting while keeping the contents same.

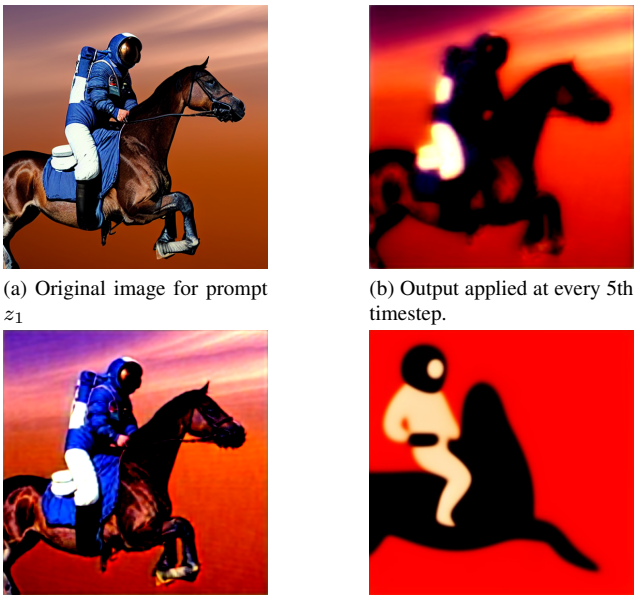
Local modification: In some situations, we want to preserve the global scene but modify some specific artefacts in the image. We utilize cross attention maps corresponding to that object. In this method, we can alter the intensity of the effect induced by a specific word. In Fig. 4a, we change the attention weight of the word river and we can see the water flow in the river increases from left to right. Similarly, in Fig. 4b, we increase the attention weight of the word 'trees' and see that it replaces trees with buildings as we lower its attention-weight. More results can be seen in Fig. 11.

4.5. Adding Style Using Textual Inversion and Observing with DAAMs

In our experiments, we worked with the pre-trained Stable Diffusion 2 model provided by StabilityAI. We attempted to add the style of artist Alex Grey to the model. Using the textual inversion method of [4] we trained the model on 3 images to teach the new concept. Imagegen prompts with the new style's token were used to teach the text encoder. We trained the model for 1000 training iterations, a learning rate of 1e06, and used an Adam optimizer. Every 50 iterations we observe the DAAM heatmap looking for the pixel attributions for the new style. We can see in the figure some examples of the output of the model at different iterations in training. The style is encoded to a satisfying degree and we check the output of a prompt, "a tree in the style of ..." to evaluate the new style. We see in figure 6 that the level of influence for the style is evenly spread across the generated pixels, showing how the style is encoded by the diffusion model.

4.6. Observing Style with Scheduler-Level changes

The work of [7], brought out an idea of a Linear Multistep Scheduler for discrete beta schedules. Based on the original k-diffusion implementation, a scheduler has been adopted to the Stable Diffusion by [11]. The k-diffusion scheduler takes in a timestep to predict the updated version of the sample being diffused. The timesteps dictate where in the diffusion process the step is, where data is generated by iterating forward in time and inference is executed by propagating backwards through timesteps.



(a) Original image for prompt z_1

(b) Output applied at every 5th timestep.

(c) Scheduler used to compute noisy at every 5th timestep.

(d) Output applied at every 50th timestep.

Figure 7. Observing style with scheduler-level changes.

$$x_{i+1} \leftarrow x_i + (x_{i+1} - \hat{x}_i)\left(\frac{1}{2}d_i + \frac{1}{2}d_i\right) \tag{11}$$

Four sets of experiments were carried out on the scheduler level and each are reported in Figure 7. ***a)*** When the output of the scheduler was applied to the latent input of the model at every timestep, which is the best way to run a diffusion model, and the results were sharp and as expected ***b)*** The output was applied for every 5^{th} timestep, we could see that there were significant loss of information in the image. ***c)*** The scheduler is used to compute the noisy sample at every 5^{th} timestep. The result here indicates that the noise ***d)*** The scheduler is used to compute the noisy sample at every 50^{th} timestep. The prompt used at all times was: ***An astronaut riding a horse.***, and the results for these are presented in Figure 7.

4.7. Observing Style with Prompt changes during de-noising

Another way in which we analyzed the DM [11] was based on the way in which the outputs changed when the prompts changed during the de-noising step using the K-diffusion scheduler. The work of [7], brought out an idea

of a Linear Multistep Scheduler for discrete beta schedules. Based on the original k-diffusion implementation, a scheduler has been adopted to the Stable Diffusion by [11]. This would mean that whatever the model has learnt till that particular step with a particular prompt has to be maintained and propagated through the next steps with a different prompt. The first prompt is a sentence which is converted into a CLIP [10] embedding z_1, of size 77x768, which are then interpolated with the text conditioned latent UNet outputs at every step as directed by the scheduler. Since we are running a total of 100 steps, we use z_1 for the first 50 steps and z_2, that are obtained from the second prompt for the rest of the 50 iterations. There were 3 prompts with which we experimented and they include:
a) $prompt_1$ = "A bear climbing a tree" $prompt_2$ = "A lion climbing a tree" ***b)***$prompt_1$ = "An astronaut riding a horse" $prompt_2$ = "An astronaut riding a zebra" ***c)***$prompt_1$ = "A dog playing with a ball" $prompt_2$ = "A cat playing with a ball".

5. Discussion and Summary

In this work, we have demonstrated image editing capabilities using different components of diffusion models. We primarily focus on text prompts for a navigating through the image generation process. Initially, we use naive techniques like interpolation and basic arithmetic operations on text-embedding to condition the diffusion process. Despite some good results, these methods either a) cause a train-test mismatch by exceeding the bound of embedding values or b) not able to retain the semantic layout and structure of the scene. To address this, we investigate the cross attention maps [6] and their key role in connecting each word in the text prompt to structure of the image scene. By using these maps, we are able to alter specific aspects in the scene while preserving the local artefacts or global scene. Using DAAMs to visualize the training process for stable diffusion and observing the influence of text on image pixels also shed light on these models. This is another step in giving more explainability to diffusion models and applying these techniques to get better end results. We have also explored the inner workings of the scheduler and de-noiser that allows us to understand how the DM interprets prompts and changes the output space based on the conditioned input.

This work is a first step in direction of understanding and interpreting the underlying concepts in diffusion models.

6. Acknowledgement

We want to thank Dr. Bruce Maxwell for his valuable inputs and guidance that helped us improve our work. We also want to thank Computer Vision and Learning LMU Munich for providing the pre-trained models of Stable Diffusion and bloc97 for providing a template code of Hertz *et al*. work [6].

Although, this was a collaborative effort to reach a common goal, our individual work can be described as - a) Prateek - Abstract, Introduction, Related Works (Generative Models, Diffusion Models, Stable Diffusion), Methods (Latent Space Interpolation, Latent Embedding Arithmetic Operations, Navigating the cross attention), Experiments (Stable Diffusion on Drawbench, Interpolate in Text Embedding Space, Arithmetic Operations on Text Embeddings, Cross Attention Guidance); b)Vikram - Experiments(Observing Style with Prompt changes during de-noising and Observing Style with Scheduler-Level changes); and c) Darshil - Methods (Observation of Style Encoding Using Pixel-Level Attribution Maps), Experiments (Teaching a New Style)

(a) The latent vector conditioned as a bear during the first 50 iterations of $prompt_1$

(b) The latent vector conditioned as a lion during the first 50 iterations of $prompt_2$

(c) The latent vector conditioned only on $prompt_1$ during de-noising in the first 50 iterations.

(d) The latent vector conditioned on $prompt_2$ for the next 50 iterations.

(e) The latent vector conditioned only on $prompt_1$ during de-noising in the first 50 iterations, (looks somewhat like a dog)

(f) the latent vector conditioned on $prompt_2$ for the next 50 iterations, (looks like a cat with dog ears)

Figure 8. Observing style with prompt changes during de-noising.

References

[1] Eirikur Agustsson, Alexander Sage, Radu Timofte, and Luc Van Gool. Optimal transport maps for distribution preserving operations on latent spaces of generative models. *CoRR*, abs/1711.01970, 2017. 3

[2] Jeff Donahue Andrew Brock and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *ICLR*, 2019. 1

[3] Jascha Sohl Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *CoRR*, abs/1503.03585, 2015. 2

[4] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H. Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion. 2022. 4, 7

[5] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. 2014. 1, 2

[6] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022. 1, 3, 8

[7] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models, 2022. 7

[8] Damian Lesniak, Igor Sieradzki, and Igor T. Podolak. Distribution-interpolation trade off in generative models. 2018. 3

[9] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. *CoRR*, abs/2102.09672, 2021. 2

[10] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. 1, 8

[11] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *CoRR*, abs/2112.10752, 2022. 1, 2, 7, 8

[12] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding, 2022. 1, 4, 5, 6

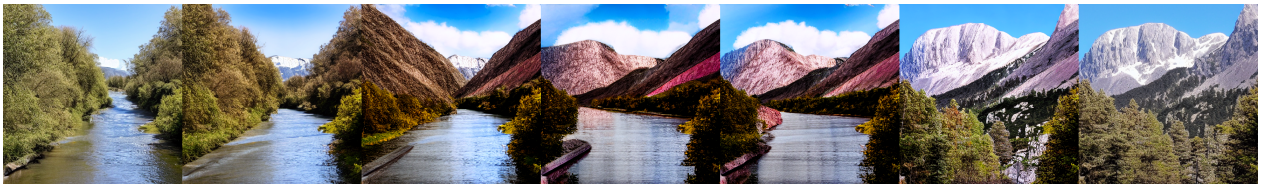
[13] Ken Shoemake. Animating rotation with quaternion curves. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '85, page 245–254. Association for Computing Machinery, 1985. 3

[14] Raphael Tang, Linqing Liu, Akshat Pandey, Zhiying Jiang, Gefei Yang, Karun Kumar, Pontus Stenetorp, Jimmy Lin, and Ferhan Ture. What the daam: Interpreting stable diffusion using cross attention. *CoRR*, 2022. 4

[15] Tom White. Sampling generative networks: Notes on a few effective techniques. *CoRR*, abs/1609.04468, 2016. 3

(a) Linear interpolation through prompts: *A river* and *Mountains* creating a transition from L - R

(b) Spherical linear interpolation through prompts: *A river* and *Mountains* creating a transition from L - R



(c) Normalized linear interpolation through prompts: *A river* and *Mountains* creating a transition from L - R

Figure 9. Comparison of linear, spherical and normalized interpolation methods on the same prompts

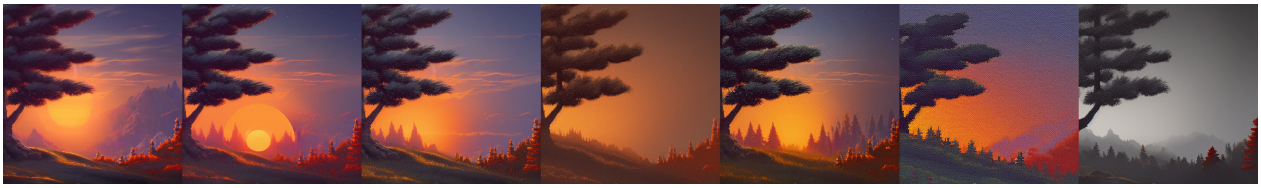
(a) Feature subtraction: *River - Trees*

(b) Feature subtraction: *Winter - Trees*

(c) Feature substitution: *Snow street scene - trees + buildings*

(d) Feature substitution: *A red sunset sky - red + blue*

Figure 10. A transition of arithmetic operations Eq. (10) and Eq. (7) for different text prompts where λ is increased from $L \rightarrow R$.



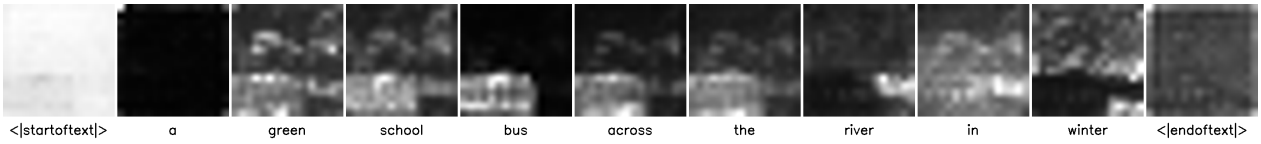
(a) Prompt: *A fantasy landscape with a pine tree in the foreground and a red sun setting in the distance, trending on artstation*, the word *fantasy* is swapped with L-R i) original, ii) cinematic, iii) realistic, iv) sepia, v) bokeh, vi) pixelated, vii) greyscale

(b) Prompt: *A green valley*, the word *green* is swapped with L-R i) original, ii) dry, iii) wet, iv) desert, v) snow, vi) volcano, vii) pink

(c) Prompt: *A mountain valley with a river at sunset* where the attention weight of the word *sunset* is increased from L - R

Figure 11. More results from cross attention injection: By swapping a word of the initial prompt in a) and b) and attention re-weighting in c)

(a) Prompt: *A green school bus across the river in winter*



(b) Attention map for each token

Figure 12. A sample prompt and its corresponding attention map for each token