A Project Report on

AI based Art Creator Using NLP

Submitted In Partial Fulfilment of The Requirement for the Award of

The Degree

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

Submitted By

Aditya Kumar	**(1907350130009)**
Akash Chaurasia	**(1907350130010)**
Navneet Sagar	**(1907350130042)**
Shagun Singh	**(1907350130056)**

Under The Supervision of

Mr. Vivek Kumar Jaiswal
(Assistant Professor)

Department Of Information Technology

RAJKIYA ENGINEERING COLLEGE, BIJNOR

(Government Engineering College affiliated to AKTU Lucknow)

Chandpur- 246725

(Batch 2019-2023)

CANDIDATE'S DECLARATION

We hereby declare that the work presented in the report titled '**AI based Art Creator using NLP**' submitted towards the partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Information Technology from Rajkiya Engineering College Bijnor, UP, India, is an authentic record of my own work carried out in the period of session 2022-2023 under the sincere guidance of **Mr. Vivek Kumar Jaiswal**. It is also stated that no earlier submission of the subject matter of the work demonstrated in this report has been made for the award of any other degree in this or any other University/Institute.

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Name:	Name:
Roll No.:	Roll No.:
Date:	Date:
Signature:	Signature:
Name:	Name:
Roll No.:	Roll No.:
Date:	Date:
Signature:	Signature:

The B. Tech (Report) Viva-Voce examination of **Aditya Kumar** (1907350130009), **Akash Chaurasia** (1907350130010), **Navneet Sagar** (1907350130042), **Shagun Singh** (1907350130056) has been held on ………………… and accepted.

External Examiner **Supervisor** **Head of Department**

CERTIFICATE

This is to certify that Project **-AI Based Art Creator Using NLP** which is submitted by Aditya Kumar (1907350130009), Akash Chaurasia (1907350130010), Navneet Sagar (1907350130042), Shagun Singh (1907350130056) in partial fulfilment of the requirement for the award of degree B. Tech in Department of Information Technology of Dr. APJ Abdul Kalam Technical University Lucknow Uttar Pradesh, is a record of the candidate own work carried out by them under my supervision.

The matter embodied in this Project Report is original and has not been submitted for the award of any other degree.

Supervisor's Name:

Signature:

Date:

ACKNOWLEDGEMENT

First and foremost, I would like to express my gratitude to my supervisor **Mr. Vivek Kumar Jaiswal**, Assistant Professor, Department of Information Technology, Rajkiya Engineering College, Bijnor for the useful comments, remarks, and engagement through the learning process of this bachelor's report. I cannot thank them enough for their tremendous support and help. They motivated and encouraged me throughout this work. I consider myself extremely fortunate to have a chance to work under their supervision. Despite their busy schedule, they were always approachable and took their time off to guide me and gave appropriate advice.

I also wish to thank whole heartedly all the faculty members of the Department of Information Technology for the invaluable knowledge they have imparted to me and for teaching the principles most excitingly and enjoyably.

We would also like to thank **Director, Dr. N.K. BADAL, Dr. ISHAN BHARDWAJ**, **HOD (IT)**, and **Asst. Prof. Mr. VIVEK KUMAR JAISWAL** who, instead of their busy schedule, always guided us in the right direction I would like to thank the entire staff member for motivation, guidance and support.

Name: Aditya Kumar	Name: Akash Chaurasia
Roll No.: 1907350130009	Roll No.: 1907350130010
Date:	Date:
Signature:	Signature:
Name: Navneet Sagar	Name: Shagun Singh
Roll No.: 1907350130042	Roll No.: 1907350130056
Date:	Date:
Signature:	Signature:

ABSTRACT

This project using Stable Diffusion Model is an innovative project that aims to bridge the gap between natural language processing and computer vision by generating realistic images from textual descriptions. The project leverages the Stable Diffusion Model, a state-of-the-art generative model that utilizes diffusion processes to create high-quality images.

The primary objective of this project is to develop a robust system capable of translating descriptive textual input into visually coherent and semantically meaningful images. The system employs a two-step process: text embedding and image synthesis. In the text embedding phase, the input text is encoded into a latent space representation, capturing the essential features and context of the textual description. The Stable Diffusion Model is then employed to generate images conditioned on this latent representation.

The Stable Diffusion Model, known for its ability to capture complex data distributions, serves as the core engine for image synthesis. It utilizes a series of diffusion steps to iteratively refine a random noise vector into a realistic image that aligns with the given textual input. The model's stability properties help ensure the preservation of essential visual elements and prevent artifacts or distortions during the generation process.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
NLP	Natural Language Processing
NLU	Natural Language Understanding
NLG	Natural Language Generation
NLI	Natural Language Interaction
GAN	Generative Adversarial Language
CGAN	Conditional Generative Adversarial Language
GPU	Graphics Processing Unit
StackGAN	Stacked Generative Adversarial Language
CNN	Convolutional Neural Network
GLIDE	Guided Language-to-Image Diffusion for Generation and Editing
VAE	Variational Autoencoder
CLIP	Contrastive Language Image Pretraining
UNet	U-Shaped Network Architecture
SGD	Stochastic Gradient Descent
API	Application Programming Interface

CHAPTER 1

INTRODUCTION

Using free-form text to generate or manipulate high-quality images is a challenging task, requiring a grounded learning between visual and textual representations. Manipulating images in an open domain context was first proposed by the seminal Open-Edit, which allowed text prompts to alter an image’s content.

Using Stable Diffusion Model project aims to address the challenge of generating realistic images from textual descriptions. This innovative project merges natural language processing and computer vision techniques to create a system capable of translating text into visually coherent and semantically meaningful images. By leveraging the power of the Stable Diffusion Model, a state-of-the-art generative model, the project endeavors to push the boundaries of AI technology in image synthesis.

The ability to convert text into images has numerous practical applications across various domains, including creative industries, virtual environments, gaming, and simulation. The project seeks to enable designers, content creators, and virtual environment developers to effortlessly generate visual content based on textual prompts, streamlining the creative process and enhancing productivity.

In summary, the AI Based Art Creator Using NLP using Stable Diffusion Model project endeavours to develop a robust system capable of translating textual descriptions into visually coherent and semantically meaningful images. By combining natural language processing with computer vision and harnessing the capabilities of the Stable Diffusion Model, this project pushes the boundaries of AI technology, unlocking new possibilities in image synthesis and creative content generation.

1.1. DEEP LEARNING

Deep learning is a subset of machine learning that uses artificial neural networks with multiple layers, known as deep neural networks, to perform tasks such as image and speech recognition, natural language processing, and decision-making. These neural networks are inspired by the structure and function of the human brain, and are able to learn and improve their performance through the use of large amounts of data and computing power.

There are several types of deep learning networks, including feedforward neural networks, recurrent neural networks, and convolutional neural networks. Each type has its own strengths and is suited for different tasks.

Deep learning has been successfully applied in a wide range of industries, including healthcare, finance, and transportation. In healthcare, for example, deep learning algorithms have been used to analyze medical images and improve the accuracy of disease diagnosis. In finance, deep learning algorithms have been used to detect fraud and predict stock prices.

Deep learning requires large amounts of labeled data and powerful computing resources. This has led to the development of specialized hardware, such as graphics processing units (GPUs), to accelerate the training of deep learning models.

Despite the successes of deep learning, it is not without its limitations. Some of the challenges include interpretability, overfitting, and ensuring that the models are fair and unbiased.

Overall, deep learning has the potential to revolutionize a wide range of industries and improve our ability to analyze and understand data. However, it is important to continue to address the challenges and limitations of the technology in order to fully realize its potential.

1.2. NATURAL LANGUAGE PROCESSING (NLP)

Natural Language Processing (NLP) is a subfield of Artificial Intelligence (AI) that deals with the interaction between computers and human languages. NLP techniques are used to analyze, understand, and generate human languages in a way that computers can process and understand.

One of the main goals of NLP is to enable computers to understand human language as it is spoken or written, and to respond in a way that is natural and appropriate. This involves a number of different techniques, including natural language understanding, natural language generation, and natural language interaction.

Natural Language Understanding (NLU) is the process of extracting meaning from natural language text or speech. This can include tasks such as named entity recognition, sentiment analysis, and topic modeling. NLU is used in a wide range of applications, including chatbots, virtual assistants, and automated customer service systems.

Natural Language Generation (NLG) is the process of creating natural language text or speech from structured data or knowledge. This can include tasks such as summarizing text, generating text from data, and generating responses to natural language input. NLG is used in a wide range of applications, including text generation, text summarization, and natural language question answering.

Natural Language Interaction (NLI) is the process of allowing a computer to interact with a human in natural language. This can include tasks such as natural language dialogue systems, natural language command and control systems, and natural language question answering systems. NLI is used in a wide range of applications, including chatbots, virtual assistants, and automated customer service systems.

There are several approaches to NLP, including rule-based systems, machine learning-based systems, and deep learning-based systems. Rule-based systems rely on a set of predefined rules and patterns to analyze and understand natural language. Machine learning-based systems use statistical models trained on large amounts of data to analyze and understand natural language. Deep learning-based systems use neural networks to analyze and understand natural language.

NLP has a wide range of applications, including natural language search engines, text-to-speech systems, language translation systems, and automated customer service systems. NLP is also used in a wide range of industries, including healthcare, finance, and retail.

In conclusion, NLP is a rapidly growing field that is becoming increasingly important as the amount of natural language data continues to grow. NLP techniques are used to analyze, understand, and generate human languages in a way that computers can process and understand. NLP has a wide range of applications and is used in a wide range of industries.

1.3. ARTIFICIAL INTELLIGENCE (AI)

Artificial Intelligence (AI) is a branch of computer science that deals with the development of intelligent machines that can perform tasks that typically require human intelligence, such as learning, problem-solving, decision-making, and pattern recognition. The goal of AI is to create machines that can think and reason like humans.

There are several different types of AI, including:

1.3.1. Reactive machines

These are the simplest form of AI, and they can only react to their environment. They do not have the ability to remember past events or learn from them. Examples include chess-playing computers and self-driving cars.

1.3.2. Limited memory

These machines have the ability to remember past events, but they cannot use this information to make decisions or predictions. Examples include robots that can navigate around obstacles and drones that can follow a pre-determined flight path.

1.3.3. Theory of mind

These machines have the ability to understand and simulate human emotions and mental states. They can use this information to make decisions and predictions. Examples include robots that can interact with humans in a natural and human-like way.

1.3.4. Self-aware

These are the most advanced form of AI, and they have the ability to understand their own existence and consciousness. They can also understand and simulate the consciousness of others. Examples include robots that can understand and respond to human emotions and mental states.

AI technology is used in a wide range of applications, including healthcare, finance, transportation, and manufacturing. For example, AI-powered robots can assist in surgeries, AI-powered financial trading algorithms can make predictions about stock prices, and AI-powered self-driving cars can reduce the number of accidents caused by human error.

However, there are also concerns about the potential negative consequences of AI, such as job displacement, privacy violations, and the possibility of AI-controlled machines causing harm to humans.

Overall, AI is a rapidly-evolving field with the potential to revolutionize many industries and improve human lives in a wide range of ways. However, it is important to monitor and regulate the development and use of AI to ensure that the benefits outweigh the risks.

CHAPTER 2

MODELS

2.1 StackGAN

StackGAN is a generative adversarial network (GAN) model that is designed to generate high-quality images from textual descriptions. It was proposed in 2017 by researchers from the University of California, Berkeley and Adobe Research.

The basic idea behind StackGAN is to use a two-stage approach to generate images from text. In the first stage, a text encoder takes the textual description as input and generates a low-resolution image. This low-resolution image is then fed into a conditional GAN (cGAN) along with the textual description. The cGAN generates a high-resolution image that matches both the textual description and the low-resolution image.

The novelty of StackGAN lies in its ability to generate images with fine-grained details that are faithful to the textual description. This is achieved by stacking multiple GANs, each trained to generate images with progressively higher resolutions. The final stage of the StackGAN model generates images with a resolution of 256x256 pixels.

Here is a brief overview of the StackGAN architecture:

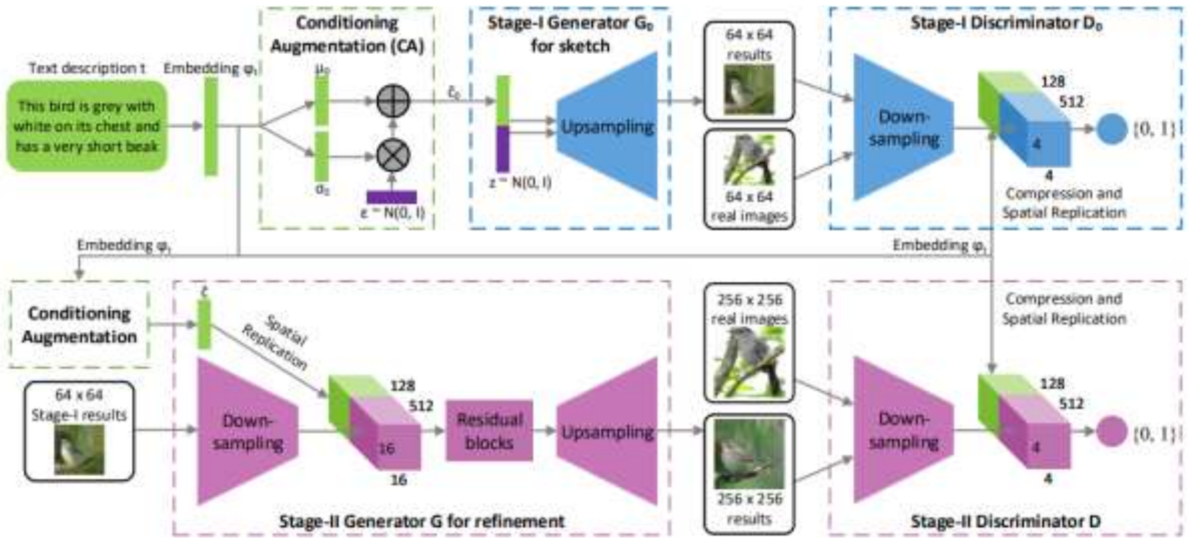


Fig. 2.1- StackGAN architecture [21]

Text encoder: Takes the textual description as input and generates a 128-dimensional feature vector.

Stage-1 GAN: Takes the 128-dimensional feature vector as input and generates a 64x64 low-resolution image.

Stage-2 GAN: Takes the low-resolution image and the textual description as input, and generates a 256x256 high-resolution image.

To train the StackGAN model, the researchers used a combination of adversarial loss, perceptual loss, and text embedding loss. The adversarial loss ensures that the generated images are realistic, while the perceptual loss ensures that the generated images are visually similar to the real images. The text embedding loss encourages the model to generate images that match the textual description.

Overall, StackGAN is a powerful model that can generate high-quality images from textual descriptions. It has a wide range of applications, including image editing, virtual try-on, and visual storytelling.

2.2. GLIDE

The Glide model is a type of generative model that is based on deep neural networks. Specifically, it is a variant of the StyleGAN2 architecture, which was introduced by researchers at NVIDIA in 2019. The StyleGAN2 architecture is designed to generate high-quality, diverse images with realistic details and textures.

The Glide model takes a similar approach to image generation as the StyleGAN2 architecture, but with some modifications that make it more efficient and scalable. The key idea behind the Glide model is to use a generator network to synthesize images from a random noise vector, and a discriminator network to distinguish between real and generated images. During training, the generator network learns to produce images that are similar to the ones in the training dataset, while the discriminator network learns to distinguish between real and generated images.

The generator network in the Glide model is based on a series of convolutional layers that transform the input noise vector into a high-resolution image. The architecture of the generator network is designed to produce images with a wide range of styles and features, which makes the model highly flexible and versatile. The discriminator network, on the other hand, is designed to recognize the features and patterns that are specific to real images, and to distinguish them from generated images.

One of the key features of the Glide model is its ability to control the style and attributes of the generated images. This is achieved through the use of style vectors, which are used to modulate the output of the generator network. By varying the style vectors, the model can generate images with different styles, such as different hair colors or facial expressions.

In practice, using the Glide model for image generation involves several steps. First, you need to prepare a large dataset of images to use for training the model. This dataset should be diverse and representative of the types of images you want the model to generate. You also need to preprocess the images to ensure they are all the same size and format.

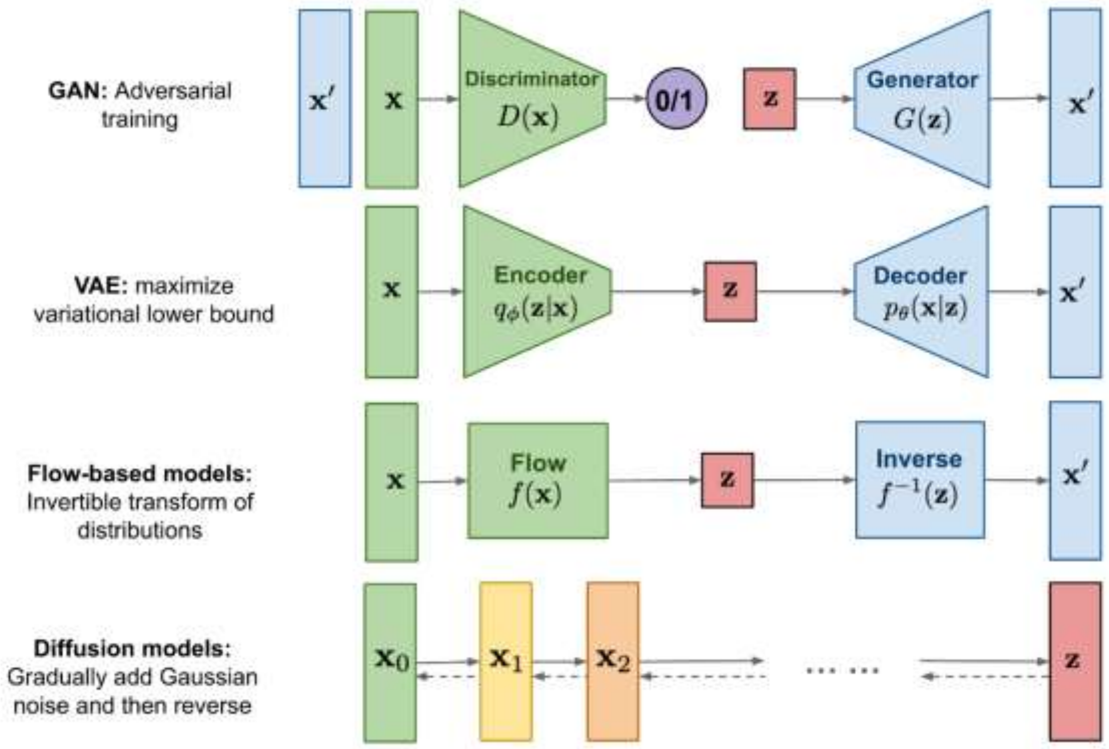
Next, you can train the Glide model using a deep learning framework such as TensorFlow or PyTorch. During training, the model learns to generate new images that are similar to the ones in your training dataset. This process can take a long time, and may require specialized hardware such as a high-end GPU.Once the model is trained, you can use it to generate new images by feeding it random noise vectors. The model will use these vectors to generate new images that are similar to the ones in your training dataset. You can also control the style and attributes of the generated images by varying the style vectors.

2.3. STABLE DIFFUSION

The Stable Diffusion Model is a variant of diffusion models that has gained popularity in the field of generative modelling. It leverages the principles of diffusion processes to generate high-quality and diverse samples from complex distributions.

Diffusion models are a class of generative models that aim to learn the data distribution by modelling the process of iteratively applying a sequence of diffusion steps. The diffusion process starts from a simple initial distribution (e.g., a noise distribution) and gradually transforms it to the target data distribution through a series of diffusion steps. Each diffusion step involves a diffusion process that evolves the current state towards the target distribution by injecting noise.

The Stable Diffusion Model introduces stability regularization into the diffusion process, which helps to improve the stability and convergence properties of the model. Stability regularization encourages the diffusion process to maintain important features and structures of the data distribution during the progression of diffusion steps. This regularization prevents the diffusion process from diverging or collapsing to low-quality samples.



GAN: Adversarial training
$\mathbf{x}'$
$\mathbf{x}$
Discriminator $D(\mathbf{x})$
0/1
$\mathbf{z}$
Generator $G(\mathbf{z})$
$\mathbf{x}'$
VAE: maximize variational lower bound
$\mathbf{x}$
Encoder $q_\phi(\mathbf{z}|\mathbf{x})$
$\mathbf{z}$
Decoder $p_\theta(\mathbf{x}|\mathbf{z})$
$\mathbf{x}'$
Flow-based models: Invertible transform of distributions
$\mathbf{x}$
Flow $f(\mathbf{x})$
$\mathbf{z}$
Inverse $f^{-1}(\mathbf{z})$
$\mathbf{x}'$
Diffusion models: Gradually add Gaussian noise and then reverse
$\mathbf{x}_0$
$\mathbf{x}_1$
$\mathbf{x}_2$
... ...
$\mathbf{z}$

Fig. 2.2- Comparison of various models [16]

CHAPTER 3

METHODOLOGY

3.1. STABLE DIFFUSION MODEL

The Stable Diffusion Model is a type of probabilistic model used in machine learning and statistical inference to describe how information diffuses through a network or system. This model is particularly useful for analyzing large-scale social networks or other systems in which information spreads from one individual or node to others.

The Stable Diffusion Model assumes that each individual or node in the network has a "state" or "opinion" about a particular topic or idea, which can either be positive or negative. At each time step, an individual's state can change depending on the states of its neighbors and the diffusion parameter of the network. The diffusion parameter determines how quickly information spreads through the network and can be estimated from the data.

The Stable Diffusion Model is based on a stochastic differential equation that describes how the state of each individual changes over time. The equation is:

dX_t = α(θ-X_t)dt + σdW_t

where X_t is the state of the individual at time t, α is the diffusion parameter, θ is the "resting point" of the system (i.e., the state that the system tends to converge to over time), σ is the diffusion constant (i.e., the noise level of the system), and dW_t is a Wiener process (i.e., a random process that represents the unpredictable fluctuations in the system).

The Stable Diffusion Model can be fit to data using maximum likelihood estimation or Bayesian inference. Once the model is fit, it can be used to predict how information will spread through the network over time, and to identify the most influential individuals or nodes in the network.

3.1.1. STABLE DIFFUSION

The basic idea behind diffusion models is rather simple. They take the input image $\mathbf{x}0$ and gradually add Gaussian noise to it through a series of T steps. We will call this the forward process. Notably, this is unrelated to the forward pass of a neural network. If you'd like, this part is necessary to generate the targets for our neural network (the image after applying $t<T$ noise steps).

Afterward, a neural network is trained to recover the original data by reversing the noising process. By being able to model the reverse process, we can generate new data. This is the so-called reverse diffusion process or, in general, the sampling process of a generative model.

$q(\mathbf{x}t|\mathbf{x}t-1)=\mathrm{N}(\mathbf{x}t;\boldsymbol{\mu}t=1-\beta t\mathbf{x}t-1,\mathbf{\Sigma}t=\beta t\mathbf{I})$

Thus, we can go in a closed form from the input data $\mathbf{x}0$ to $\mathbf{x}T$ in a tractable way. Mathematically, this is the posterior probability and is defined as:

$q(\mathbf{x}1{:}T|\mathbf{x}0)=t=1\prod Tq(\mathbf{x}t|\mathbf{x}t-1)$

3.2 CLIP

CLIP (Contrastive Language-Image Pre-training) is a training procedure unlike common practices in the vision community. For a period of time, the capabilities of model/training methods are benchmarked on the ImageNet dataset that spans 1000 classes. We train on a subset of ImageNet, and test it on a different subset to measure how well a model generalises. While straightforward, this convention overlooks the exponentially scaling image collections on the internet and the potential benefits it could bring; CLIP, indeed, shows that it is a LOT we are missing out on.

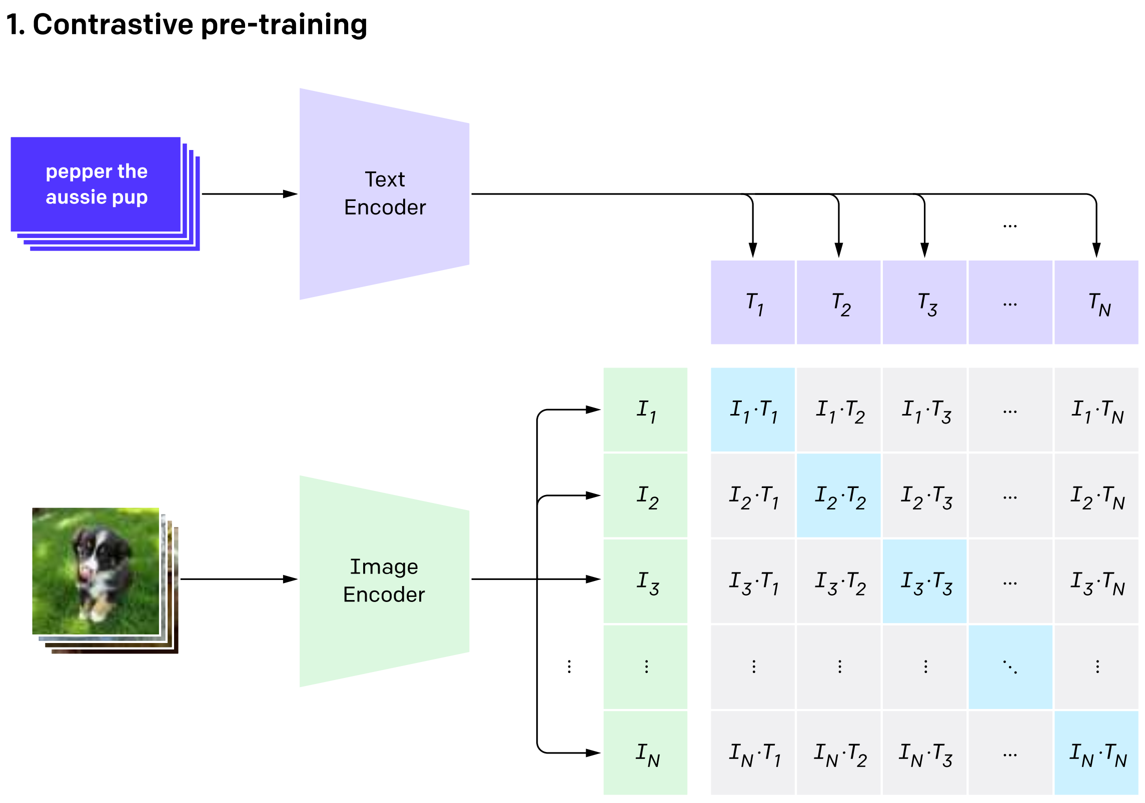

1. Contrastive pre-training
pepper the aussie pup
Text Encoder
Image Encoder
T_1 T_2 T_3 ... T_N
I_1 I_2 I_3 ⋮ I_N
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
$I_2 \cdot T_1$ $I_2 \cdot T_2$ $I_2 \cdot T_3$ ... $I_2 \cdot T_N$
$I_3 \cdot T_1$ $I_3 \cdot T_2$ $I_3 \cdot T_3$ ... $I_3 \cdot T_N$
$I_N \cdot T_1$ $I_N \cdot T_2$ $I_N \cdot T_3$ ... $I_N \cdot T_N$

Fig. 3.1 Pretraining of text encoder and Image Encoder[2]

Besides a large set of images retrieved online, CLIP also utilises texts, a somewhat unorthodox setting at the time of its proposal. Contrary to traditional models that uses an image encoder (e.g., CNN) and a classifier (e.g., fully-connected network) CLIP jointly trains an image encoder and a text encoder to encourage a close embedding space between ones that form a pair via contrastive learning.

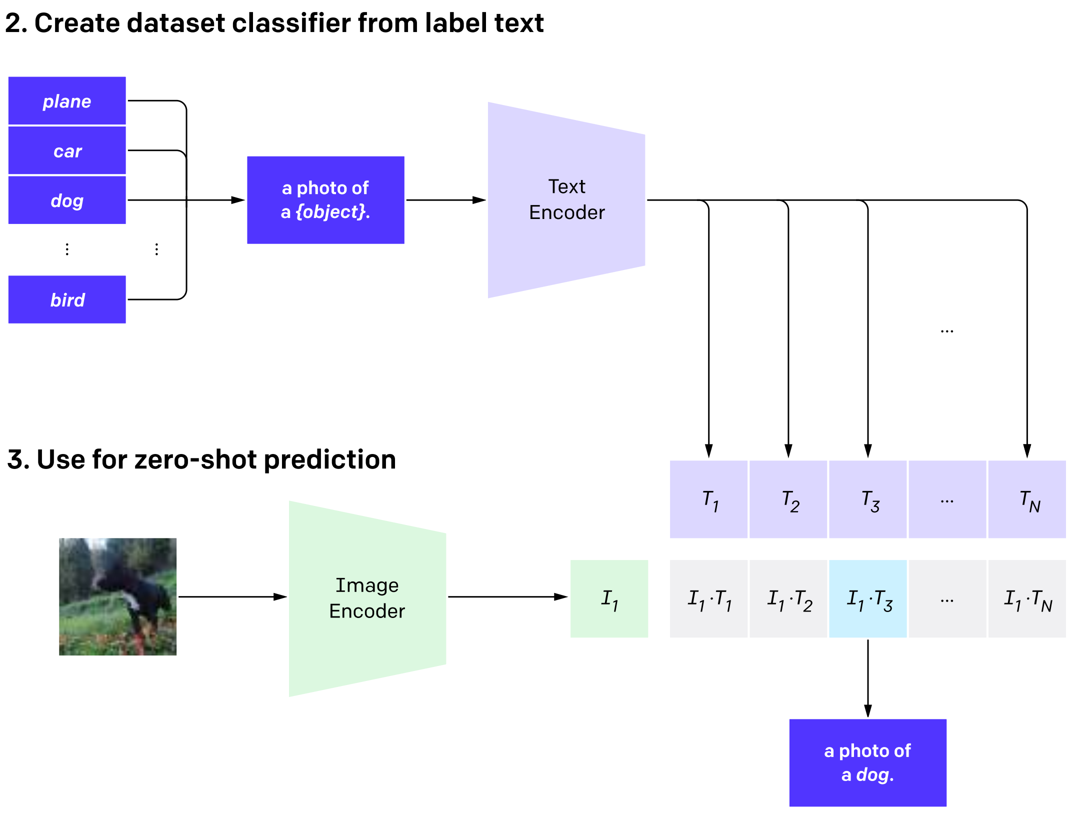

2. Create dataset classifier from label text
plane
car
dog
bird
a photo of a {object}.
Text Encoder
3. Use for zero-shot prediction
Image Encoder
I_1
T_1 T_2 T_3 ... T_N
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
a photo of a dog.

Fig. 3.2- Encode the images and text snippets[2]

Successful training will then lead to texts and images referring to the same object within close proximity. For example, the embedding of the sentence "a photo of a puppy" will be nearby an actual canine image. During inference on a finite set of classes (Figure 1.2), we can simply put the class names into a sentence and find the text embedding closest to the query image. Because CLIP is trained on an extremely vast quantity of 400 million image-text pairs, it has almost become "zero-shot". Any class you can think of is almost guaranteed to be part of this enormous dataset; you would not need any retraining, for any classes, with any additional data.

3.2.1.1 How can we use CLIP?

Despite its simplicity, most of us cannot access hardware to process a 400 million image-text dataset nor the dataset itself. But owing to its zero-shot capability, a pre-trained checkpoint is universal to all subsequent classification tasks. OpenAI has published its code for training and checkpoints on their GitHub:

3.2.1.2 Beyond Simple Classification

Lots of datasets beyond images also contain text labels. Accurate matching of texts and images can therefore be leveraged to other domains. Below we list a couple of interesting extensions of CLIP:

- **Soundify**

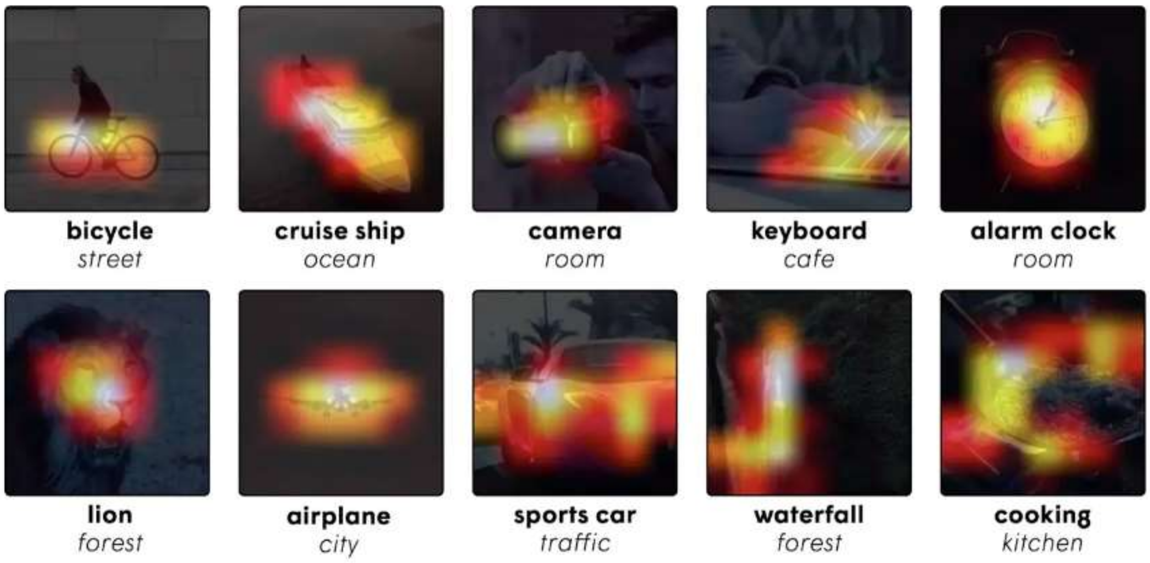


Fig. 3.3- Soundify Result [2]

Lin et al. attempt to alleviate the time consumption of overlaying sound on videos during video editing by matching the closest sound label to the video itself through CLIP.

- **CLIP-Forge**

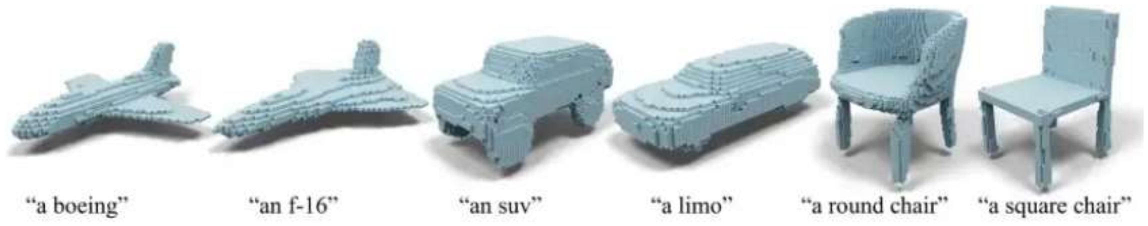


Fig. 3.4- CLIP-Forge Result [2]

CLIP-Forge matches texts with shape renderings to achieve zero-shot 3D shape generation using an unlabelled dataset of 3D objects.

3.3. Prior Model

There is a long history of debate about how much of human knowledge is innate and how much is learned from experience/data. This is also known as the nature vs. nurture debate (See for example some of Gary Markus' Research Paper for more details on this). Traditional Bayesian methods explicitly model these two components with a prior component that can be derived from existing knowledge and a likelihood that is learned from the data.

The recent rapid advances in deep learning continue to demonstrate the significance of end-to-end training with no apriori knowledge (such as domain-aware feature engineering) in many computers vision and NLP tasks. However, when models require reasoning or need to do forward prediction, or operate in low data regime, most AI researchers and practitioners will agree that incorporating prior knowledge, along with end-to-end training can introduce better inductive bias, beyond what is provided in the training data. This is especially true in complex domains such as healthcare where precision of the systems are central, and the combinatorics required for generalization is large.

Despite the general agreement that innate structure and learned knowledge need to be combined, there is no simple approach to incorporate this innate structure into learning systems. As a scientific community, we are just starting to research approaches to incorporate prior into deep learning models. In fact, not surprisingly, there is little consensus on whether priors needs to be hard-coded into the model or can be learned through, for example, using a reward.

The prior is, generally speaking, a probability distribution that expresses one's beliefs about a quantity before some evidence is taken into account. If we restrict ourselves to an ML model, the prior can be thought as of the distribution that is imputed before the model starts to see any data. However, a more holistic view into the process should consider not only the model training, but the design of the system as a whole, including the choice of model and the data. Therefore, the prior includes anything from the choice of the algorithm itself to the way that the data is labeled (this is in fact related to what Markus calls innate algorithms, representational formats, and knowledge).

Therefore, a perhaps not so obvious way to inject prior knowledge and beliefs into the process is through the training data selection. Indeed, by selecting and labeling training samples for any supervised learning model, we encode knowledge into the learning model. The data labelling process might be trivial in some domains, but will require a good deal of domain knowledge in others. For example, to label cats and dogs we need to understand the difference between these two animals, which might seem very “common sense”, but to label medical images as “cancer” or “not cancer”, we need deep medical expertise.

As yet another example, an interesting but negative case of injecting “prior belief” in the data selection/labelling process is related to some well-known examples of algorithmic bias. If the belief of whoever is creating the predictive model is that people of a certain background are more likely to commit a crime, this belief is likely to be fed into the training and testing process by selecting and labeling a dataset that confirms that prior. In fact, most existing datasets, like collections of written texts, have priors/biases that will affect the learned model.

In recent approach, we presented an approach to incorporate prior knowledge into DL systems by using synthetic data. While we presented this approach for a particular application (medical diagnosis), I believe this has broader implications that can be used in many other domains. To be clear, synthetic data has been used to some extent in other domains. For example, self-driving researchers have used synthetic data from the Grand Theft Auto videogame to train machine learning systems. However, our approach follows a flexible and reproducible process that involves the following steps:

Use an existing rule-based expert system to generate data

1. Inject noise to the generated data to make learned system more resilient
2. Combine synthetic data from real-world data
3. Train a deep learning system

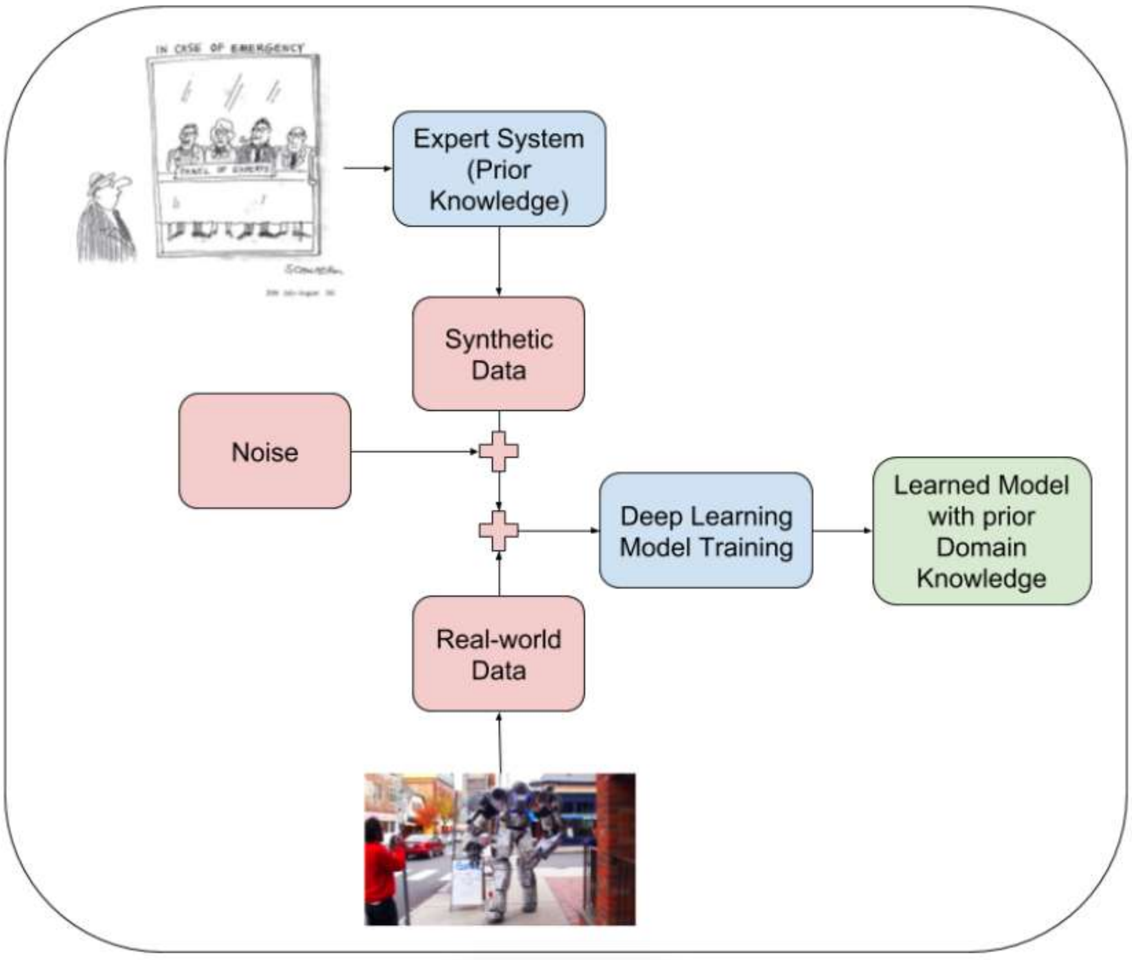

IN CASE OF EMERGENCY
PANEL OF EXPERTS
Expert System (Prior Knowledge)
Synthetic Data
Noise
Deep Learning Model Training
Learned Model with prior Domain Knowledge
Real-world Data

Fig. 3.5- Prior working [6]

The process of combining synthetic data with noise plus real world data to train a deep learning model has several advantages:

1. It allows to introduce innate domain-specific knowledge

2. It allows to control the balance of innate vs. real-world learned knowledge by simply controlling the balance between synthetic and real-world data.

3. Adding noise to the synthetic data allows us to learn a system that is more robust to noisy data in inference time

4. Introducing real-world data means that we can have an extensible continuous learning system that is better over time than the expert system specified by the domain experts.

CHAPTER 4
MECHANISM

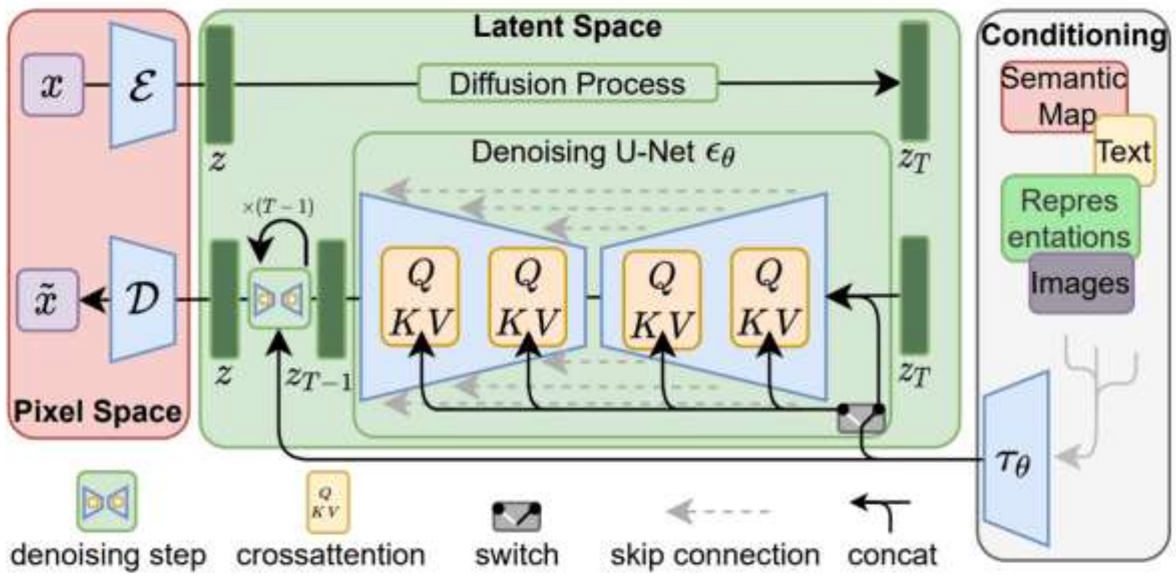


Fig. 4.1- Latent Diffusion Model [11]

4.1 Decoder Diffusion Model

The Decoder Diffusion model is a variant of the Transformer architecture used in natural language processing tasks such as language translation and text summarization. The model uses a diffusion process to spread information throughout the decoder, allowing it to make more informed predictions.

The decoder in the model is composed of multiple layers, each of which is divided into two sublayers: a multi-head self-attention mechanism and a position-wise feed-forward network. The diffusion process is applied between these sublayers, where the output of the self-attention mechanism is used as input to the feed-forward network in the next layer.

This allows the model to gradually refine its predictions as it processes the input, rather than relying solely on the information from the encoder or the final output of the self-attention mechanism. This has been shown to improve the model's ability to handle long-term dependencies and make more accurate predictions.

Overall, the decoder diffusion model is a valuable addition to the transformer-based architectures, and it has been shown to improve the performance on a wide range of NLP tasks.

4.1.1. Denoising Diffusion Model

The idea of denoising diffusion model has been around for a long time. It has its roots in Diffusion Maps concept which is one of the dimensionality reduction techniques used in Machine Learning literature. It also borrows concepts from the probabilistic methods such as *Markov Chains* which has been used in many applications. The original Denoising Diffusion method was proposed in *Sohl-Dickstein et al.* [1].

A denoising diffusion modeling is a two step process: the forward diffusion process and the reverse process or the reconstruction. In the forward diffusion process, gaussian noise is introduced successively until the data becomes all noise. The reverse/ reconstruction process undoes the noise by learning the conditional probability densities using a neural network model.

4.1.2. Reverse diffusion

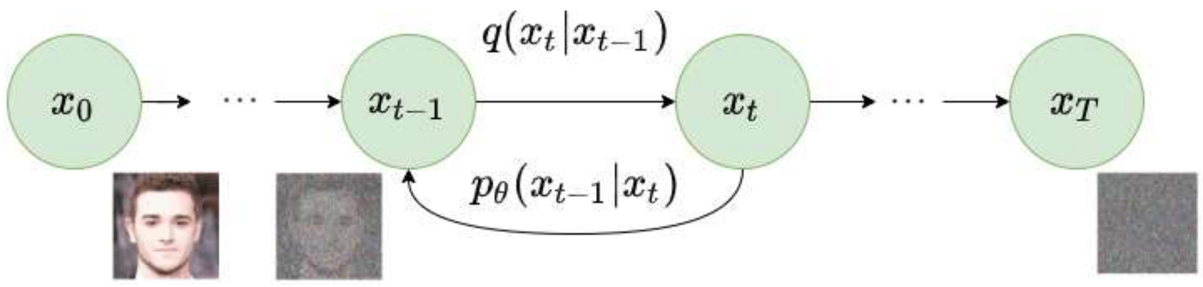


Fig. 4.2- Reverse Diffusion [6]

4.1.3. Forward Process

We can formally define the forward diffusion process as a *Markov Chain* and therefore, unlike an encoder in the VAEs it doesn't require a training. Starting with the initial data point , we add Gaussian noise for ***T*** successive steps, and obtain a set of noisy samples. The prediction of probability density at time ***t*** is only dependent on the immediate predecessor at time ***t-1*** and therefore, the conditional probability density can be computed as follows:

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

The complete distribution of the whole process can then be computed as follows:

$$q(\boldsymbol{x}_{0:T}|\boldsymbol{x}_0) = q(\boldsymbol{x}_0)\prod_{t=1}^{T} q(\boldsymbol{x}_t|x_{t-1})$$

Here, the mean and variance of the density function depends on a parameter βτ, which is a hyper parameter whose value can either be taken as a constant throughout the process or can be gradually changed in the successive steps. For a differential parameter value assignment, there can be range of function that can be used to model the behavior (e.g. sigmoid, tanh, linear etc.

The above derivation is enough to predict the successive states, however, if we would like to sample at any given time interval ***t*** without going through all the intermediary steps, therefore, allowing an efficient implementation, then we can re-formulate the above equation by substituting the hyper-parameter as ατ = 1 — βτ**.** The reformulation of the above then becomes:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}\left(\boldsymbol{x}_t; \sqrt{\overline{\alpha}_t}\boldsymbol{x}_0, (1-\overline{\alpha}_t)\boldsymbol{I}\right)$$

In order to produce samples at a time step ***t*** with probability density estimation available at time step ***t-1,*** we can employ another concept from thermodynamics called, '*Langevin dynamics'*. According to *stochastic gradient Langevin dynamics* we can sample the new states of the system only by the gradient of density function in a *Markov Chain* updates. The sampling of a new data point at time ***t*** for a step size **ε** based on previous point at time ***t-1*** can then be computed as follows:

$$\boldsymbol{x}_t = \boldsymbol{x}_{t-1} + \frac{\epsilon}{2}\nabla_x p(\boldsymbol{x}_{t-1}) + \sqrt{\epsilon}\boldsymbol{z}_i \qquad \boldsymbol{z}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{1})$$

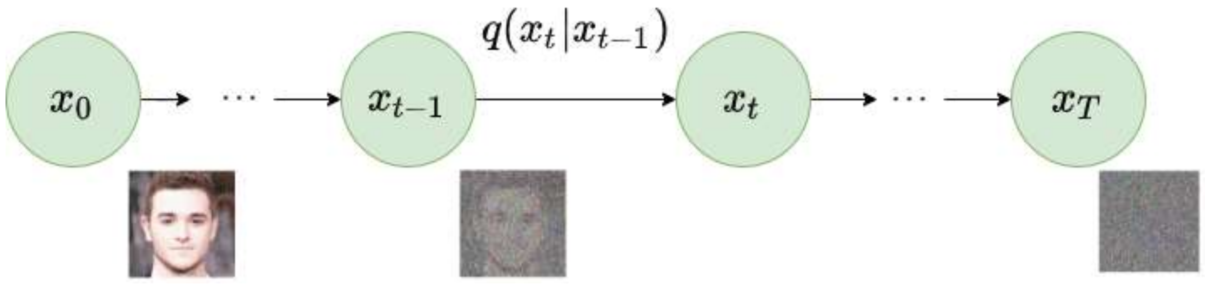


Fig. 4.3- Forward Diffusion[5]

4.2 VARIANCE SCHEDULE

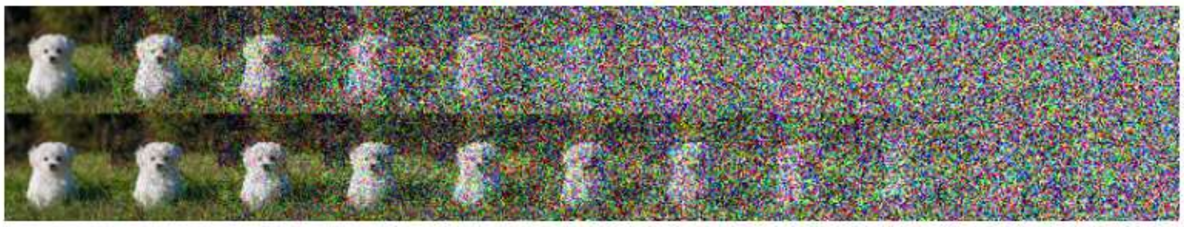


Fig. 4.4- Variance Scheduling [22]

Reconstruction

The reverse process requires the estimation of probability density at an earlier time step given the current state of the system. This means estimating the **q**(χτ-1 | χτ) when *t*=***T*** and thereby generating data sample from isotropic Gaussian noise. However, unlike the forward process, the estimation of previous state from the current state requires the knowledge of all the previous gradients which we can't obtain without having a learning model that can predict such estimates. Therefore, we shall have to train a neural network model that estimates the **ρθ**(χτ-1|χτ) based on learned weights **θ** and the current state at time ***t***. This can be estimated as follows:

$$\boldsymbol{p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))}$$

$$\boldsymbol{p_\theta(x_{0:T}) = p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)}$$

The parameterization for mean function were proposed by *Ho. et al.* [3] and can be computed as follows:

$$\boldsymbol{\mu_\theta(x_t, t) = \frac{1}{\sqrt{a_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\overline{\alpha}_t}}\epsilon_\theta(x_t, t)\right)}$$

The authors in *Ho. et al.* [3] suggested to use a fixed variance function as Σθ = βτ. The sample at time ***t-1*** can then be computed as follows:

$$\boldsymbol{x_{t-1} = \frac{1}{\sqrt{a_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\epsilon_\theta(x_t, t)\right) + \sigma_t z}$$

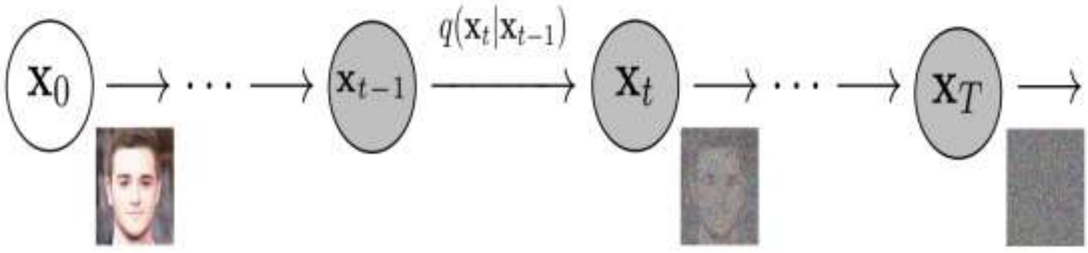


Fig. 4.5- Adding noise to the image [5]

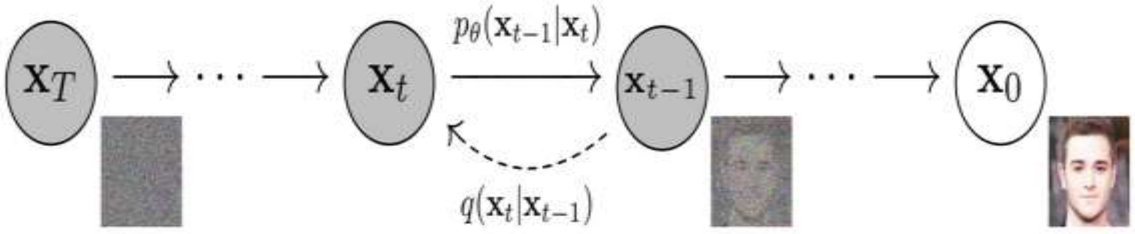

$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$
$\mathbf{x}_T$
$\mathbf{x}_t$
$\mathbf{x}_{t-1}$
$\mathbf{x}_0$

Fig. 4.6- Recreating image by denoising [6]

4.3. DIFFUSION MODELS

We consider the Gaussian diffusion models introduced by Sohl-Dickstein et al. (2015) and improved by Song & Ermon (2020b); Ho et al. (2020). Given a sample from the data distribution x0 ∼ q(x0), we produce a Markov chain of latent variables x1, ..., xT by progressively adding Gaussian noise to the sample:

$$q(x_t|x_{t-1}) \coloneqq \mathcal{N}(x_t; \sqrt{\alpha_t}x_{t-1}, (1-\alpha_t)\mathcal{I})$$

If the magnitude 1 − αt of the noise added at each step is small enough, the posterior q(xt−1|xt) is well approximated by a diagonal Gaussian. Furthermore, if the magnitude 1 − α1...αT of the total noise added throughout the chain is large enough, xT is well approximated by N (0, I). These properties suggest learning a model pθ(xt−1|xt) to approximate the true posterior:

$$p_\theta(x_{t-1}|x_t) := \mathcal{N}(\mu_\theta(x_t), \Sigma_\theta(x_t))$$

which can be used to produce samples x0 ∼ pθ(x0) by starting with Gaussian noise xT ∼ N (0, I) and gradually reducing the noise in a sequence of steps xT −1, xT −2, ..., x0. While there exists a tractable variational lower-bound on log pθ(x0), better results arise from optimizing a surrogate objective which re-weighs the terms in the VLB. To compute this surrogate objective, we generate samples xt ∼ q(xt|x0) by applying Gaussian noise to x0, then train a model θ to predict the added noise using a standard mean-squared error loss:

$$L_{\text{simple}} := E_{t\sim[1,T],x_0\sim q(x_0),\epsilon\sim\mathcal{N}(0,\mathbf{I})}[||\epsilon - \epsilon_\theta(x_t, t)||^2]$$

Ho (2020) show how to derive μθ(xt) from θ(xt, t), and fix Σθ to a constant. They also show the equivalence to previous denoising score-matching based models (Song & Ermon, 2020b; a), with the score function ∇xt log p(xt) ∝ θ(xt, t). In a follow-up work, Nichol & Dhariwal (2021) present a strategy for learning Σθ, which enables the model to produce high quality samples with fewer diffusion steps. We adopt this technique in training the models in this paper. Diffusion models have also been successfully applied to image super-resolution (Nichol & Dhariwal, 2021; Saharia et al., 2021b). Following the standard formulation of diffusion, high-resolution images y0 are progressively noised in a sequence of steps. However, pθ(yt−1|yt, x) additionally conditions on the downsampled input x, which is provided to the model by concatenating x (bicubic upsampled) in the channel dimension. Results from these models outperform prior methods on FID, IS, and in human comparison scores.

$$\hat{\mu}_\theta(x_t|y) = \mu_\theta(x_t|y) + s \cdot \Sigma_\theta(x_t|y)\nabla_{x_t} \log p_\phi(y|x_t)$$

4.4. UNet

UNet is a deep learning architecture for image segmentation tasks, which was proposed in the 2015 paper "U-Net: Convolutional Networks for Biomedical Image Segmentation" by Ronneberger et al. The name "UNet" comes from the U-shaped architecture of the network, which consists of a contracting path followed by an expanding path.

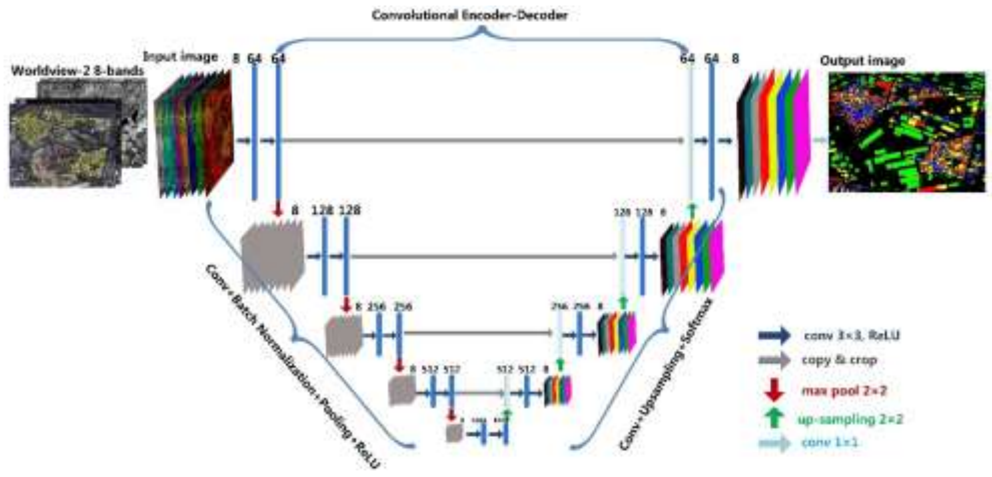


Fig. 4.7- UNet Architecture [23]

The UNet architecture is designed to segment images at the pixel level, which means that it assigns a label to each pixel in the image. This is useful for a wide range of applications, including medical imaging, where it is often necessary to identify and separate different types of tissue or organs in an image.

The UNet architecture consists of an encoder network and a decoder network. The encoder network is a series of convolutional and pooling layers that gradually reduce the spatial resolution of the input image, while increasing the number of feature maps. This "contracting path" is designed to capture the global context of the image and to learn low-level features.

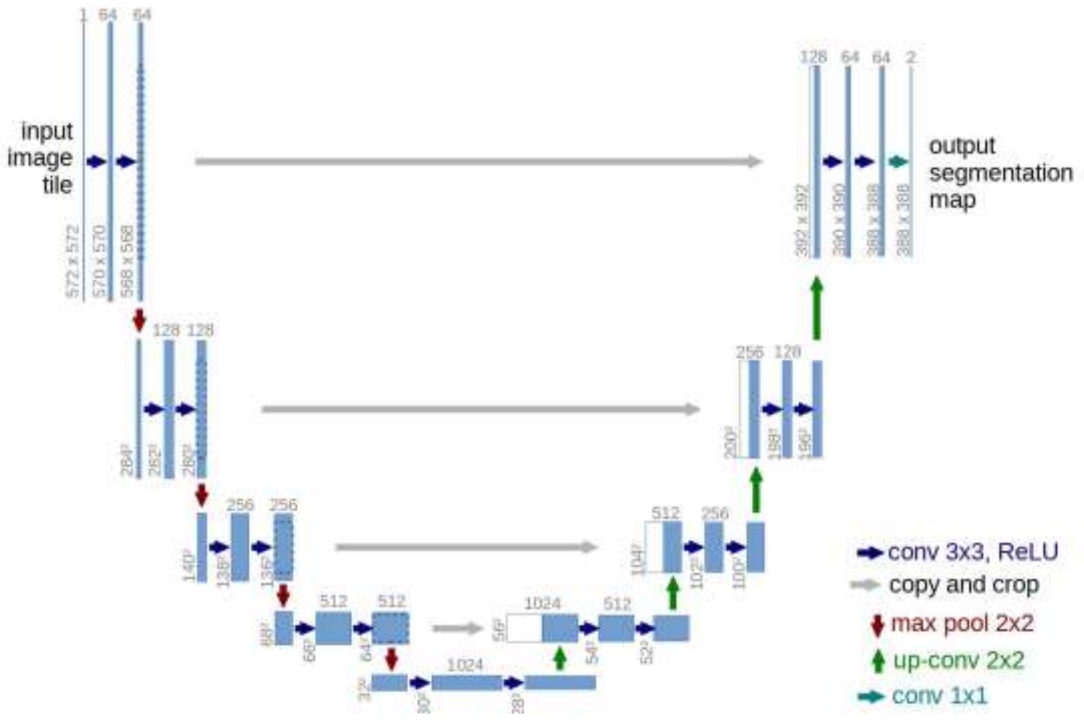


Fig. 4.8-UNet Working [23]

The decoder network is a series of upsampling and convolutional layers that gradually increase the spatial resolution of the input feature maps, while reducing the number of feature maps. This "expanding path" is designed to capture the local details of the image and to learn high-level features.

The key innovation of the UNet architecture is the use of skip connections between the encoder and decoder networks. These skip connections allow the decoder network to access high-resolution feature maps from the encoder network, which can help to preserve fine-grained details in the segmentation.

During training, the UNet architecture is trained using a pixel-wise cross-entropy loss function, which measures the difference between the predicted segmentation map and the ground truth segmentation map. The model is typically trained using stochastic gradient descent (SGD) with a small learning rate, and may also use data augmentation techniques such as random cropping, flipping, or rotation to improve generalization.

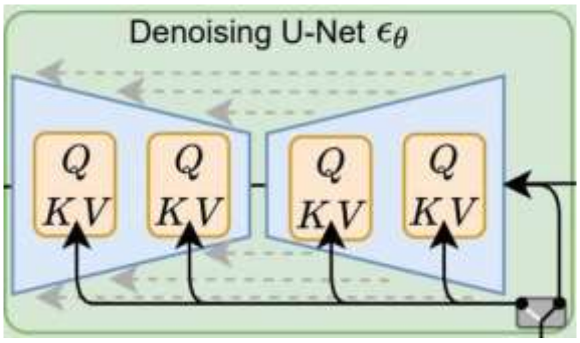


Fig. 4.9- UNet in Diffusion Model [24]

4.5. GRADIO

Gradio is a Python library that makes it easy to build and deploy machine learning models with user-friendly interfaces. With Gradio, you can quickly create web-based user interfaces for your machine learning models without needing to know much about web development.

Here is a step-by-step explanation of how to deploy a machine learning model using Gradio:

1. Install Gradio: The first step is to install Gradio. You can do this using pip, which is a package installer for Python. To install Gradio, open a terminal or command prompt and enter the command `pip install gradio`.

2. Define your machine learning model: The next step is to define your machine learning model. You can use any machine learning framework you like, such as TensorFlow, PyTorch, or scikit-learn. For example, if you are building an image classifier using TensorFlow, you might define your model using the Keras API.

3. Define your input and output interfaces: Once you have your machine learning model, you need to define the input and output interfaces for your Gradio app. Gradio provides a simple API for defining input and output interfaces. For example, you might define an input interface for uploading an image and an output interface for displaying the predicted class and confidence score.

5. Define your app: After you have defined your model and your input and output interfaces, you need to define your Gradio app. You can do this by creating an instance of the `gradio.Interface` class and passing in your input and output interfaces. You can also set various options for your app, such as the title, description, and theme.
6. Launch your app: Finally, you can launch your Gradio app by calling the `launch` method on your Gradio interface instance. This will start a local web server that serves your app, which you can access using a web browser. You can also deploy your app to a cloud server, such as Heroku or Google Cloud, to make it accessible over the internet.

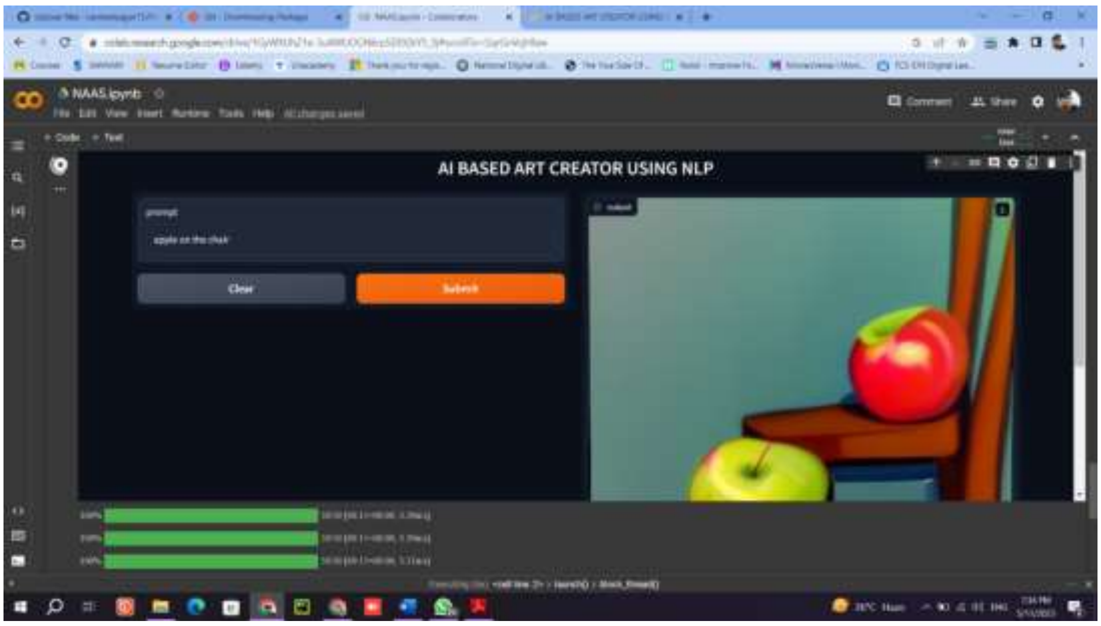


Fig. 4.10- Gradio

CHAPTER 5

SAMPLE OUTPUTS AND SCREENSHOTS

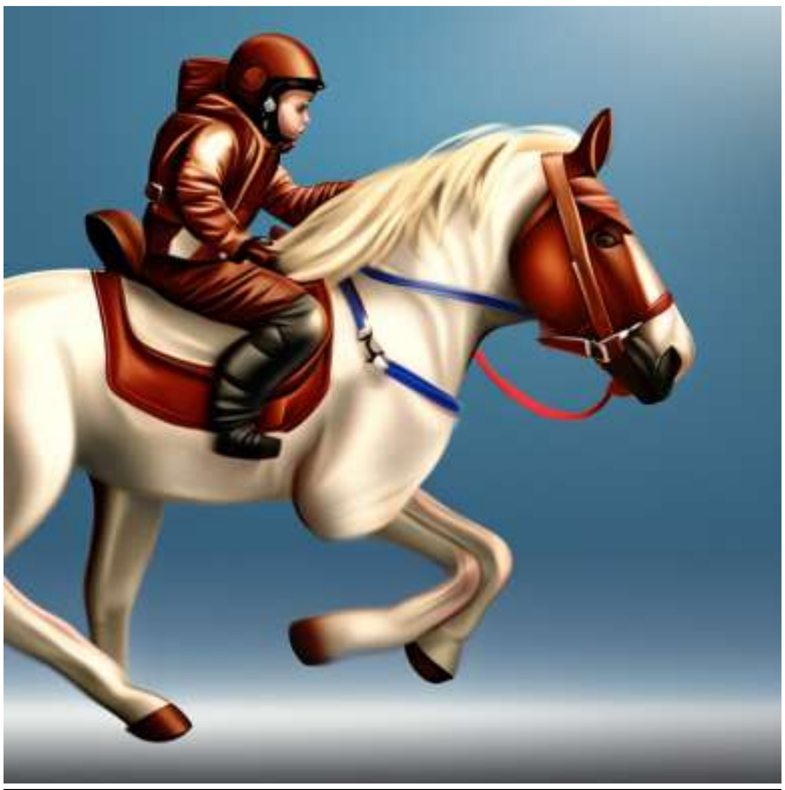


Fig 5.1- An astronaut riding horse in photorealistic view

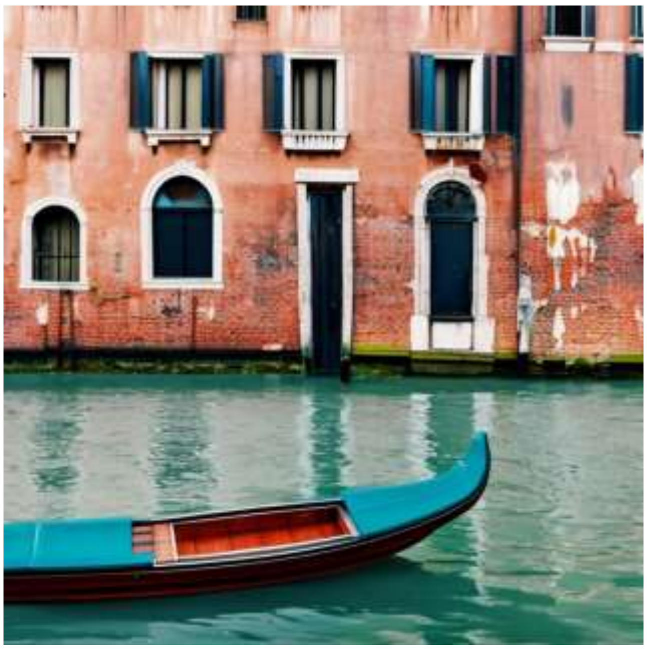


Fig 5.2- A boat in the canals of venice

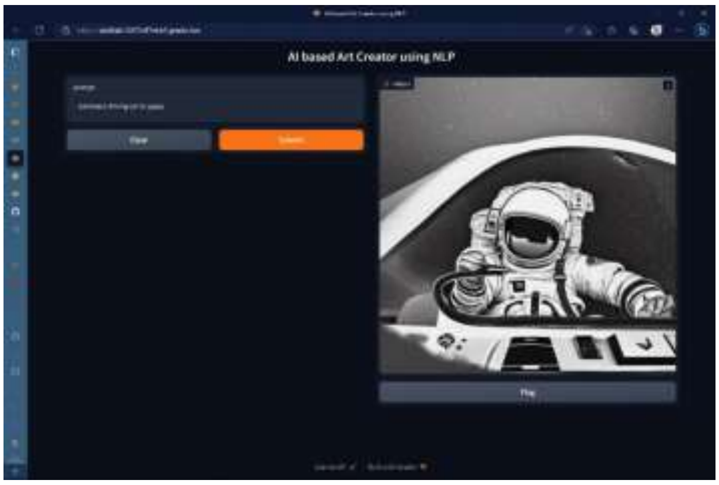


Fig 5.3- An astronaut driving a car

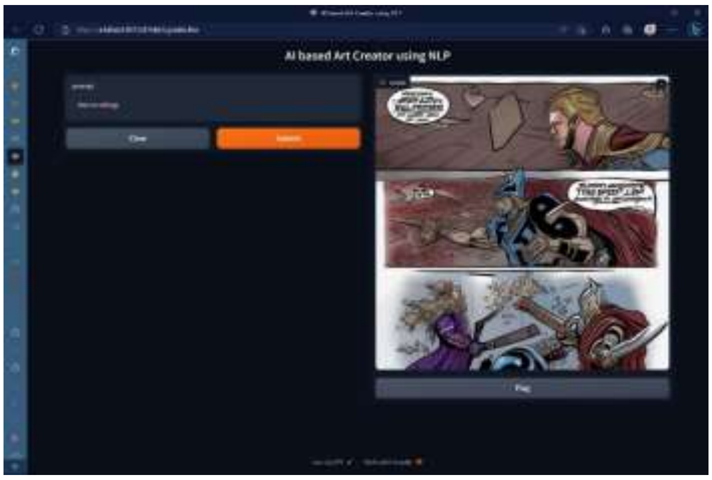


Fig 5.4- Thor vs Vikings

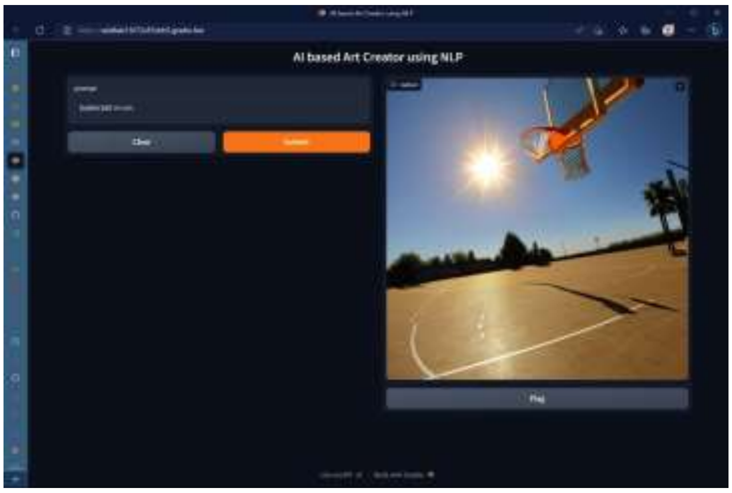


Fig 5.5- Basketball on Sun.

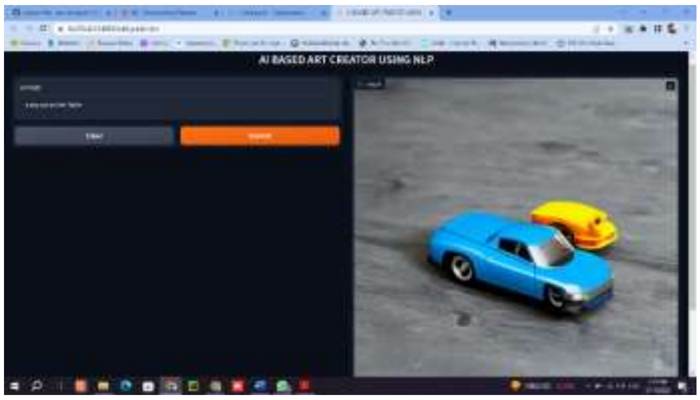


Fig 5.6- A Toy car on the Table.

CHAPTER 6

FUTURE WORK

- Our future aim will be to Generating more realistic images that will look as real-world existing image
- We would try to add a spelling correction option in the text field that will help in the assumption of the users' text prompt.
- We will try to develop a web or android application for the end users.
- Work on development of API for multi usage an interoperability on different devices

CHAPTER 7
REFERENCES

[1] Ramesh A., Pavlov M., Goh G., Gray S., Voss C., Radford A., Chen M., and Sutskever I.: Zero-Shot Text-to-Image Generation. In: Meila, M., and Zhang, T. (eds.) Proceedings of the 38th International Conference on Machine Learning. Proceedings of Machine Learning Research, pp. 8821–8831. PMLR (2021)

[2] Alec Radford, Jong Wook Kim, Chris Hallacy, AdityaRamesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry,Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. arXiv preprint arXiv:2103.00020, 2021.

[3] Prof. Sainath Patil,Akanksha Singh, Sonam Anekar, Ritika Shenoy : Text to Image using Deep Learning. ISSN: 2278-0181, published on International Journal of Engineering Research & Technology (IJERT)

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

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

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

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

[8] Nichol, Alex, et al. GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models. arXiv:2112.10741, arXiv, 8 Mar. 2022

[9] Ho, Jonathan, and Tim Salimans. Classifier-Free Diffusion Guidance. 2021. openreview.net

[10] Saharia, Chitwan, et al. Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding. arXiv:2205.11487, arXiv, 23 May 2022

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

[12] Ho, Jonathan, et al. Cascaded Diffusion Models for High Fidelity Image Generation. arXiv:2106.15282, arXiv, 17 Dec. 2021

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

[14] O'Connor, Ryan. Introduction to Diffusion Models for Machine Learning AssemblyAI Blog, 12 May 2022

[15] Rogge, Niels and Rasul, Kashif. The Annotated Diffusion Model . Hugging Face Blog, 7 June 2022

[16] Das, Ayan. “An Introduction to Diffusion Probabilistic Models.” Ayan Das, 4 Dec. 2021

[17] Song, Yang, and Stefano Ermon. Generative Modeling by Estimating Gradients of the Data Distribution. arXiv:1907.05600, arXiv, 10 Oct. 2020

[18] Song, Yang, et al. Score-Based Generative Modeling through Stochastic Differential Equations. arXiv:2011.13456, arXiv, 10 Feb. 2021

[19] Song, Yang. Generative Modeling by Estimating Gradients of the Data Distribution, 5 May 2021

[20] Luo, Calvin. Understanding Diffusion Models: A Unified Perspective. 25 Aug. 2022

[21] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, Dimitris N. Metaxas. StackGAN++: Realistic Image Synthesis with Stacked Generative Adversarial Networks. 10 dec. 2016.

[22] Ting Chen, Google Research, Brain Team, On the Importance of Noise Scheduling for Diffusion Model, 21 May. 2022.

[23] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation 18 May 2015.

[24] Website source: https://images.app.goo.gl/zp55sCcHxHaSbay66, 26 May 2023