TEXT-TO-IMAGE GENERATOR

Tribhuvan University

Institute of Science and Technology

A Final Year Project Submission in

Partial Fulfillment of the Requirement for the Degree of

Bachelor of Science in Computer Science and Information Technology

Under the Supervision of

Mr. Avishek Kuinkel

Submitted by:

Neha Shrestha (24287 / 7^{th} Semester / 2076)

Norden Ghising Tamang (24290 / 7^{th} Semester / 2076)

Submitted to:

TRINITY INTERNATIONAL COLLEGE

Department of Computer Science and Information Technology

Dillibazar Height, Kathmandu, Nepal

January 18, 2024

ABSTRACT

The “Text-to-Image generator” presents an innovative approach to synthesize visual content from textual descriptions. Leveraging the power of LDMs, the system employs a novel iterative process to refine a latent representation space, enabling the generation of high-quality images from diverse textual prompts. The project aims to provide users with a versatile tool for creative expression, educational illustration, and efficient content creation by combining the strengths of latent diffusion models and convolutional neural networks in a unified framework. This project uses advanced Generative AI techniques like the diffusion model, UNet, and Variational Autoencoder to generate visual outputs.

The project's core focus lies in image generation, where textual prompts act as creative guides the generation process. By incorporating the diffusion model, the system undergoes a refined process of noise reduction in a latent space, resulting in the production of images. Variational Auto-encoders used for latent image manipulation and UNet is used for prediction of noise. Through iterative Markov process, an image is generated.

The implementation employs the PyTorch framework along with the Diffusers library ensuring a flexible and efficient development environment. These frameworks provide specialized toolkit for managing the diffusion process, optimizing performance, and enhancing the overall stability of the image generation pipeline.

Keywords: *UNet, Variational Auto-encoders, DDPM, CNN, Python Programming, PyTorch, Diffusers*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

CE	Conditional Embedding
CNN	Convolutional Neural Networks
DDPM	Denoising Diffusion Probabilistic Models
GAN	Generative Adversarial Network
LDM	Latent Diffusion Model
MSE	Mean Squared Error
SiLU	Sigmoid Linear Units
TE	Time Embedding
VAE	Variational Auto-encoders
VLB	Variational Lower Bound
VQGAN	Vector Quantized Generative Adversarial Network

CHAPTER – 1: INTRODUCTION

1.1 Introduction

Generative AI, an evolving paradigm within artificial intelligence, showcases a creative and innovative part of technology that empowers machines to produce content resembling existing data. This versatile technology spans various mediums including images, text, audio, video and more. The concept behind this mechanism involves extracting and understanding patterns and information present in the data on which the models are trained.

One of the interesting generative AI models is the LDM. In contrast to the previously defined models, this model does not rely on extensive spatial compression. It operates within a learned latent space which is characterized by superior scaling properties concerning spatial dimensionality [1]. In simpler terms, the model doesn't compromise image quality or information while compressing spatial aspects, ensuring a more effective representation of the data. The main idea behind this is to systematically decompose the data structure through an iterative forward diffusion process. Then a reverse diffusion process is applied that restores structure in the data, yielding a highly flexible and tractable model of the data. This diffusion probabilistic (diffusion) model operates as a latent text-to-image diffusion model, repeatedly diminishing noise in a latent representation space and then transforming that representation into a complete image [3].

This project proposes a model that shows the interplay of noise and neural networks to produce compelling results. Thousands of images are meticulously registered with layers of noise, forming a corpus of pure noise images. Subsequently, these noisy images undergo training in a Markov chain process to revert to their original state through neural networks [3]. The machine learns to discern and eliminate this noise so adeptly that the model becomes proficient in transforming any arbitrary noisy input into a new image that aligns with our training data. This process empowers the model to generate the desired image corresponding to any textual prompt.

1.2 Problem Statement

Generative AI emerges as a solution to the challenge of data scarcity by enabling the creation of vast training datasets. Moreover, overcoming communication barriers associated with novel concepts, the text-to-image model serves as a facilitator for improved understanding and learning. Furthermore, this model expedites the transformation of ideas into visual representations, offering a swift and intuitive process devoid of technical complexities. In essence, generative AI not only addresses data limitations but also acts as a catalyst for enhanced communication and streamlined visual expression, catering to a diverse range of users who may encounter difficulties in grappling with new information.

1.3 Objective

The objectives of this project are:

- To transform textual descriptions into visually appealing images.
- To ensure the generated images are coherent and contextually accurate.
- To provide a user-friendly interface for users to input text and retrieve generated images.

1.4 Scope & Limitations

The scopes of this project are:

- The project can be used to express textual prompts to visual images.
- The system can serve as an educational tool, aiding in the creation of visual aids for educational materials.
- Preferred text prompts can be given by users to derive images.
- The project facilitates easy user interaction.

The limitations of this project are:

- The project's limitation lies in its ability to generate only a limited class of images (2-5).
- The quality and diversity of generated images heavily depend on the training data.
- The system might struggle with highly complex and abstract textual prompts.
- Without a powerful GPU system, processing is slow and scalability is limited.

1.5 Development Methodology

The developmental methodology for this project involves conducting a thorough literature review on LDM and Diffusion Probabilistic Models. Further, the procedures are collecting and preparing relevant datasets, adding noise to the datasets, designing U-Net with proper attention mechanism and embeddings of time and conditions, implementing VAEs, executing the model using different frameworks, evaluating and validating the model's performance, iteratively refining the model based on the feedback and evaluation metrics, and documenting the entire development process in a comprehensive report.

1.6 Report Organization

The report is organized as follows:

Chapter 1 introduces the concept of Generative AI and its ability to craft diverse content formats. The focus here is on probabilistic models that can generate high-quality images and propose the use of latent diffusion models. This chapter outlines the problem statement, objectives, scope and limitations of the project. Overall, chapter 1 lays the foundation for the rest of the report by briefly addressing the goals of the project.

Chapter 2 of the report contains background information and a literature review on the topic of the diffusion probabilistic model and its utilization in the generation of sophisticated images. The literature review covers various studies on the use of a combination of CNNs and transformers, employing pre-trained autoencoders and applying cross-attention mechanisms. The chapter provides a summary of the major findings and contributions of each study.

Chapter 3 of the report is dedicated to system analysis. This section covers the study of several analyses such as requirement analysis, feasibility analysis and project analysis. The requirement analysis includes both functional and non-functional requirements. The feasibility analysis examines technical, schedule and economic feasibility. Moreover, the analysis section provides a flow diagram that illustrates the project's workflow. Specifically, a flow diagram depicts the step-by-step process of the project, providing a concise understanding of how the project works.

Chapter 4 of this report emphasizes the design aspect of the project. This commences with a sequence diagram that outlines the sequence of interactions between the system and the user. Following that is an activity diagram that exemplifies the various stages that occur within the system. Additionally, this chapter includes a detailed description of the algorithm used in the project.

Chapter 5 describes the implementation and testing part of the project. This chapter includes the tools used for building the system i.e. the programming language and frameworks.

CHAPTER – 2: BACKGROUND STUDY AND LITERATURE REVIEW

2.1. Background Study

Diffusion models are a class of generative models in machine learning that simulate the data generation process by transforming a simple and easily sampleable distribution, typically a Gaussian distribution, into a more complex data distribution of interest. Diffusion models were inspired by the natural diffusion process in physics, which describes how molecules move from high-concentration to low-concentration areas.

The idea of diffusion models was first introduced in a 2015 paper named "The Deep Unsupervised Learning using Nonequilibrium Thermodynamics" and gained momentum in 2020 with the publication of several papers [6]. In this paper, the approach to achieving both flexibility and tractability is to systematically and slowly destroy data structure through an iterative forward diffusion process followed by the reversal of a Markov diffusion chain. The result is an algorithm that can learn a fit to any data distribution which is straightforward to manipulate conditional and posterior distributions.

Diffusion models have diverse applications across several domains, such as text-to-video synthesis, image-to-image translation, image search, and reverse image search. They have better image quality, interpretable latent space, and robustness to overfitting compared to traditional generative models.

2.2. Literature Review

The article “Denoising Diffusion Probabilistic Models” is built on top of the 2015 paper “The Deep Unsupervised Learning using Nonequilibrium Thermodynamics" [3]. Here, the authors took inspiration from nonequilibrium thermodynamics and found a correlation between diffusion probabilistic models and denoising score matching with Langevin dynamics. They have presented a class of latent variable models that use a parameterized Markov chain to gradually add noise to the data in the opposite direction of sampling. The models are trained using a forward diffusion process that maps data to noise and a learnt, parametrized reverse process that performs iterative denoising, starting from pure random noise. The parameterization of the diffusion models claims to be the primary contribution to the best sample quality results. They make a clear statement that their models do not have competitive log-likelihoods compared to other models. The sampling procedure of the diffusion model is discovered to be progressively decoding often resembling autoregressive decoding.

In another article by Nicole et al., a few simple modifications are made on top of the above paper to achieve an improved version of DDPM [5]. In this paper, DDPMs can achieve log-likelihoods with other likelihood-based models even on highly diverse datasets like ImageNet. The authors discovered a learnable variance schedule in the reverse process using simple reparameterization and hybrid learning objectives that combine the VLB with a simplified objective from the above paper. They found that while the linear noise schedule used in the previous paper worked well for high-resolution images, it was sub-optimal for images of resolution 64×64 and 32×32. Therefore, to address the problem of a forward noising process being too noisy, they have constructed a cosine schedule which has a more gradual loss. Incorporating all the requirements, they realized that pre-trained hybrid models could achieve good samples with as few as 50 forward passes if the previous DDPM required 100s of forward passes to produce good samples. They have additionally used precision and recall to compare the distribution coverage between DDPMs and GANs, finally noting that the former covered a larger portion of the target distribution.

In a study by Esser et al., the efficiency of synthesizing high-resolution images is maximized by integrating the usage of transformers as well as CNNs [2]. The power of the inductive bias of CNNs with the articulation of transformers enabled to model and thereby produce quality images. The approach is to use a convolutional VQGAN to learn a codebook of context-rich visual parts, whose composition is subsequently modelled with an autoregressive transformer architecture. A discrete codebook provides the interface between these architectures and a patch-based discriminator enables strong compression while retaining high perceptual quality. The study was bounded up to only 16×16 discrete latent spaces which produced only 256 tokens thus, no super big image could be taken as input otherwise reconstruction would have degraded drastically. Therefore, the researchers used the concept of a sliding attention window which solves the lower computation limitation of the project. They have also replaced MSE loss with perceptual loss and adversarial loss.

With diffusion models explained as a sequential technique, the typical operation is performed in pixel space. The formulation algorithm executes by adding or removing noise to a tensor of the same size as the original image resulting in slow inference speed and high computational cost. Thus, the paper “High-Resolution Image Synthesis with Latent Diffusion Models” breaks the ice and deals with issues of previous approaches through the use of latent space of powerful pre-trained autoencoders [1]. The researchers used the models that can be interpreted as an equally weighted sequence of denoising autoencoders which can be trained to predict a denoised variant of their input. The cross-attention conditioning mechanism is used to train a large 1.4 billion parameter text image diffusion model. This model consists of the U-Net and the transformer backbone, jointly trained on the publicly available LAION 400M dataset. The resulting model can compose samples from complex text prompts and write user-specific text.

CHAPTER – 3: SYSTEM ANALYSIS

3.1 System Analysis

System analysis is a detailed examination of a system or a proposed project to identify its components, their interrelationships, and how they work together to achieve a particular goal or solve a problem. The primary purpose of system analysis is to improve the efficiency and effectiveness of the system, ensuring that it meets the needs of its users.

3.1.1 Requirement Analysis

Requirement analysis is the systematic process of gathering, documenting, and analyzing user needs and expectations for a system or project. It involves understanding the functionalities, constraints, and objectives to ensure that the resulting solution effectively meets the identified requirements.

3.1.1. i Functional requirement

The functional requirements of the system include

- An actor and text-to-image generator are present.
- User provides a text prompt which leads to a series of other activities followed by visualizing the image finally.
- The system trains the model.

Actors:

User: Interacts with text to image generator.

System: Executes the model and handles processing.

Use cases:

Provide input prompt:

- User gives input text for processing.
- Includes: Validation of input, Generate image, Model, Return generated image
- Extends: Display input error (for invalid input).

Visualize image:

- The user views the generated image.
- Includes: Return generated image.

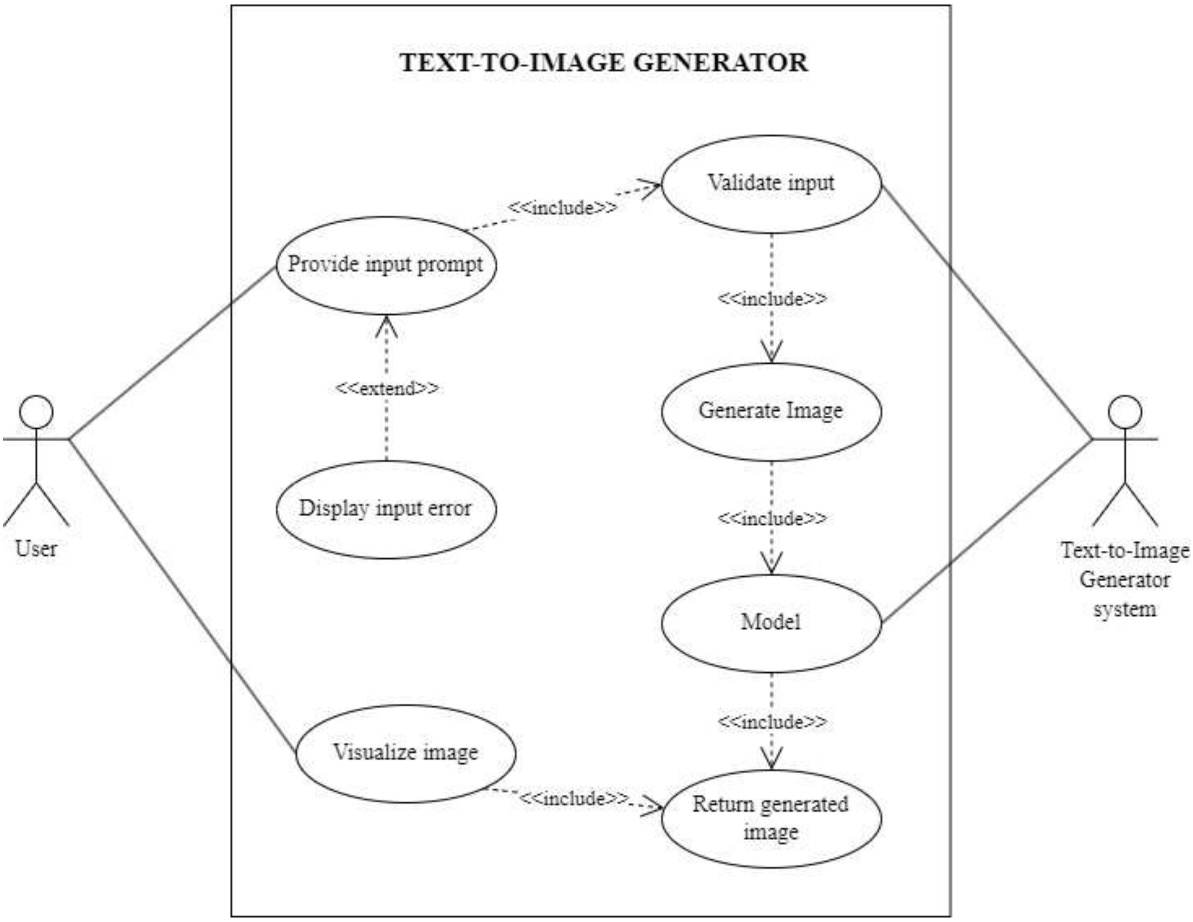

Figure 1: Use Case Diagram

3.1.1. ii Non-functional requirement

The non-functional requirements of the system include:

- **Performance:** The system should be able to generate high-quality images from textual prompts within a reasonable time frame.
- **Reliability:** The model must be able to demonstrate stability by producing consistent and reliable results, minimizing the occurrence of image generation errors or inconsistencies.
- **Usability:** The user interface should be easy to use and understand. It should be intuitive and responsive, allowing users to interact with the system.

3.1.2 Feasibility Analysis

3.1.2.i Technical feasibility

This project is carried out in Visual Studio Code and also in Google Colab and Kaggle Notebooks.

3.1.2.ii Schedule feasibility

With the necessary resources including hardware, software, and personnel, the project can be completed within a three-month timeframe. A Gantt chart in the figure below provides a tentative schedule outlining the project's timeline.

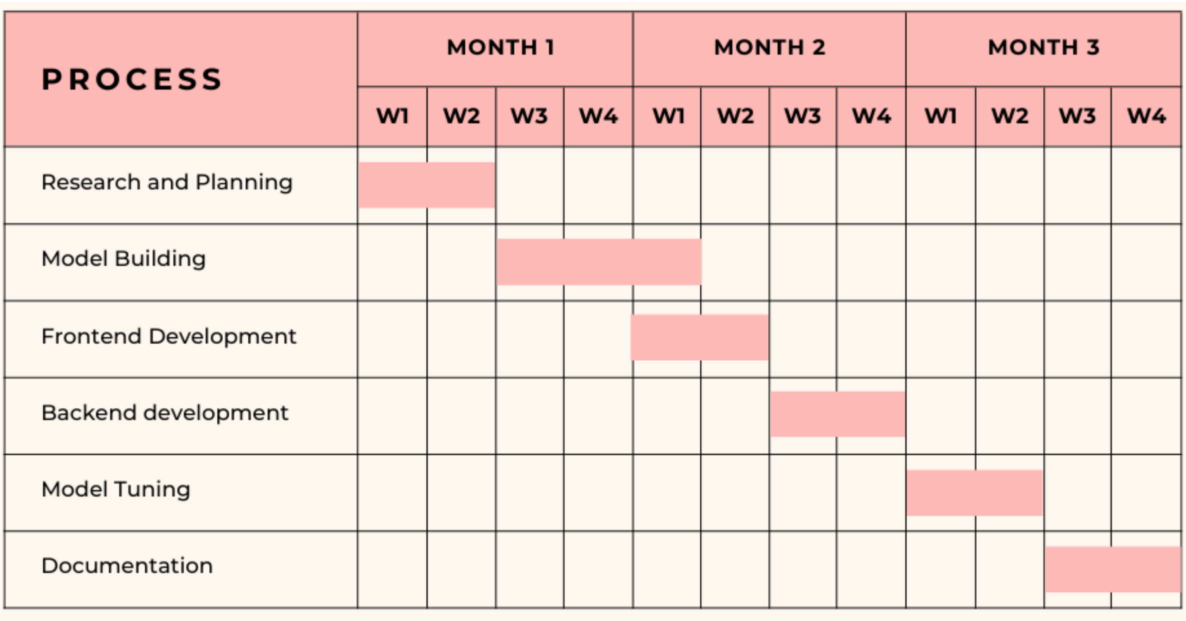

Figure 2: Gantt-Chart of the project

3.1.2. iii Economic feasibility

This project is flexible in terms of economic expenses. All the software tools used for developing the project are free. The project was developed with the help of the personal devices of the team members and no other computational infrastructure was bought, rented or used throughout the project

3.1.3 Analysis

Analysis in a flow diagram refers to the examination and representation of processes, activities, or steps within a system or a project. It involves breaking down a complex system into manageable components and illustrating the sequential flow of actions or data between these components.

3.1.3.i Flow diagram

The flow diagram consists of various software process stages. It is initiated by loading the dataset, followed by data cleaning and data transformation activities. Thereafter, the data is fed into the Model for training. The model's effectiveness is evaluated by validating it against the test dataset. Finally, after the training and testing process, users can generate images by providing text prompts.

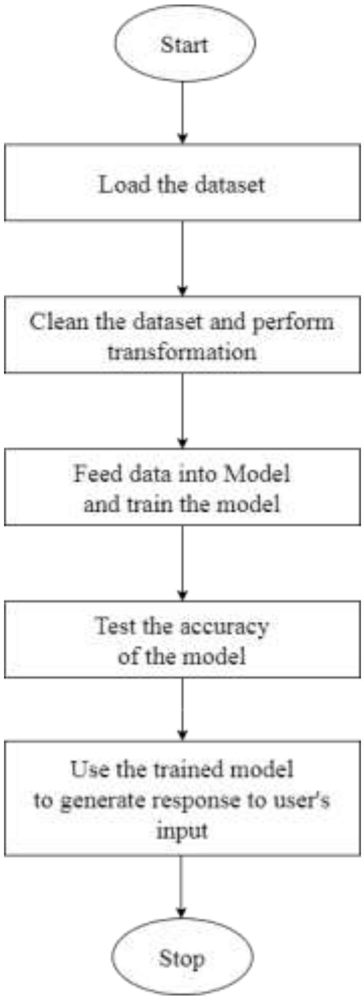

Figure 3: Flow Diagram of the system

CHAPTER – 4: SYSTEM DESIGN

4.1 Design

4.1.1 Sequence diagram

The sequence diagram in the system provides a sequential flow of activities between the user, the system and the model. In the below sequence diagram, the user initiates by entering a text prompt into the system. Then the system validates if the prompt is okay or not and delivers the appropriate message accordingly. If the prompt is valid, the system requests the model to generate the image. The model returns the required generated image to the system which then presents it to the user.

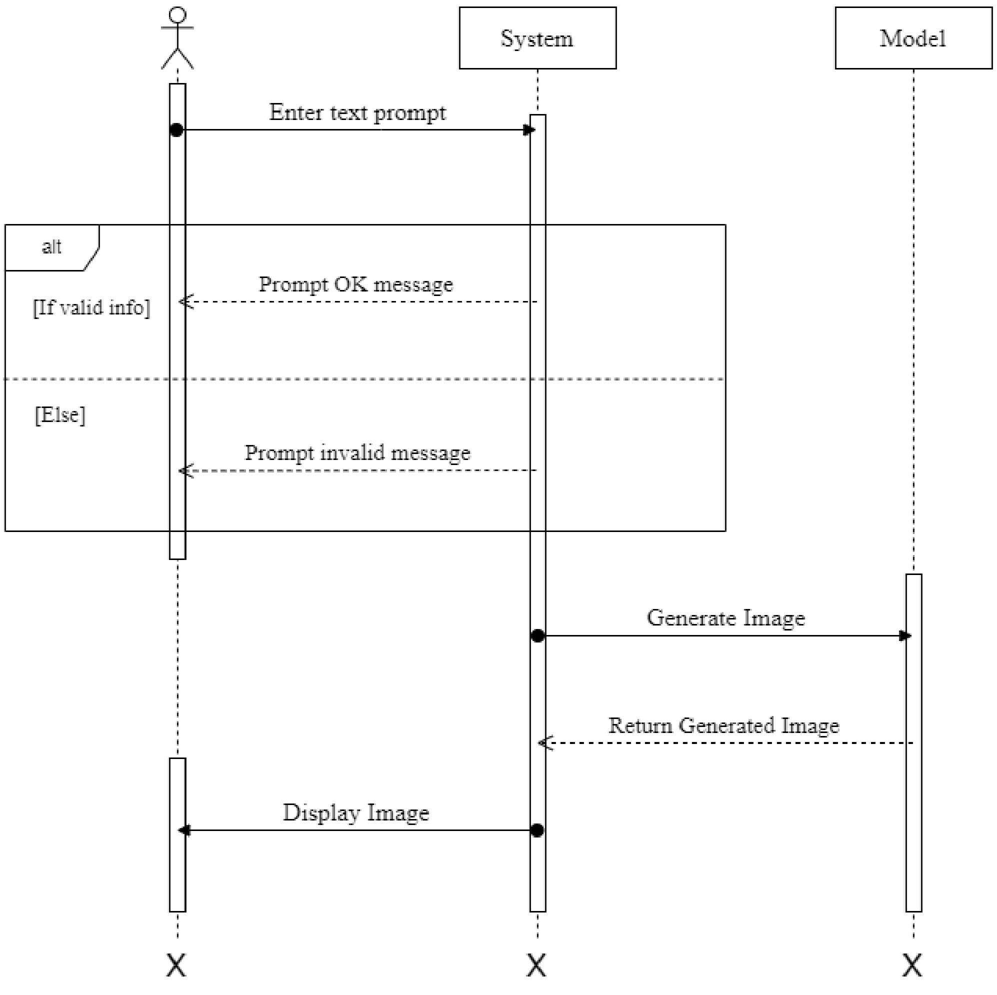

Figure 4: Sequence Diagram

4.1.2 Activity diagram

An activity diagram is the graphical workflow of all the sequential activities that take place inside the system itself. In the below activity diagram, the workflow of a system involving an actor and a text-to-image generator system is shown. The first activity is user input, where the actor enters the necessary prompt into the system. The second activity is validation, where the system checks the user input to ensure that it meets the required standards. The third activity is generating the required image using the model. The fourth activity is returning the generated image to the user for visualizing which is the last activity in the diagram.

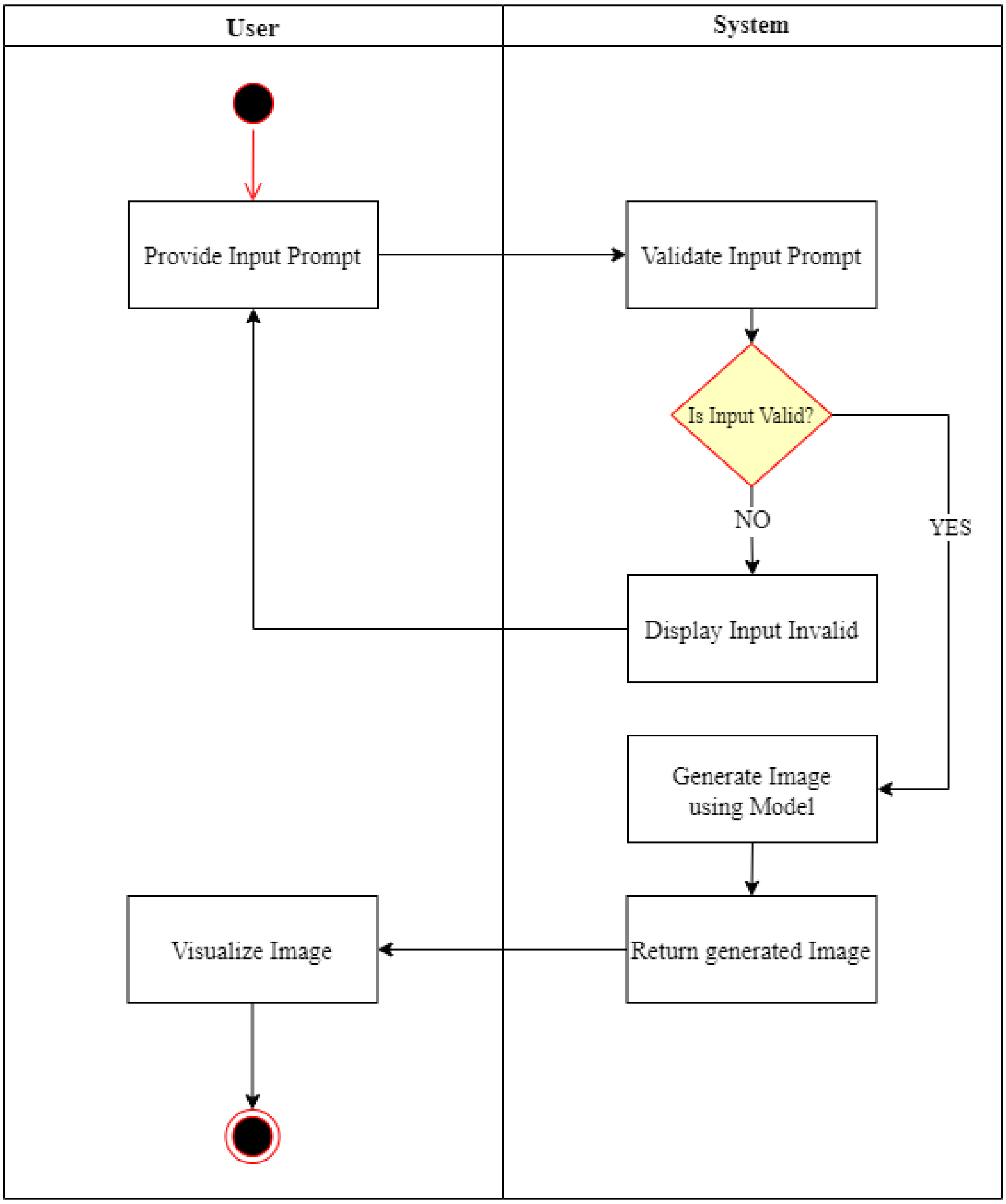

Figure 5: Activity Diagram

4.2 Algorithm Details

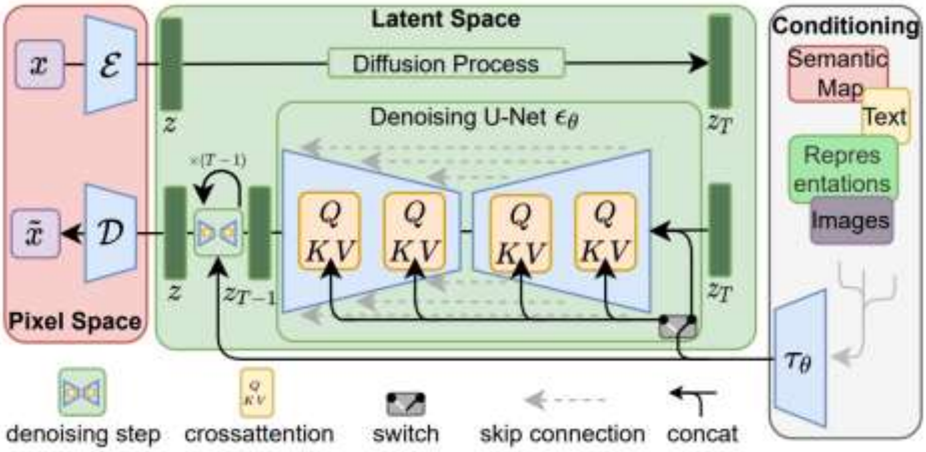

Figure 6: Latent Diffusion Architecture [1]

Figure 6 is the architecture of the Latent Diffusion Model which we are going to implement in our project. The components which make up this architecture are listed below:

i. Auto-encoder:

An auto-encoder is a neural network used for data compression and reconstruction, consisting of an encoder and a decoder. It compresses input into latent space, reconstructs it and aids in training diffusion models on latent space. As proposed in the paper, KL regularized VAE will be used in order to encode the images into latent images during training process and likewise decode the latent images into images for both training and sampling process.

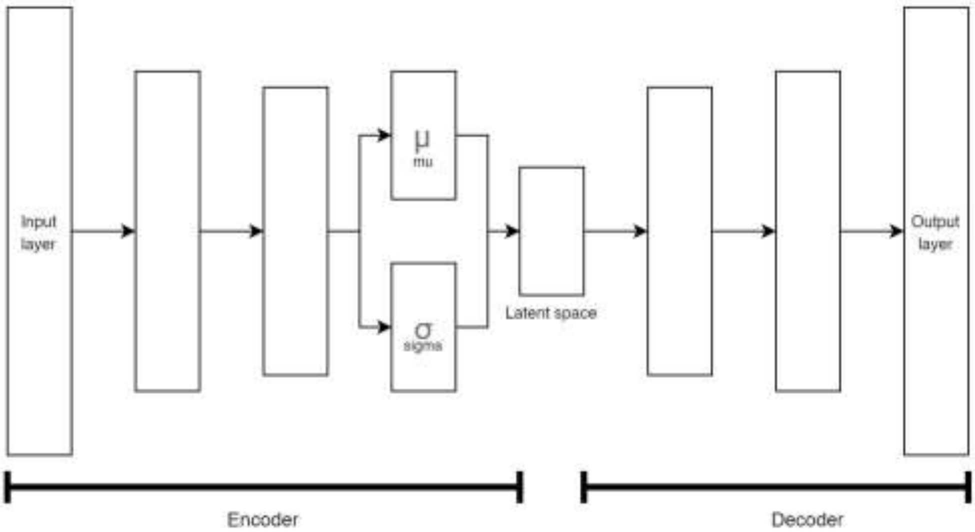

Figure 7: VAE Architecture

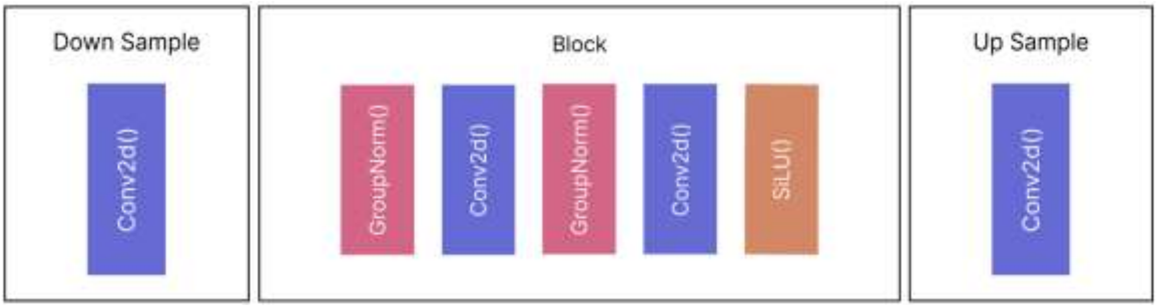

Figure 8: VAE Down Sample, VAE Block, VAE Up Sample (Left-to-right)

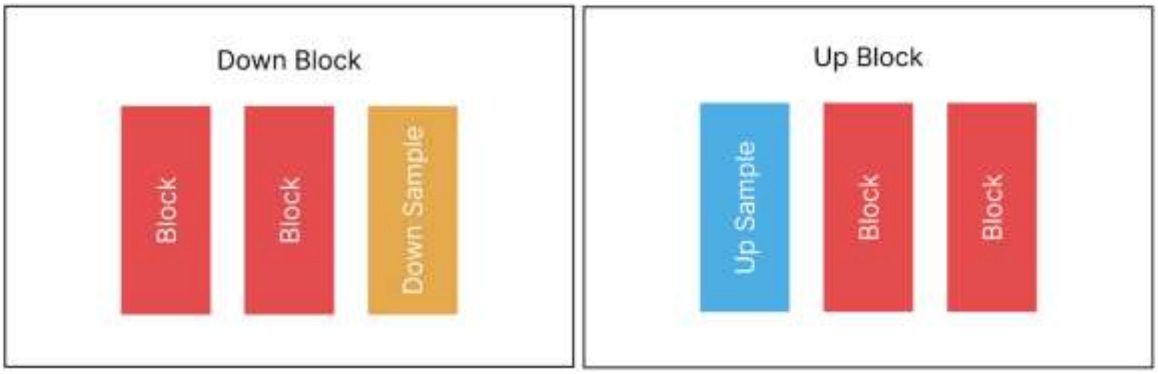

Figure 9: VAE Down and Up Block

ii. Gaussian Noise / Normal Distribution:

The Gaussian noise is added using the formula given below. The formula is simply a Normal Distribution based on mean and variance of the data.

$$q(x_t|x_{t-1}) = N(x_t;\ \sqrt{1-\beta_t}x_{t-1},\ \beta_t I)$$

iii. Conditioning:

Conditioning inputs such as text assist the model to get more knowledge about the image generation tasks which we can achieve by using text embedding. This text embedding, will be attached with time embedding and fed into the UNet.

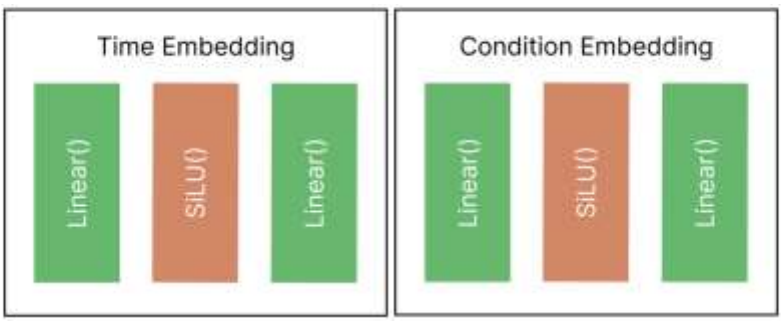

Figure 10: Time Embedding and Condition Embedding

iv. Loss function:

For the loss function, we will be using MSE loss given by the following formula:

$$L_{LDM} := E_{E(x),y,\epsilon\sim N(0,1),t}[\| \epsilon - \epsilon_\theta(z_t, t, \tau_\theta(y)) \|_2^2]$$

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$

v. Denoising U-Net:

Denoising U-Net is a neural network in LDMs to predict noise of the noisy latent image. This U-Net was first introduced for image segmentation tasks but its uses are far more prevalent in other fields. This architecture will be used for predicting noise but there will be some changes in order to fuse it with LDM.

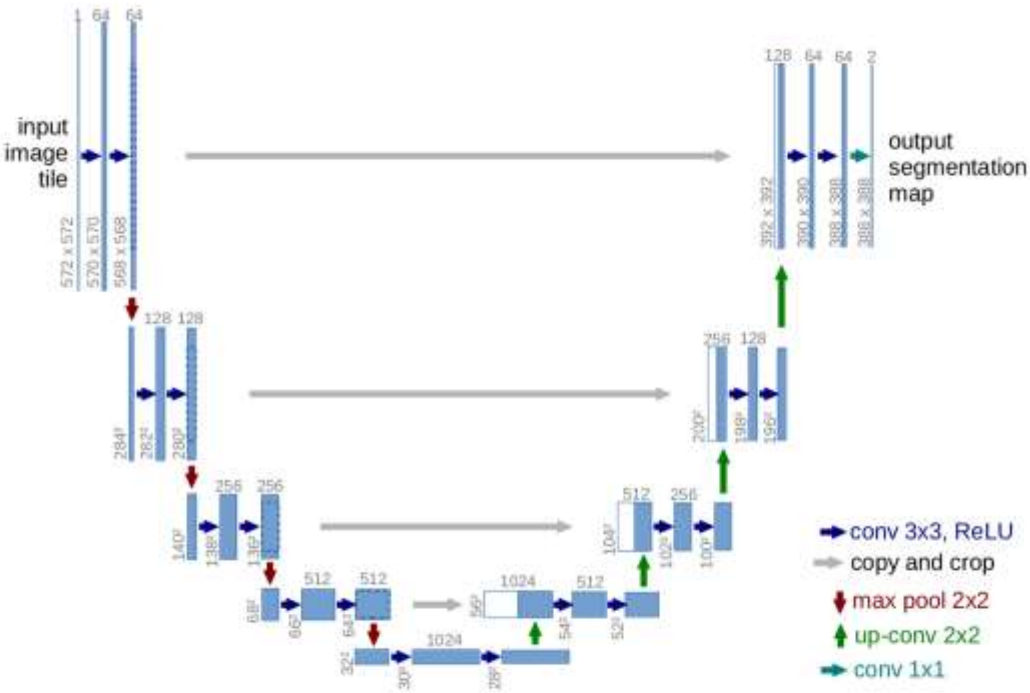

Figure 11: UNet Architecture [4]

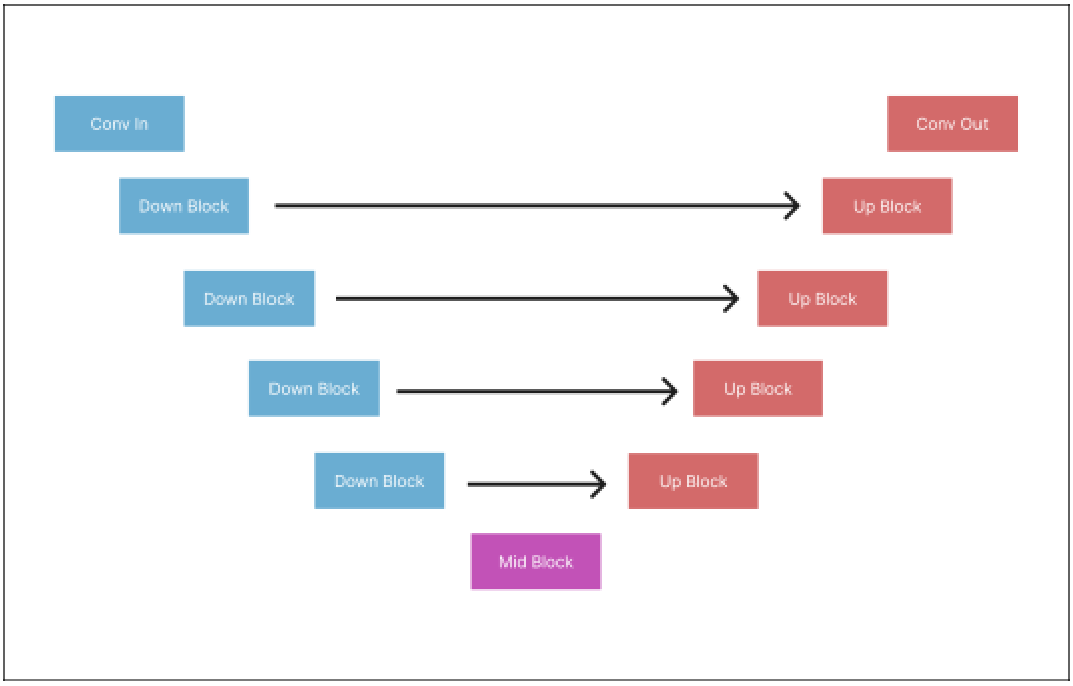

Figure 12: UNet for LDM

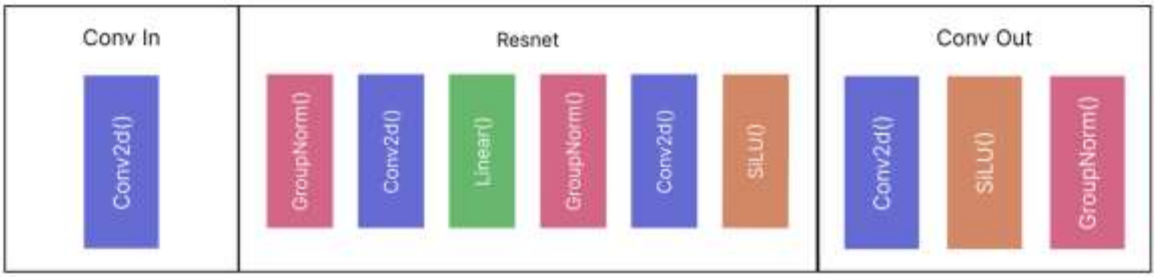

Figure 13: Conv In, Resnet, Conv Out (Left-to-right)

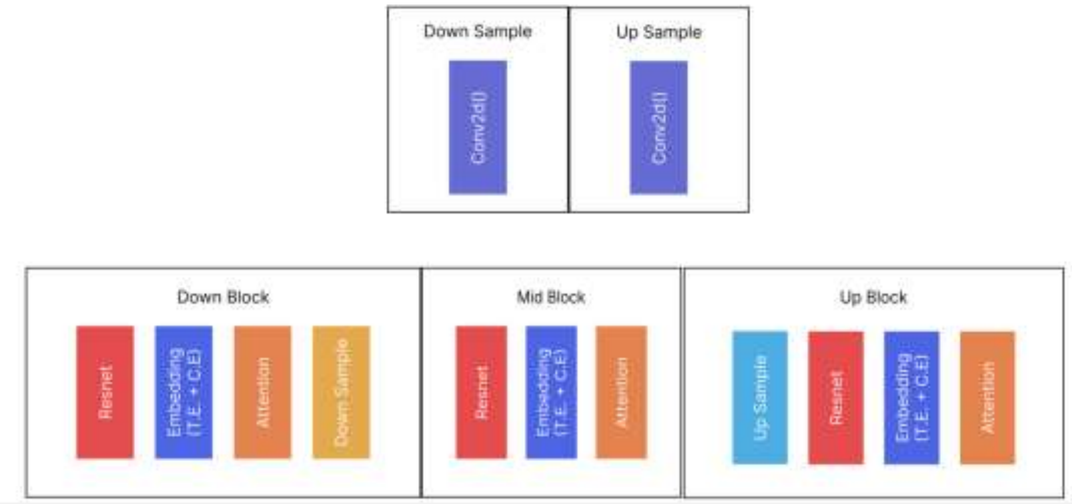

Figure 14: UNet: Down Sample, Up Sample, Down Block, Mid Block, Up Block

Training Algorithm:

Step-1: Start with an input image.

Step-2: Encode the input image into a compressed latent space using an autoencoder.

Step-3: Add Gaussian noise to the latent space to create a noisy latent space.

Step-4: Feed the noised image to a U-Net with added time embedding and condition embedding (text) to get the predicted noise.

Step-5: Remove the predicted noise from the image at timestep (T) and add noise of timestep (T-1)

Step-6: Repeat Step-4 and Step-5 until timestep (T=0).

Algorithm 1 Training

repeat
  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
  $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
until converged

Algorithm 2 Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \ldots, 1$ **do**
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$
end for
return $\mathbf{x}_0$

Figure 15: Training and Sampling Algorithm [3]

Sampling Algorithm:

Step-1: Start with a random Gaussian noise.

Step-2: Feed the noised image to a U-Net with added time embedding and condition embedding (text) to get the predicted noise.

Step-3: Remove the predicted noise from the image at timestep (T) and add noise of timestep (T-1)

Step-4: Repeat Step-4 and Step-5 until a clear image is obtained.

CHAPTER – 5: IMPLEMENTATION AND TESTING

5.1 Implementation

Implementation is part of the action to put all the plans to work using different tools and programming languages.

5.1.1 Tools Used

For the implementation of this project, Python was selected as the primary programming language. In addition to Python's native functionality, several supporting libraries were utilized to facilitate the development process. The following libraries were particularly instrumental in the project's successful completion:

1. PyTorch
2. Diffusers
3. Matplotlib
4. Numpy
5. PIL library
6. Gradio

These libraries helped in achieving the desired functionality and allowed for efficient coding practices.

5.1.2 Implementation Details of Modules

i. app.py

   This module runs the Gradio interface of the program. This launches an interface where the users interact with the program providing input text prompt and viewing the output image. The method is dependent on the functionalities provided by ‘ldm_module.py’.

ii. dataset.py

   a. CustomDataset

      This class, defined in ‘dataset.py’, extends the PyTorch’s dataset class and is designed to handle the dataset of folder images. It takes the path to the image folder as input and provides methods to load and preprocess the image data.

b. NumpyDS

Similar to CustomDataset, NumpyDS is another class in 'dataset.py' that extends the CustomDataset class. It is specialized in handling latent images represented as NumPy arrays.

iii. data_preprocessing.py

a. data_to_latents()

This function takes in 'VAE model', 'data_folder', 'latent_folder' and other data preprocessing parameters. Then it uses 'CustomDataset' to load the dataset and feed it into PyTorch's DataLoader which loads the dataset for preprocessing. In preprocessing, the loaded image data are encoded by the VAE and saves the latents in NumPy (.npy) file extension format in the destination folder given by 'latent_folder'.

b. latent_dataloader()

This function simply takes the latent folder destination and loads the dataset using 'NumpyDS' class and then uses PyTorch's Dataloader to load the data.

iv. ldm_module.py

Here, the functionalities related to input validation and image generation are present. It initializes and manages instances of neural networks, specifically UNet and VAE, and loads their weights. A DDPMSampler with 1000 timesteps is also instantiated for noising the image.

A 'generate_image()' function is present which involves creating random noise, converting text into tensors, passing them through UNet, and predicting noise in reverse order over a specified number of timesteps (1000). Finally after timestep reaches 0, generated image is returned.

v. ldm.ipynb

This Jupyter notebook is used for training purpose. Following are the tasks carried out by the notebook.

- Instantiates Unet, VAE and DDPMscheduler.

- Utilizes the functionality provided by “dataset.py” and “data_preprocessing.py” for data preprocessing.
- Executes a standard PyTorch training loop for 1000 epochs, saving the trained model.
- Plots the loss during training.
- Conducts inference using the trained model, demonstrating the generative capabilities of the text-to-image model.

REFERENCES

[1] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, Bjorn Ommer, “High-Resolution Image Synthesis with Latent Diffusion Models”, *Arxiv*, Apr. 13, 2022. [Online]. Available: https://arxiv.org/pdf/2112.10752.pdf

[2] Patrick Esser, Robin Rombach, Bjorn Ommer, “Taming Transformers for High-Resolution Image Synthesis” *Arxiv*, Dec. 16, 2020. [Online]. Available: https://arxiv.org/pdf/2006.11239.pdf

[3] Jonathan Ho, Ajay Jain and Pieter Abbeel, “Denoising Diffusion Probabilistic Models”, *Arxiv*, Dec. 16, 2020. [Online]. Available: https://arxiv.org/pdf/2006.11239.pdf

[4] Olaf Ronneberger, Philipp Fischer and Thomas Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation”, *Arxiv*, May. 18, 2015. [Online]. Available: https://arxiv.org/pdf/1505.04597.pdf

[5] Alex Nichol and Prafulla Dhariwal, “Improved Denoising Diffusion Probabilistic Models,” Feb. 2021. [Online]. Available: https://arxiv.org/pdf/2102.09672.pdf

[6] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, S. Ganguli, and S. Edu, “Deep Unsupervised Learning using Nonequilibrium Thermodynamics,” Nov. 2015. Available: https://arxiv.org/pdf/1503.03585.pdf