Starting Great Adventure: A Deep Dive into Stable Diffusion

Xiangrui Kong*
Purdue University
kong98@purdue.edu

Gilbert Hsu*
Purdue University
hsu226@purdue.edu

Abstract

We explore the integration and advancements in the fields of computer vision and natural language processing through the lens of deep learning technologies, with a particular focus on the development and application of the text-to-image generative model, Stable Diffusion. We begin by examining the mathematical foundation shared by the original Denoising Diffusion Probabilistic Models and its more advanced iteration, Stable Diffusion. The project involved reconstructing Stable Diffusion version 1.5 from scratch, utilizing online resources and pre-trained weights to facilitate forward inference experimentation. To complement the experience for training, a simplified version of the DDPM was developed specifically for smaller datasets including MNIST, Fashion-MNIST, and CIFAR-10. Experimental outcomes with MNIST were particularly encouraging, showcasing clear class recognition in generated images. However, as the complexity increased with the Fashion-MNIST and CIFAR-10 datasets, the performance of our streamlined model declined, indicating the model's limitations with complex inputs. Additionally, our exploration included integrating pre-trained weights into our Stable Diffusion model, yielding mixed but intriguing results. In addition, our educational contributions include the creation of a comprehensive Jupyter Notebook which elucidates the mathematical and conceptual framework of these models and includes executable Python code with annotations. The study provides a valuable experience in exploring stable diffusion models, translating theoretical knowledge from lectures into practical applications with widely used models. It bridges the gap between academic learning and real-world implementation, offering insights into the complexities and capabilities of current AI technologies.

1. Introduction

In the rapidly evolving landscape of deep learning, two pivotal fields have witnessed significant breakthroughs and advancements: computer vision (CV) and natural language processing (NLP). Computer vision, the branch of artificial intelligence that deals with how machine learning models can be made to gain high-level understanding from digital images or videos, has made remarkable progress in recent years. Techniques such as convolutional neural networks (CNNs), generative adversarial networks (GANs), and other models have dramatically improved the ability of machines in either classification or generative tasks. At the same time, the field of natural language processing similarly has been through major advancements. Leveraging deep learning models to parse, understand, and generate human language, NLP models such as recurrent neural networks (RNNs) and, more strikingly, transformers largely facilitate and influence a wide range of explorations in machine learning. Standing in the middle ground, the state-of-the-art text-to-image generative model Stable Diffusion utilized the strengths of these two fields. Originated from the Denoising Diffusion Probabilistic Models (DDPM), the Stable Diffusion uses latent diffusion model achieving remarkable performance and make itself the cornerstone in text-to-image synthesis.

In our final project, we firstly dived into the mathematical backbone, shared by original DDPM model and stable diffusion. Then we explored and built the Stable Diffusion version 1.5 from scratch using online references. However, considered the massive training time and resources needed for the entire model, we load the pre-trained weights and experimented the model for forward inference. In order to gain our own training experiences for diffusion model, we additionally built a simplified version of DDPM model targeting only for smaller datasets including MNIST, Fashion-MNIST, CIFAR-10, and achieved reasonable results. Finally, as a takeaway from this project opportunity, and aiming for educational purpose, we created one Jupyter Notebook introducing and explaining our findings and outcomes along the way, which contains detailed math formulations, logical concepts behind each components in the model, as well as the python implementation with comments. We hope this brief tutorial could help other students and learners ease the high learning edges towards such a complicated but intriguing topic.

2. Related Works

The field of image generation has experienced substantial progress over the last decade, driven by advancements in deep learning architectures and the availability of large datasets. The emergence of **Generative Adversarial Networks**(GANs) [2] marked a significant milestone in the capacity of models to generate high-quality, realistic images. GANs operate on a framework where two neural networks, the generator and the discriminator, are trained concurrently in a adversarial scenario. The generator creates images intended to be indistinguishable from real images, while the discriminator evaluates their authenticity by predicting whether the image is generated or sampled from the datasets. This technique has been extensively explored and refined, yielding various iterations such as DCGAN, and BigGAN [1, 7] each improving the fidelity and resolution of generated images.

Parallel to the development of GANs, **Variational Auto-encoders** (VAEs) [5] have been another cornerstone in generative models. VAEs frame image generation as an optimization problem, aiming to approximate the underlying probability distribution of training data through latent variables. VAEs excel in generating structured outputs and ensuring stable training. Even though they generally lag behind GANs in producing images with comparable sharpness and detail, their structures and reparameterization tricks are universally implemented in later generative models for the task of learning model latent space, including Stable Diffusion.

Emerging from these developments, **Diffusion Probabilistic Models** have gained prominence as a powerful alternative for generating high-quality images since the publication of Denoising Diffusion Probabilistic Models (DDPM) [3]. Divided into two separate processes, DDPMs firstly add Gaussian noises to the input images in the forward process, and conversely convert random noise into coherent images through a reverse Markov process. The generative power of this model stems from the prediction of the underlying neural network UNet. Utilizing similar but more complex structure, Stable Diffusion [9] integrates these diffusion techniques with insights from the field of natural language processing [6] to facilitate text-to-image generation. The model employs a transformer-based architecture, enabling it to interpret textual descriptions effectively. This cross-disciplinary approach leverages the strengths of DDPMs in image synthesis and the advanced capabilities of transformers in language understanding, positioning Stable Diffusion as a state-of-the-art representative in the field of creative AI models.

Comparatively, other **text-to-image models** like DALL-E [8] and Imagen [10] also seek to take advantages from both NLP and computer vision but utilize different underlying mechanisms. DALL-E employs a variant of the GPT-3 architecture to generate images from textual prompts, whereas Imagen enhances this integration by using a more refined text encoder. Each of these models presents unique approaches and trade-offs regarding image quality, training complexity, and resource requirements.

3. Approach

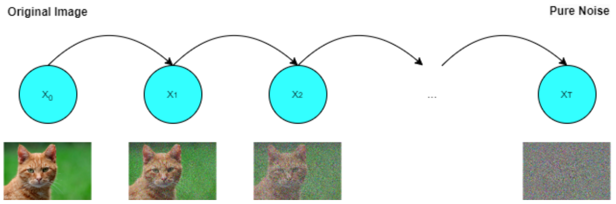
In our investigation of Stable Diffusion, we began by familiarizing ourselves with the foundational principles of Diffusion Models, focusing on the underlying mathematics as the maths acted as the backbone for both DDPM and the latent Diffusion approach integral to Stable Diffusion. We then embarked on reconstructing Stable Diffusion version 1.5 from scratch using only PyTorch. Through hands-on experimentation with our model and extensive review of existing literature, we realized that training the full Stable Diffusion model, with its myriad complex components, was beyond our current capabilities; hence, we confined our efforts to forward inference to engage with the model's capabilities. To further our understanding and practical experience with diffusion models, we developed a simplified version of DDPM tailored for smaller datasets, which allowed for manageable training duration.

In this section, we present our findings in the following structure: initially, we delve into the mathematical foundations; subsequently, we explore the streamlined, trainable DDPM; and conclude with the implementation of Stable Diffusion. In the first part, given the dense and complex mathematical content inherent in the model, our discussion will focus on key equations while omitting intermediate steps. Furthermore, from an educational perspective, when introducing the mathematics, we will incorporate our insights and interpretations. For a more detailed analysis, please refer to our comprehensive notebook . In the final part, recognizing the shared fundamental architecture between DDPM and Stable Diffusion and avoiding discussing repetitive contents, we will thoroughly examine the additional components that distinguish these models.

3.1. Math of DDPM

To provide a clear overview, the DDPM (diffusion) model fundamentally involves a two-step process: the forward process, which adds noise, and the reverse process, which focuses on denoising. These steps are distinctly categorized into two separate phases. The core concept of this model is based on a defined sequence: if we know how to systematically add noise to an image until it becomes entirely random, then, conversely, we can predict and remove the noise at each step in the reverse process, ultimately reconstructing a clean, noise-free image.

3.1.1 Forward Process



In the diffusion model, the forward process is formalized by the equation

$$X_t = \sqrt{a_t}X_{t-1} + \sqrt{1-a_t}Z_t \tag{1}$$

where: X_t represents the image at timestep t, which has undergone t iterations of noise addition. Z_t denotes the noise introduced at each timestep, drawn from a Gaussian normal distribution, $\mathcal{N}(0,1)$. Timestep t quantifies the sequential stages of the process, with each stage corresponding to an incremental introduction of noise. The noise level parameter a_t is defined as $a_t = 1 - \beta_t$, where β_t is a hyperparameter that incrementally increases with each timestep. This increment in β_t results in a corresponding decrease in a_t. The rationale for defining a_t in this manner is rooted in the dynamics of noise addition. As a_t diminishes, $\sqrt{1-a_t}$ increases, thereby enhancing the proportion of noise added at each successive timestep. This increase is crucial because, in the initial stages, even minimal noise is conspicuous against a clean image. However, as the image accumulates noise, subsequent noise additions become less perceptible. Consequently, escalating the noise intensity with each timestep ensures that its impact remains significant, thereby facilitating a controlled progression from a clean to a thoroughly noised state. This mechanism underlies the effectiveness of the noise addition strategy in the diffusion process.

One notable feature of the forward process, rather than incrementally adding noise to an image at each timestep, a more efficient approach directly relates the initial image X_0 to its noised state X_T with arbitrary timestep t in closed form. Expanding the recursive formula $X_t = \sqrt{a_t}X_{t-1} + \sqrt{1-a_t}Z_t$. Through iterative substitution and by the multiplication rule of Gaussian distribution, this expansion yields:

$$X_t = \sqrt{a_t a_{t-1}}X_{t-2} + \sqrt{1-a_t a_{t-1}}\bar{Z} \tag{2}$$

where $\bar{Z} \sim N(0,1)$. Therefore, following same derivation, the formula refines to a close form:

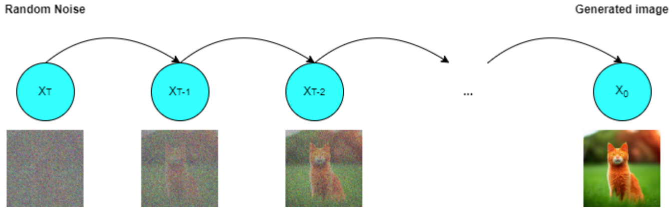
$$X_T = \sqrt{a_t a_{t-1}...a_1}X_0 + \sqrt{1-a_t a_{t-1}...a_1}\bar{Z} \tag{3}$$

$$X_t = \sqrt{\bar{a}_t}X_0 + \sqrt{1-\bar{a}_t}\bar{Z} \tag{4}$$

where $\bar{a}_t = \prod_1^t a_i$.

This final equation simplifies the forward process of adding noise, summarizing the transformation from a clean image to a fully noised state in a single step. Having established the forward process, the next phase involves reversing this procedure, known as the denoising process, which is critical for reconstructing the original image from its noised state.

3.1.2 Reverse Process



In the diffusion model, the reverse process serves to incrementally remove noise from an image, contrary to the forward process that introduces noise. To elucidate the reverse process, we consider reconstructing the less noisy image X_{t-1} from a given noisy image X_t. The transition from the forward to the reverse process leverages Bayes' Theorem, enhanced by including the initial image X_0:

$$q(X_{t-1}|X_t, X_0) = \frac{q(X_t|X_{t-1}, X_0)q(X_{t-1}|X_0)}{q(X_t|X_0)} \tag{5}$$

where each of the probabilities are defined:

$$q(X_t|X_{t-1}, X_0) = \sqrt{a_t}X_{t-1} + \sqrt{1-a_t}Z_t \tag{6}$$

$$q(X_{t-1}|X_0) = \sqrt{\bar{a}_{t-1}}X_0 + \sqrt{1-\bar{a}_{t-1}}\bar{Z} \tag{7}$$

$$q(X_t|X_0) = \sqrt{\bar{a}_t}X_0 + \sqrt{1-\bar{a}_t}\bar{Z} \tag{8}$$

The key to the reverse process lies in integrating over the distributions derived from the forward model, specifically through the manipulation of Gaussian distribution properties, where $q(X_t|X_{t-1}, X_0)q(X_{t-1}|X_0) \propto$

$$exp(-\frac{1}{2}(\frac{(X_t - \sqrt{a_t}X_{t-1})^2}{1-a_t} + \frac{(X_{t-1} - \sqrt{\bar{a}_{t-1}}X_0)^2}{1-\bar{a}_{t-1}})) \tag{9}$$

From this formulation, we can observe that combining the distributions simplifies to the summation of the exponentiated terms of their variances and means. Therefore for the entire equation, $q(X_{t-1}|X_t, X_0) = \frac{q(X_t|X_{t-1},X_0)q(X_{t-1}|X_0)}{q(X_t|X_0)} \propto$

$$exp(-\tfrac{1}{2}(\tfrac{(X_t-\sqrt{a_t}X_{t-1})^2}{1-a_t} + \tfrac{(X_{t-1}-\sqrt{\bar{a}_{t-1}}X_0)^2}{1-\bar{a}_{t-1}} - \tfrac{(X_t-\sqrt{\bar{a}_t}X_0)^2}{1-\bar{a}_t})) \tag{10}$$

The expression encapsulates the Gaussian nature of the noise removal process, highlighting the influence of the noise parameters (a_t and $\bar{a}_{t-1}$) and the noise standard deviation ($\sqrt{1-a_t}$, $\sqrt{1-\bar{a}_{t-1}}$) on the posterior distribution of X_{t-1}.

By extracting X_{t-1} from this complex formulation, we arrive at a representation of X_{t-1} as a Gaussian random variable centered around a mean μ that depends linearly on X_t and X_0, with a variance that reflects the cumulative noise parameters up to $t-1$. This demonstrates how previous states (X_0 and X_t) influence the estimation of X_{t-1} in the reverse process:

$$\frac{1}{\sigma^2} = \frac{a_t(1-\bar{a}_{t-1}) + \beta_t}{\beta_t(1-\bar{a}_{t-1})} \tag{11}$$

and

$$\frac{2\mu}{\sigma^2} = (\frac{2\sqrt{a_t}X_t}{\beta_t} - \frac{2\sqrt{\bar{a}_{t-1}}X_0}{1-\bar{a}_{t-1}}) \tag{12}$$

By further simplification, we reach the closed form:

$$\sigma^2 = \frac{1-\bar{a}_{t-1}}{1-\bar{a}_t}\beta_t \tag{13}$$

$$\mu = \frac{1}{\sqrt{\bar{a}_t}}(X_t - \frac{\beta_t}{\sqrt{1-\bar{a}_t}} \cdot Z_t) \tag{14}$$

By observation, the primary unknown component of the above equations are the noise term involved in constructing the mean in the reverse process of the diffusion model. To address this uncertainty, we employ a deep neural network to estimate the noise characteristics effectively. This neural network serves as a parameterized function that learns to approximate the noise distribution from the observed data, thereby facilitating the denoising process by accurately reconstructing the mean and standard deviation of the underlying noise distribution.

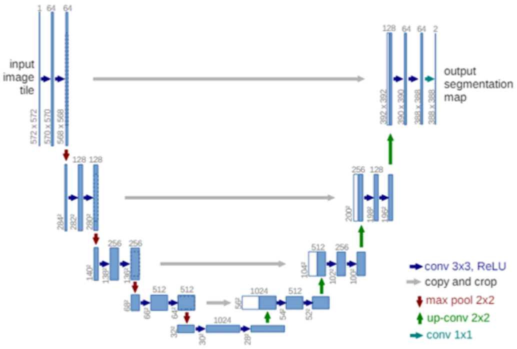
3.2. Implementation of Trainable DDPM

Given the considerable computational demands and complexities associated with training stable diffusion models on our own computers, we have build a diffusion model that is trainable on personal hardware. Retaining the fundamental architecture of the U-Net, we have strategically modified the encoder of input noises to consist solely of convolutional layers. This decision is predicated on the understanding that convolutional layers are capable of generating embeddings with sufficient representational capacity for the scope of our objectives.

We also generate images directly from the noise, bypassing the need for a separate decoder of output images. We follows the classifier-free guidance scheme where the output is influenced by the presence or absence of context (i.e., class labels in this case). The predicted noise is updated in-place at every timestep, with the U-Net providing the necessary transformations guided by the context labels and noise levels. The method of 'Classifier-Free Diffusion Guidance' is used to control the generation without needing an external classifier or decoder. It uses two versions of the input, one with the context and the other without, and combines them using a guidance scale guide_w. [4]. Moreover, in place of a traditional text encoder, we have implemented a one-hot encoding scheme to process the labels of the 10 classes present in our target datasets, such as MNIST. This adaptation aligns with our focus on non-textual data and simplifies the model by removing unnecessary complexity associated with natural language processing. The one-hot encoding provides a direct and efficient method of injecting class-specific information into the model, allowing us to concentrate on the visual aspects of the data.

Through these purposeful choices, our model is better suited to environments with limited computational resources while still being poised to exploit the distinctive characteristics of datasets like MNIST. This custom approach paves the way for experimentation and potential advancements in the application of diffusion models to specialized domains of image data. We will briefly describe the structure of our model.

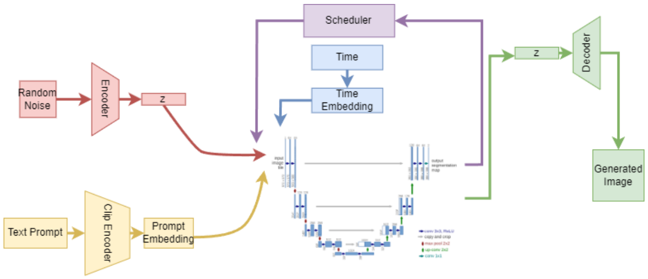
The ResidualConvBlock is a basic architectural component within our trainable diffusion model, adhering to the design principles of a standard ResNet-style convolutional block. The block comprises two convolutional layers, each followed by batch normalization and a GELU activation function. It also provides an option to integrate a residual skip connection, which is a feature borrowing from ResNet architecture.



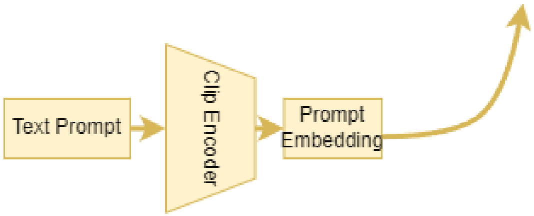
For the Unet, there are three components which are U-Net Downscaling Block, U-Net Upscaling Block, and Embedding Fully Connected Block. The Unet down has a ResidualConvBlock used to maintain the integrity of the input features and a MaxPool2d layer with a kernel size of 2 to reduce the feature map dimensions by half, effectively condensing the image information and increasing the receptive field for subsequent layers. On the other hand, the Unet Up class is designed to perform the inverse operation within the U-Net structure—namely, the upscaling of feature maps. It consists a ConvTranspose2d layer for spatial upscaling of the feature maps, doubling their dimensions, followed by two successive Residual Convolutional Blocks, which refine the upscaled features and enhance the model's ability to reconstruct details in the output image. The Embedding Fully Connected Block serves as a generic, one-layer fully connected neural network for embedding. The block contains a Linear layer to map the input to the embedding space, followed by a GELU activation function. A second Linear layer further transforms the embedding, allowing for a richer and more nuanced representation.

By integrating these components, we have build a U-Net equipped with time embedding and a context mask that predicts noise in images using DDPM. The model first processes images through initial convolution and downsampling to create feature-rich vectors. It then seamlessly incorporates temporal and contextual data via specialized embedding layers, ensuring the output is both contextually accurate and temporally consistent.

3.3. Implementation of Stable Diffusion

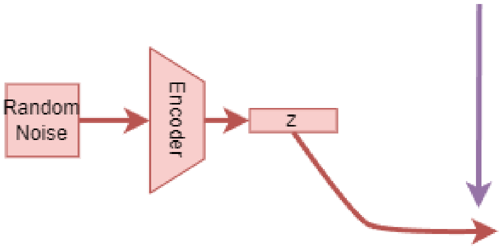


Since we have already discussed the U-Net and Denoising Diffusion Probabilistic Models (DDPM) in previous sections, we will now focus on three critical add-on components in the Stable Diffusion : the CLIP Encoder for text prompts, the VAE Encoder of input noise, and the VAE Decoder for output images.



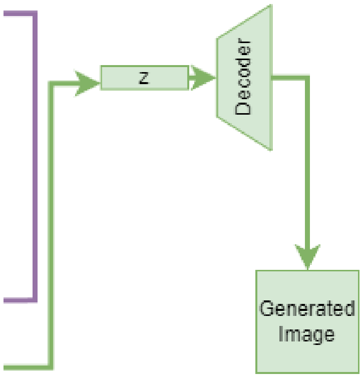
The CLIP text encoder is instrumental in converting textual input into rich, contextual embeddings, utilizing a design inspired by the Transformer architecture. This encoder has two primary components: the CLIP Embedding and the CLIP Layer. The CLIP Embedding initially maps each token from a predefined vocabulary to a high-dimensional space, integrating a unique position vector for each token to maintain sequential awareness. This setup ensures the model recognizes the order of tokens, which is crucial for understanding the context of the text. The CLIP Layer, embodying a single layer of Transformer-style architecture, manages sequences of embeddings. It begins by applying layer normalization to the input embeddings, followed by self-attention, which allows the model to focus on relevant parts of the text adaptively. A residual connection is then added to the original input, enhancing the gradient flow and enabling the construction of deeper models without the risk of vanishing gradients. Subsequently, another residual connection follows, which passes through a series of linear transformations and a non-linear activation function (a quick GELU approximation) to refine the feature representation further.

With the integration of the CLIP Embedding and CLIP Layer, the model achieves a sophisticated handling of textual data, preparing it for further processing. This setup ensures that the embeddings generated are not only contextually rich but also temporally coherent, which is pivotal when these text features are used to guide the image synthesis process in stable diffusion models



For the VAE Encoder, it is used to compress input images into a compact latent space representation, facilitating the generation of new images based on the learned distribution. Starting with a convolutional layer that expands the input from 3 color channels to 128 feature channels, it sets the stage for deeper feature processing. Subsequent layers include multiple VAE ResidualBlock units that enrich feature maps while maintaining depth, distributed with stride-based convolutions for downsampling. It effectively reduce spatial dimensions while increasing depth up to 512 channels. An attention block refines the ability to focus on principle features, crucial for capturing complex dependencies within images.

The encoder's role extends beyond compression, serving as the bridge to the latent space where image features are encoded as distributions characterized by means and variances. This setup facilitates the generation of new images through the reparameterization trick, allowing for noise-injected sampling from these distributions. The architecture concludes with a bottleneck formation via convolutional layers, reducing features to a minimal set, and a final empirical scaling to normalize the outputs. This structured approach enables the encoder to efficiently extract essential information from input images, preparing them for effective reconstruction or variation in the generative phases of the model.



Lastly, for the VAE Decoder, it is integral for transforming the intermediate outputs from the U-Net into high-resolution images. This decoder includes a series of upsampling and convolutional processes, supplemented by residual blocks and attention mechanisms, to progressively refine and enhance the details of the final images.

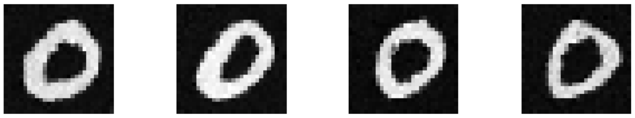
The VAE Decoder is structured to enhance and finalize images that have been iteratively refined through a diffusion process. It begins with a series of convolutional layers that adjust and upscale the intermediate features provided by the U-Net. Upsampling layers with these blocks gradually increase the spatial resolution, enhancing the detail until the final output matches the desired high-resolution.

The primary role of this decoder is to ensure that the final image output is of high quality and resolution. It accomplishes this by enhancing the detail and clarity of the upsampled features, adjusting color, texture, and edges to produce a visually compelling image. This component is crucial in stable diffusion models where the fidelity and realism of generated images are paramount, making it particularly useful in applications that require high-quality visual outputs.

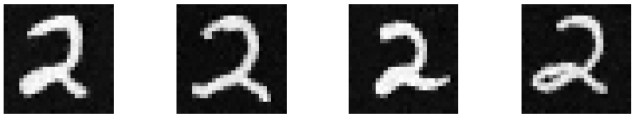
4. Results

We initiated our experiments with the MNIST dataset, which comprises a simple collection of single-channel images of digits. The training was conducted over 10 epochs, using a batch size of 256 and a training/inference time step of 400. The beta parameters were set between 1e-4 and 0.02. Below are the results for selected classes within the dataset:

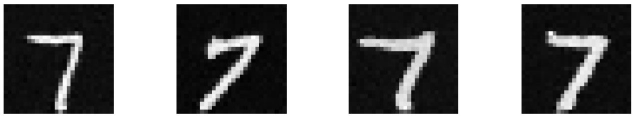
1. Zero



2. Two



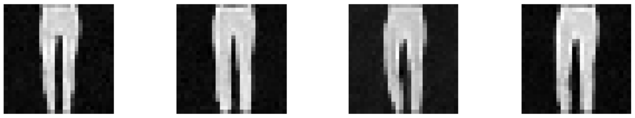
3. Seven



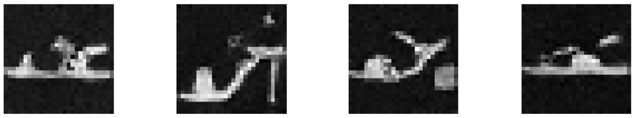
The outcomes were highly satisfactory; the classes of the generated images could be easily discerned, confirming the effectiveness of our model on this simpler dataset.

Subsequently, we applied the same model to the Fashion-MNIST dataset, a collection of Zalando's article images, also with a single-channel format. Utilizing identical hyperparameters—10 epochs, batch size of 256, and the same range of beta values—we observed the following results for some of the classes in this dataset:

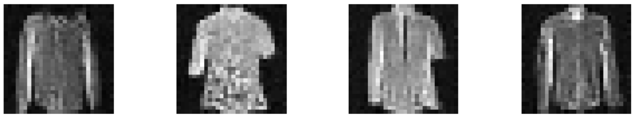
1. Trousers



2. Scandals



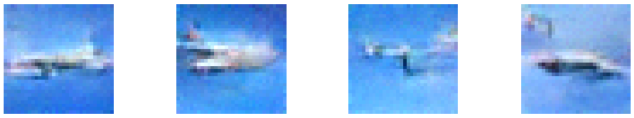
3. Shirt



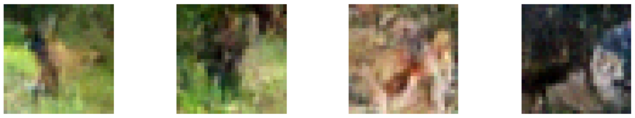
In our examination of the images generated from the MNIST Fashion dataset, it is apparent that while the images are recognizable, they do not match the higher quality observed in the MNIST digit dataset. For instance, the generated image of sandals exhibits noticeable imperfections, such as scratch-like artifacts. Additionally, the images of shirts display inconsistencies; notably, the sleeves of the middle two shirts exhibit varying lengths. These discrepancies can primarily be attributed to the inherent challenges posed by the MNIST Fashion dataset. Unlike the MNIST digit dataset, which consists of relatively uniform and simple shapes, the MNIST Fashion dataset contains more complex and varied patterns. This complexity arises from diverse clothing item shapes, varying textures, and intricate details such as seams and buttons, all of which are challenging for the model to accurately capture and reproduce.

Lastly, we extended our testing to the CIFAR-10 dataset to evaluate whether our model could adequately learn and generate images from a even more complex dataset. Initially, we used the same hyperparameters as those applied to the MNIST datasets. However, the results were unsatisfactory, prompting us to increase the number of epochs to 20 and the training/inference time step to 1000. Despite these adjustments, the images remained difficult to identify without accompanying labels, suggesting that CIFAR-10's complexity exceeds the capabilities of our relatively simple model. Nevertheless, some discernible patterns were evident in the generated images, indicating partial success in feature capture. Here are the output of some classes:

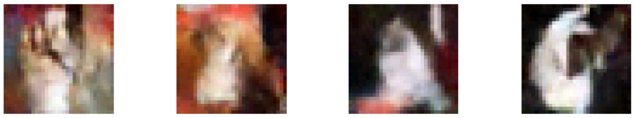
1. Airplane



2. Deer

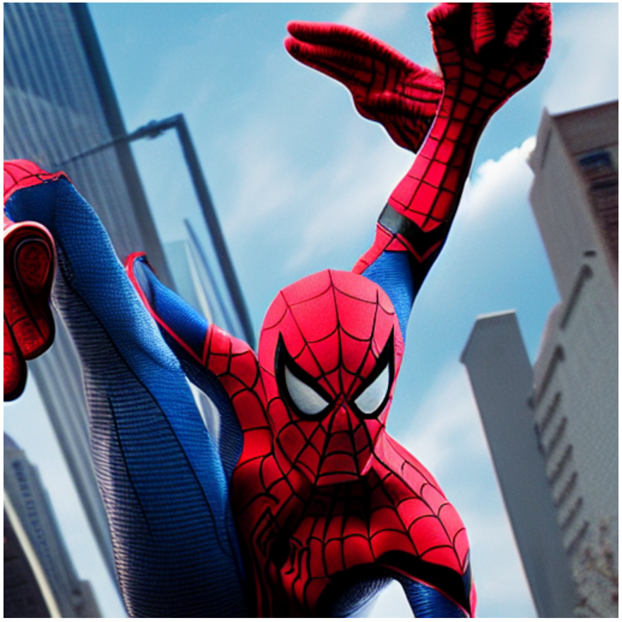


3. Dog

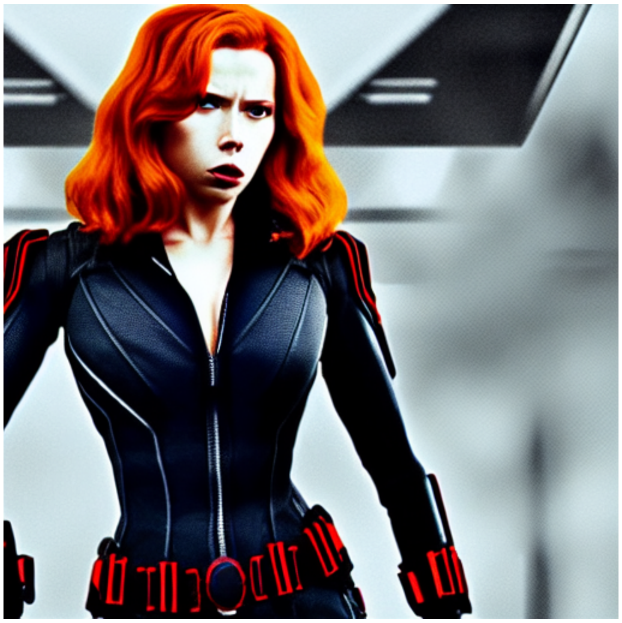


Upon inspection of the generated images, the outline of airplanes are identifiable with a blue background color, suggesting its placement in the sky. Similarly, in the image depicting a deer, the lush green colors in the scene create the feeling of being in a grassy landscape. Despite these contextual cues, identifying the precise form of the deer is still challenging, with its contours blending ambiguously into the surroundings. This issue of clarity also exists in the images of dogs, where the intricacies of their shape are not as distinct as original dataset, rendering accurate identification is somewhat difficult for our model. This series of experiments highlights the challenges and adjustments necessary for embedding more machine learning components in diffusion models' pipeline.

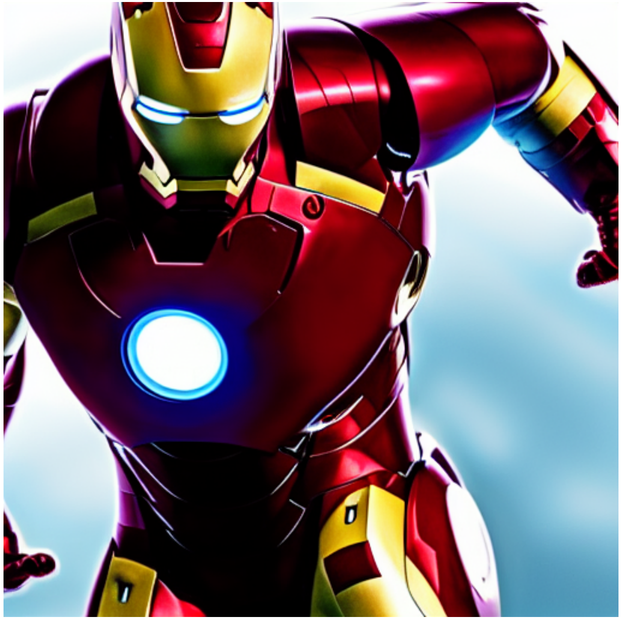
In addition to our efforts to develop and train a compact diffusion model tailored for small datasets, we have incorporated pre-trained weights into our stable diffusion model (version 1-5). Despite the limitations posed by our hardware, which preclude us from conducting full-scale training on our own computer, we remain committed to exploring the capabilities of this advanced model. To facilitate this exploration, we have utilized a variety of descriptive prompts such as ”Spiderman sliding between buildings,” along with the iconic characters ”Iron Man” and ”Black Widow.” These prompts are designed to specifically direct the image synthesis process, enabling us to assess and refine the model's ability to generate detailed and contextually accurate visual representations based on these well-known figures.



The results were intriguing and displayed varying degrees of realism. The image of Spiderman conveyed a high degree of details, though it was evident that the representation of his limbs deviated from expected human proportions, compromising the overall authenticity of the scene. Black Widow's portrayal was markedly improved, yet it was observed that the facial features, particularly around the mouth, exhibited subtle anomalies that detracted from the lifelike quality of the image.



Conversely, Iron Man's depiction approached perfection, resonating with realistic textures and convincing contours. We conjecture that the model's relative success in rendering Iron Man with such precision is attributable to the character's mask, which presents a simpler task for the generative model due to its non-biological and rigid structure. Faces, on the other hand, are difficult due to their nuanced expressions, subtle asymmetries, and dynamic textures—elements that are currently more challenging for diffusion models to capture with impeccable accuracy.



The gap in the generated images underscore the limitations and a early-version of diffusion-based generative models. Furthermore, this also highlights future research directions, encompassing the enhancement of the model's ability to navigate the subtle nuances of human features and attain a consistent level of realism across a wide range of prompts.

5. Conclusions

In conclusion, our project's in-depth analysis and practical implementations of Denoising Diffusion Probabilistic Models (DDPM) and Stable Diffusion have significantly enhanced our understanding of these complex machine learning frameworks. Through the meticulous exploration of the mathematical foundations of DDPM, accompanied by a detailed commentary, we have endeavored to understand the underlying mechanisms that facilitate the generation of high-quality images. Furthermore, by adapting and training a DDPM, followed by integrating and manipulating a pretrained Stable Diffusion model, we have not only validated the theoretical concepts but also gained invaluable hands-on experience. Our comprehensive Jupyter notebook serves as a educational tool that guides the reader through each mathematical and coding step, illuminated with insightful commentary. This resource is designed to flatten the learning curve for future researchers and enthusiasts aiming to delve into the area of text-to-image generative models. It is our hope that our contributions will empower more individuals to engage with and push the boundaries of this exciting field, thereby fostering innovation and expanding the community's collective knowledge.

References

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

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

[3] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. 2

[4] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022. 4

[5] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022. 2

[6] 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. 2

[7] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016. 2

[8] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation, 2021. 2

[9] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022. 2

[10] 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. 2