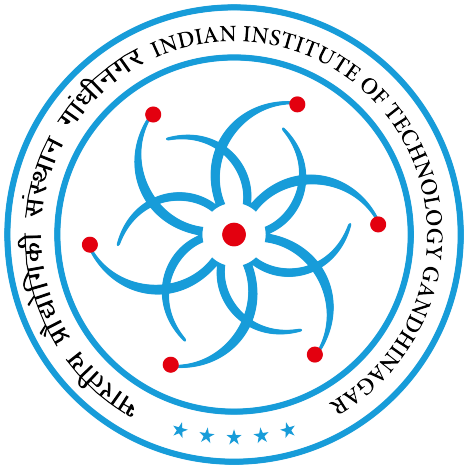
Indian Institute of Technology Gandhinagar
Computer Vision, Imaging, and Graphics (CVIG) Lab



Summer Research Internship Report
INDIAN INSTITUTE OF TECHNOLOGY GANDHINAGAR
Palaj, Gandhinagar - 382355

Generative Deep Learning for Computer Vision

Submitted by

Guntas Singh Saran
Junior Undergraduate, Computer Science and Engineering
Indian Institute of Technology Gandhinagar

Under the guidance of

Prof. Shanmuganathan Raman
Jibaben Patel Chair in Artificial Intelligence and Associate Professor
Electrical Engineering & Computer Science and Engineering
Indian Institute of Technology Gandhinagar

Contents

Chapter 1

Introduction

1.1 Background

The field of computer vision has experienced significant advancements over the past decade, largely due to the development and application of deep learning models. Among these, generative models have emerged as powerful tools capable of creating new data instances that resemble a given dataset. These models have wide-ranging applications, from image synthesis and inpainting to style transfer and data augmentation. Key types of generative models include Variational Autoencoders (VAEs) [1], Generative Adversarial Networks (GANs) [2], and Diffusion Probabilistic Models (DPMs) [3].

Variational Autoencoders (VAEs) introduced a probabilistic approach to generating data by learning latent representations. GANs, known for their two-network architecture comprising a generator and a discriminator, have been particularly successful in producing high-quality images. More recently, Diffusion Probabilistic Models have gained attention for their ability to generate detailed and diverse samples through iterative denoising processes.

1.2 Objective of the Research

The primary objective of this research is to explore and implement various generative models for computer vision tasks. Specifically, the research focuses on:

- Understanding and implementing Variational Autoencoders (VAEs), Vector-Quantized VAEs, Generative Adversarial Networks (GANs), and Diffusion Probabilistic Methods.
- Applying these models to tasks such as image inpainting, image synthesis, and editing.
- Investigating techniques like GAN inversion for image compression and editing.
- Exploring the potential of conditional generative models that leverage text and class-based conditioning.

Chapter 2

Literature Review

2.1 Variational Autoencoders (VAEs)

Variational Autoencoders (VAEs) are a type of generative model that combines principles from Bayesian inference and deep learning. They consist of an encoder, which maps input data to a latent space, and a decoder, which reconstructs the input from the latent representation. The key innovation of VAEs is the use of a probabilistic latent space, which allows for generating new data samples by sampling from this space.

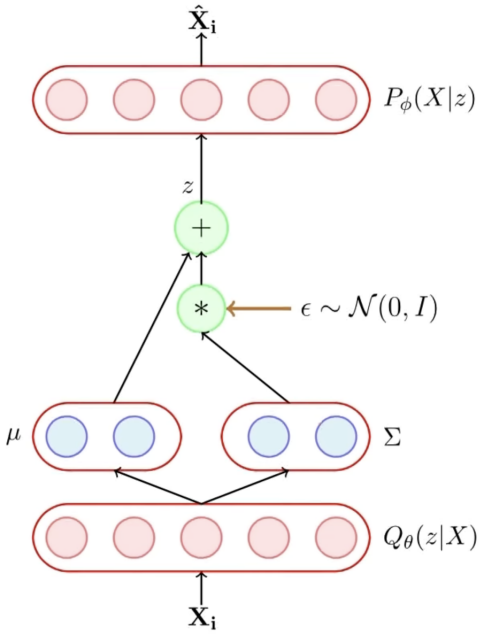


Figure 2.1: Graphical Model of a VAE

D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2022 [1] was the paper that introduced this idea of variational bayes.

2.2 Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) consist of two neural networks: a generator that creates data samples and a discriminator that distinguishes between real and generated samples. The two networks are trained simultaneously in a zero-sum game, where the generator aims

to produce realistic samples, and the discriminator strives to correctly identify them.

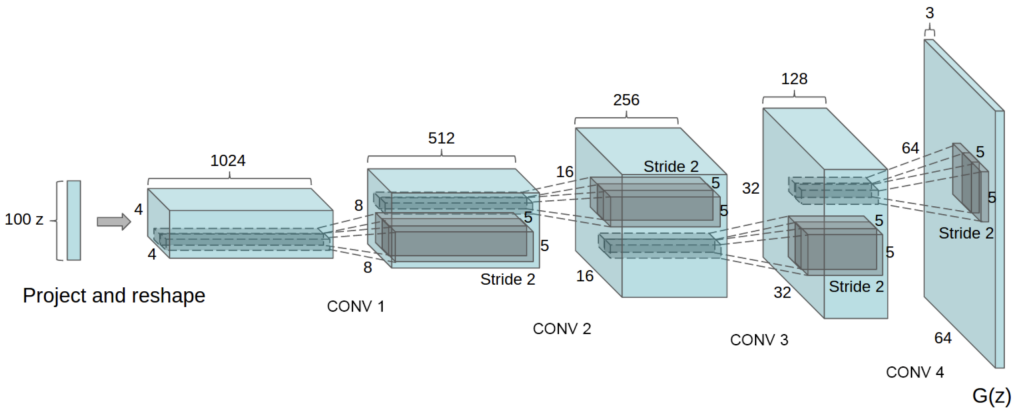


Figure 2.2: DCGAN Architecture. Source [4]

I. J. Goodfellow et al., "Generative adversarial networks," 2014. [2] This seminal paper introduces GANs, laying the foundation for numerous advancements in generative modeling. The authors demonstrate GANs' capability to generate high-quality images and discuss their potential applications.

A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2016. [4] This paper introduces DCGANs, which use convolutional layers to improve the quality and stability of GANs.

2.3 Diffusion Probabilistic Models

Diffusion Probabilistic Models (DPMs) are generative models that reverse a gradual noise-adding process to generate data. They are particularly effective in generating high-quality images with fine details.

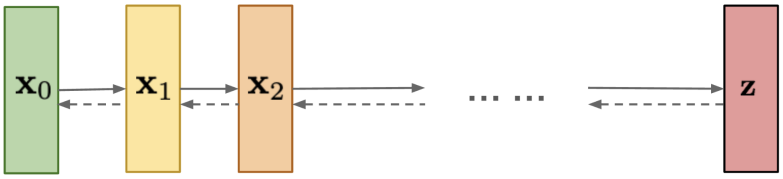


Figure 2.3: The Diffusion Process. Source [5]

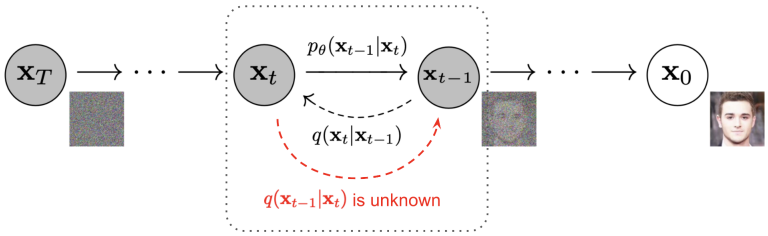


Figure 2.4: Forward and Reverse Distributions. Source [3]

J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," 2020. [3] This

paper introduces the concept of DPMs and demonstrates their effectiveness in generating high-fidelity images by progressively denoising a noisy signal.

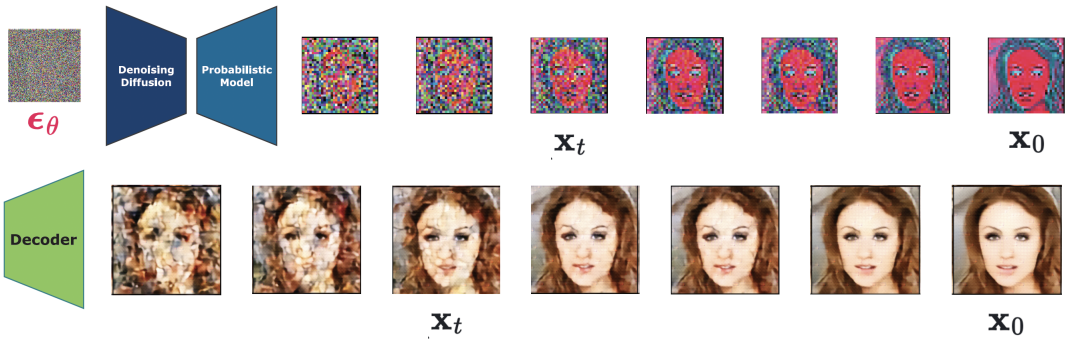


Figure 2.5: Reverse Sampling from LDMs.

R. Rombach et al., "High-resolution image synthesis with latent diffusion models," 2022. [6] This paper discusses the application of diffusion models in high-resolution image synthesis, highlighting their advantages over traditional generative models.

2.4 U-Net Architecture

The U-Net architecture, originally developed for biomedical image segmentation, has found applications in various generative modeling tasks, including image synthesis and inpainting.

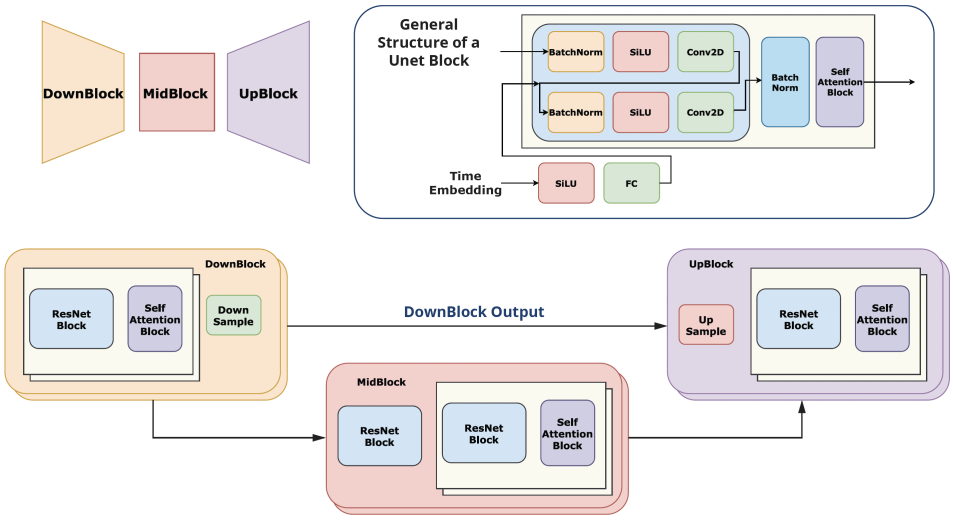


Figure 2.6: UNet Blocks used in DMs.

O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015. [7] This paper introduces the U-Net architecture, which uses a symmetric encoder-decoder structure to achieve high-quality image segmentation.

2.5 Perceptual Metrics for Image Quality

Perceptual metrics provide a way to evaluate the quality of generated images based on human visual perception. These metrics are essential for assessing the performance of generative models.

R. Zhang et al., "The unreasonable effectiveness of deep features as a perceptual metric,"

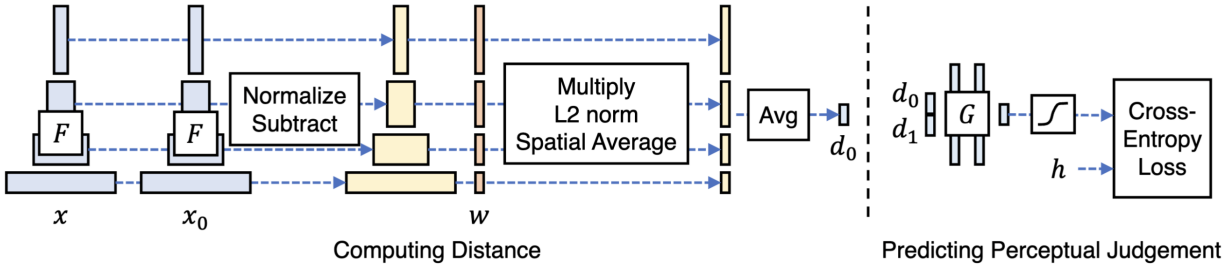


Figure 2.7: Idea of Feature Map comparison. Source [8]

Figure 2.8: Perceptual Features in Comparison. Source [8]

2018. [8] This paper proposes using deep learning features as perceptual metrics, demonstrating their effectiveness in evaluating image quality.

Chapter 3

Methodology & Results

3.1 Work environment

Linux system equipped with **NVIDIA RTX 4090** and **24GB RAM** was used for the training tasks. The entire project is hosted over at Github. We used two primary datasets: CelebAHQ-Mask, and MNIST. The datasets are described below with their key attributes.

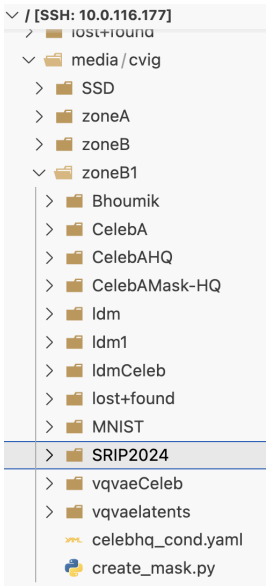


Figure 3.1: File System.

3.1.1 CelebAHQ-Mask

The CelebAHQ-Mask dataset is a high-resolution version of the CelebA dataset, which contains celebrity images with segmentation masks.

Attribute	Description
Number of Images	30,000
Image Resolution	256 x 256
Segmentation Masks	18 Attributes per image
Dataset Size	2.9 GB

Table 3.1: CelebAHQ-Mask Dataset Information

(a) The CelebAHQ Dataset

(b) Hair Segmentation Masks

Figure 3.2: Comparison of the CelebAHQ Dataset and Hair Segmentation Masks

3.1.2 MNIST

The MNIST dataset is a large database of handwritten digits that is commonly used for training various image processing systems.

Figure 3.3: MNIST Dataset

Attribute	Description
Number of Images	70,000
Image Resolution	28x28
Number of Classes	10 (digits 0-9)

Table 3.2: MNIST Dataset Information

3.2 Understanding the maths behind VAEs

We wish to achieve two goals:

1. Learning Abstraction $\rightarrow$ A hidden representation given the input $P(z|X)$ - this is achived by the Encoder $Q_\theta(z|X)$.
2. Generation $\rightarrow$ given some hidden representation using the Decoder $P_\phi(X|z)$.

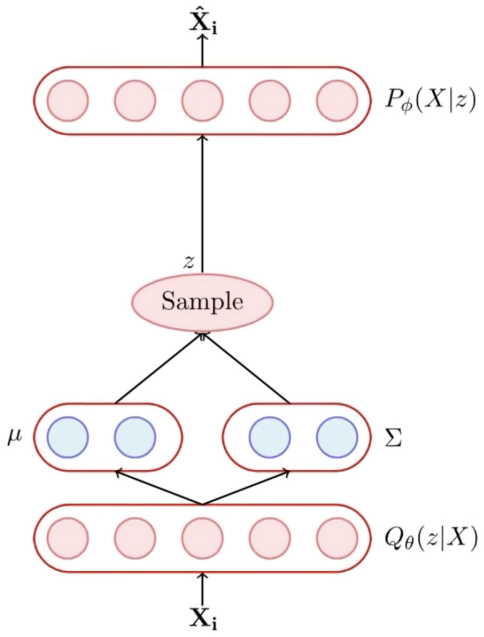


Figure 3.4: VAE Graphical Model

We assume the posterior distribution $P(z|X)$ as $Q_\theta(z|X)$. Further assume that $Q_\theta(z|X)$ is a Gaussian whose parameters are determined by our neural network $\rightarrow$ Encoder. And since we want this distribution to be close to $P(z|X)$, hence we wish to minimise their KL Divergence.

$$\min_\theta D_{KL}\left(Q_\theta(z|X) \parallel P(z|X)\right)$$

This leads to final objective as:

$$\boxed{\mathcal{L}(\theta,\phi) = \max_{\theta,\phi}\left\{\mathbb{E}_Q\left[\log(P_\phi(X|z))\right] - D_{KL}\left(Q_\theta(z|X) \parallel P(z)\right)\right\}}$$

To get the KL divergence, we make a forward pass through the Encoder to get $Q_\theta(z|X)$ and we know $P(z)$

$$Q_\theta(z|X) \sim \mathcal{N}(\boldsymbol{\mu_z}(X), \Sigma_z(X))$$
$$P(z) \sim \mathcal{N}(\mathbf{0}, I)$$

So, one half of the loss function is easily computable as:

$$D_{KL}\left(Q_\theta(z|X) \parallel P(z)\right) = \frac{1}{2}\left[\boldsymbol{\mu_z}^T(X)\boldsymbol{\mu_z}(X) + tr\left\{\Sigma_z(X)\right\} - k - \log|\Sigma_z(X)|\right]$$

Now the second part for the decoder, we'll approximate the expectation with a single z drawn sampled from $Q_\theta(z|X)$. Assume again that $P(X|z)$ as Gaussian with mean $\mu(z)$ and unit variance.

$$\boldsymbol{\mu} = f_\phi(z)$$

Hence the log-likelihood of the $P(X = x_i|z)$ can be written as:

$$\log(P(X = x_i|z)) = C - \frac{1}{2}||X_i - f_\phi(z)||^2$$

Hence the overall loss function is

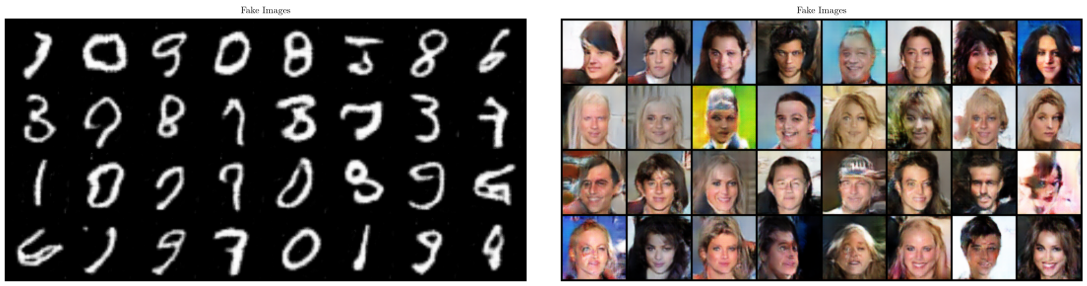
$$\boxed{\min_{\theta,\phi}\left\{||X_i - f_\phi(z)||^2 + \frac{1}{2}\left[\boldsymbol{\mu_z}^T(X)\boldsymbol{\mu_z}(X) + tr\left\{\Sigma_z(X)\right\} - k - \log|\Sigma_z(X)|\right]\right\}}$$

3.3 Implementing Generative Adversarial Nets using DCGAN

Generator : G_ϕ

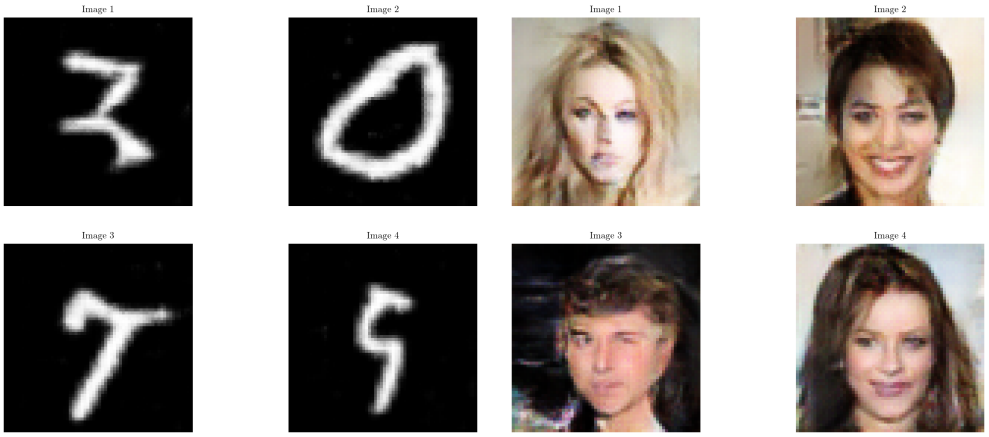
Discriminator : D_θ

$$\min_\phi \max_\theta V(G, D) = \mathop{\mathbb{E}}_{\mathbf{x}\sim p_{\text{data}}}[\log(D_\theta(\mathbf{x}))] + \mathop{\mathbb{E}}_{\mathbf{z}\sim p_z(\mathbf{z})}[1 - \log(D_\theta(G_\phi(\mathbf{z})))]$$



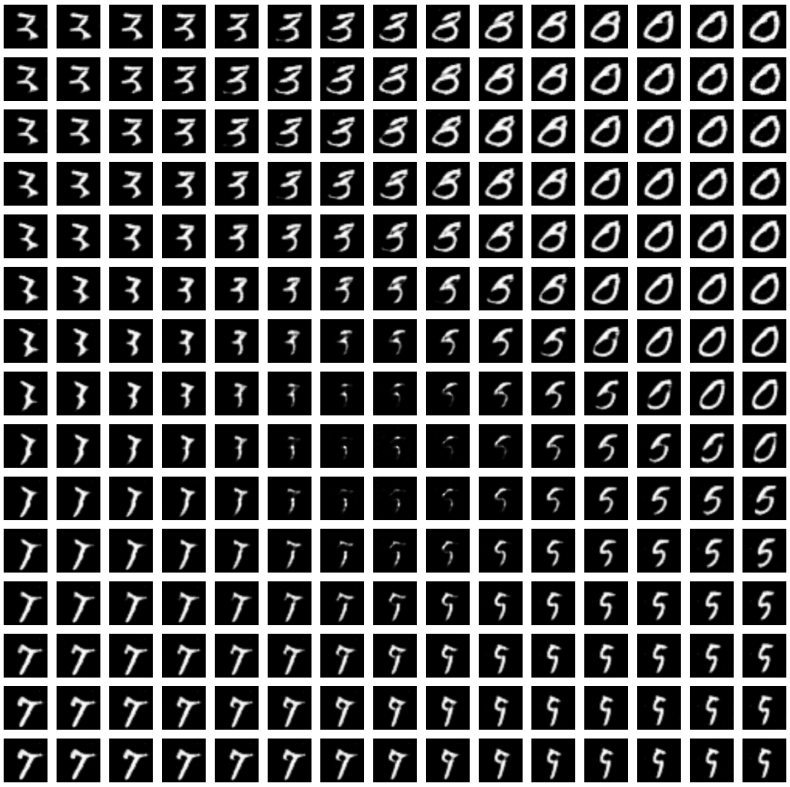
(a) Fake MNIST (b) Fake CelebA

Figure 3.5: Adversarial Images generated for MNIST and Celeb Datasets



(a) Fake MNIST (b) Fake CelebA

Figure 3.6: Four samples generated from random noise for interpolation



(a) MNIST Space

(b) CelebA Space

Figure 3.7: All points to verify that GAN has mapped to entire space

3.4 Exploring Image Editing using StyleGAN

Following the idea proposed in the paper InDomain GAN Inversion for Real-Image Editing [9], we leveraged the notebook for getting the following results.

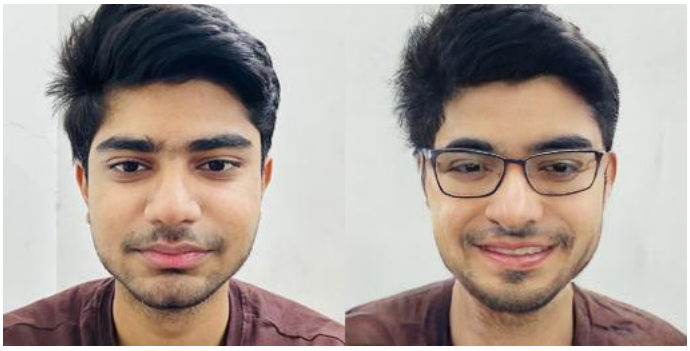


Figure 3.8: Editing Face Features for Real Images

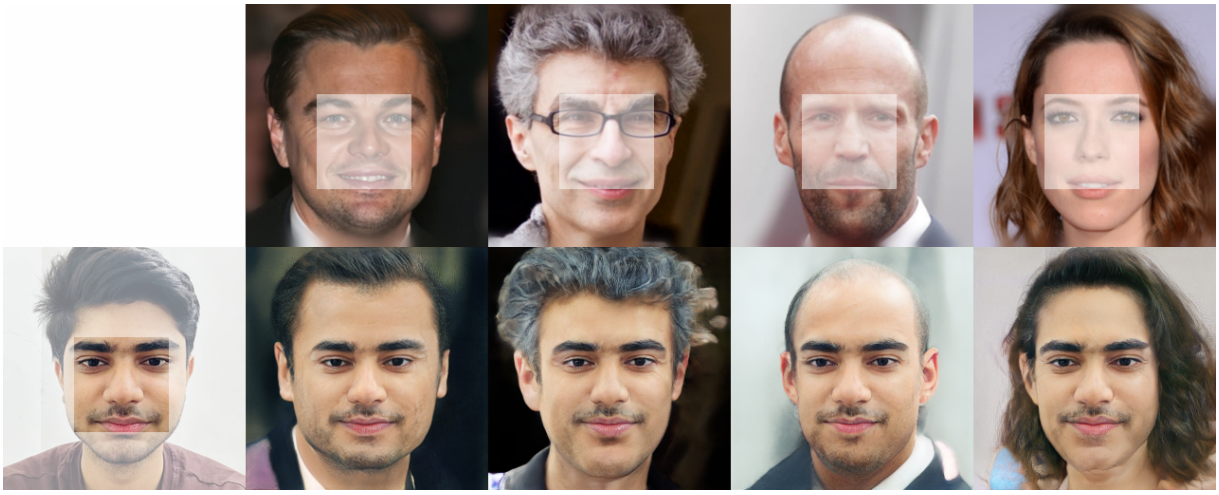


Figure 3.9: Semantic Diffusion for image editing

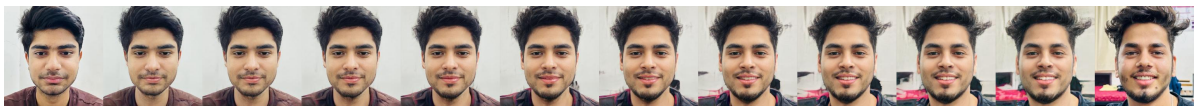


Figure 3.10: Linear Interpolation of real images

3.5 Understanding Diffusion Models

As seen in the case of VAEs, it all boils down to learning the probability distributions - $p(\mathbf{z}|\mathbf{x})$ the posterior abstraction of obtaining a hidden representation $\mathbf{z}$ given some input image $\mathbf{x}$ and the likelihood $p(\mathbf{x}|\mathbf{z})$ of generating the image samples given some hidden representation $\mathbf{z}$.

Now the most crucial task in all these generative models is trying to understand to relate the objective that we are trying to acheive and what the model actually learns. We'll see the same confusing conclusion being established by the end of this blog and then we'll realise how

beautifully all the mathematics and the tasks laid out make sense.

Like in the case of VAEs, we started off by approximating the actual $P(\mathbf{z}|\mathbf{x})$ through our probabilistic Encoder $Q_\phi(\mathbf{z}|\mathbf{x})$ and minimising the KL divergence between these two. But in order to establish the knowledge of the actual $P(\mathbf{z}|\mathbf{x})$, we went into maximising the log-likelihood of data samples $\mathbf{x}$ and eventually made the encoder learn this distribution $Q_\phi(\mathbf{z}|\mathbf{x})$ to be as close to the standard normal $\mathcal{N}(\mathbf{0}, \mathbb{I})$ as possible. Hence now drawing any $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbb{I})$ we are sure of it being close to the $\mathbf{z}$'s seen during training, allowing us to discard off the encoder entirely at inference. The way we setup the objective of making the actual and approximated distribution close to each other will stay same for Diffusion Models too and this would allow us to uncover more truth about the actual distribution itself.

3.5.1 What are Diffusion Models?

For Diffusion Models instead of one latent variable $\mathbf{z}$, we have T latent variables of the form $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T$ of same dimension as the input image $\mathbf{x}_0$, and the most interesting point is that the forward noising process is a deterministic Markov Chain, wherein Gaussian noise is added in gradual T steps, defined as:

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

Here the variances are controlled by a scheduler $\{\beta_t \in (0,1)\}_{t=1}^T$, which means for each noisy sample $\mathbf{x}_t$ is sampled from a Gaussian with $\boldsymbol{\mu}_q = \sqrt{1-\beta_t}\mathbf{x}_{t-1}$ and covariance matrix $\boldsymbol{\Sigma}_q = \beta_t\mathbb{I}$. The idea is then to learn the reverse denoising diffusion distribution $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$, which is also a Markov Chain with learned Gaussian transitions starting at $p(\mathbf{x}_T) \sim \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbb{I})$. Therefore it then becomes really important to understand the entire joint distribution $p(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T)$ denoted in shorthand as $p(\mathbf{x}_{0:T})$

3.5.2 Markov-Diffusion Process and Reparametrization

The special property of a Markov Process is that the future state is dependent only on previous state, which means

$$P(\mathbf{x}_t|\mathbf{x}_{t-1}, \cdots, \mathbf{x}_0) = P(\mathbf{x}_t|\mathbf{x}_{t-1})$$

the utility of this is that using the chain rule of joint probability, we may simply write for all our diffusion process forward steps $q(\mathbf{x}_{1:T}|\mathbf{x}_0)$ as

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \quad q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbb{I})$$

Diffusion is the process of converting samples from a complex distribution (the data here) $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ to samples of a simple distribution (isotropic Gaussian noise) $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbb{I})$. One can also observe that there is a deterministic and a stochastic component even in our case. Since any RV can be reparametrized as $Z = \sigma X + \mu$, hence we denote the $\mathbf{x}_t$ being drawn from $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ as

$$\boxed{\mathbf{x}_t = \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\boldsymbol{\epsilon}_{t-1}}$$

where $\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\epsilon}_{t-2}, \cdots \sim \mathcal{N}(\mathbf{0}, \mathbb{I})$

3.5.3 Understanding the Forward Markov Process

Do we traverse for all T steps? Certainly Not! Here's how the Markov Process allows us to reach any $\mathbf{x}_t$ from the image $\mathbf{x}_0$. Let $\alpha_t = 1 - \beta_t$

$$\begin{aligned}\mathbf{x}_t &= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\mathcal{N}(\mathbf{0}, \mathbb{I}) \\ &= \sqrt{\alpha_t}(\sqrt{\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\mathcal{N}(\mathbf{0}, \mathbb{I})) + \sqrt{1-\alpha_t}\mathcal{N}(\mathbf{0}, \mathbb{I}) \\ &= (\sqrt{\alpha_t\alpha_{t-1}})\mathbf{x}_{t-2} + (\sqrt{\alpha_t}\sqrt{1-\alpha_{t-1}})\mathcal{N}(\mathbf{0}, \mathbb{I}) + (\sqrt{1-\alpha_t})\mathcal{N}(\mathbf{0}, \mathbb{I})\end{aligned}$$

Combine the independent Gaussians with $\sigma^2 = \alpha_t(1-\alpha_{t-1}) + (1-\alpha_t) = 1 - \alpha_t\alpha_{t-1}$, hence

$$\begin{aligned}\mathbf{x}_t &= (\sqrt{\alpha_t\alpha_{t-1}})\mathbf{x}_{t-2} + (\sqrt{1-\alpha_t\alpha_{t-1}})\mathcal{N}(\mathbf{0}, \mathbb{I}) \\ &\cdots \\ &= (\sqrt{\alpha_t\alpha_{t-1}\cdots\alpha_2\alpha_1})\mathbf{x}_0 + (\sqrt{1-\alpha_t\alpha_{t-1}\cdots\alpha_2\alpha_1})\mathcal{N}(\mathbf{0}, \mathbb{I})\end{aligned}$$

With $\bar{\alpha_t} = \prod_{i=1}^{t}\alpha_i$, we finally get

$$\boxed{\mathbf{x}_t = (\sqrt{\bar{\alpha_t}})\mathbf{x}_0 + (\sqrt{1-\bar{\alpha_t}})\mathcal{N}(\mathbf{0}, \mathbb{I})}$$

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

3.5.4 The Crucial Reverse Diffusion Process

If we can reverse the above process and sample from $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$, we will be able to recreate the true sample from a Gaussian noise input, $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbb{I})$. Note that if β_t is small enough, $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ will also be Gaussian. Unfortunately, we cannot easily estimate $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ because it needs to use the entire dataset and therefore we need to learn a model p_θ to approximate these conditional probabilities in order to run the reverse diffusion process. The actual reverse distribution

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q)$$

And the approximated distribution can be represented as

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \quad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

Before moving onto defining the objective to find the approximate $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, its noteworthy to understand the actual reverse process distribution $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$. The reverse conditional distribution is tractable when condition on $\mathbf{x}_0$ and since this a Markov Process, we can safely introduce this $\mathbf{x}_0$ in the joint conditional part and then we may expand it by Bayes' Rule

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0) \cdot q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)}$$

Notice that all of these are forward processes

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0) = q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_{t-1}, (1-\alpha_t)\mathbb{I}) \propto \exp\left(-\frac{1}{2}\frac{(\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_{t-1})^2}{(1-\alpha_t)}\right)$$

$$q(\mathbf{x}_{t-1}|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0, (1-\bar{\alpha}_{t-1})\mathbb{I}) \propto \exp\left(-\frac{1}{2}\frac{(\mathbf{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0)^2}{(1-\bar{\alpha}_{t-1})}\right)$$

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbb{I}) \propto \exp\left(-\frac{1}{2}\frac{(\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0)^2}{(1-\bar{\alpha}_t)}\right)$$

Hence, we may combine all these to get a single Gaussian for $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ as

$$\begin{aligned}
q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) &\propto \mathbf{exp}\left(-\frac{1}{2}\left(\frac{(\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_{t-1})^2}{(1-\alpha_t)} + \frac{(\mathbf{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0)^2}{(1-\bar{\alpha}_{t-1})} + \frac{(\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0)^2}{(1-\bar{\alpha}_t)}\right)\right) \\
&= \mathbf{exp}\left(-\frac{1}{2}\left(\frac{\mathbf{x}_t^2 - 2\sqrt{\alpha_t}\mathbf{x}_t\mathbf{x}_{t-1} + \alpha_t\mathbf{x}_{t-1}^2}{(1-\alpha_t)} + \frac{\bar{\alpha}_{t-1}\mathbf{x}_0^2 - \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0\mathbf{x}_{t-1} + \mathbf{x}_{t-1}^2}{(1-\bar{\alpha}_{t-1})} + \frac{(\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0)^2}{(1-\bar{\alpha}_t)}\right)\right) \\
&= \mathbf{exp}\left(-\frac{1}{2}\left(\left(\frac{\alpha_t}{1-\alpha_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right)\mathbf{x}_{t-1}^2 - 2\left(\frac{\sqrt{\alpha_t}\mathbf{x}_t}{1-\alpha_t} + \frac{\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0}{1-\bar{\alpha}_{t-1}}\right)\mathbf{x}_{t-1} + F(\mathbf{x}_t, \mathbf{x}_0)\right)\right) \\
&\propto \mathbf{exp}\left(-\frac{1}{2\boldsymbol{\sigma}^2}\left(\mathbf{x}_{t-1}^2 - 2\boldsymbol{\mu}\mathbf{x}_{t-1} + \boldsymbol{\mu}^2\right)\right) \\
&\propto \mathbf{exp}\left(-\frac{1}{2\boldsymbol{\sigma}^2}\left(\mathbf{x}_{t-1} - \boldsymbol{\mu}\right)^2\right)
\end{aligned}$$

where the variance can be written as

$$\begin{aligned}
\boldsymbol{\sigma}^2 &= 1/\left(\frac{\alpha_t}{1-\alpha_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right) \\
&= \frac{(1-\alpha_t)\cdot(1-\bar{\alpha}_{t-1})}{(1-\bar{\alpha}_t)}
\end{aligned}$$

and the mean as the weighted mean of $\mathbf{x}_t$ and $\mathbf{x}_0$ be written as

$$\begin{aligned}
\boldsymbol{\mu} &= \frac{(1-\alpha_t)\cdot(1-\bar{\alpha}_{t-1})}{(1-\bar{\alpha}_t)}\left(\frac{\sqrt{\alpha_t}}{1-\alpha_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\mathbf{x}_0\right) \\
&= \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{(1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_t}\mathbf{x}_0 \\
&= \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{(1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_t}\left(\frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t)\right) \\
&(\because \mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + (\sqrt{1-\bar{\alpha}_t})\boldsymbol{\epsilon}_t) \\
&= \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_t\right)
\end{aligned}$$

Finally we have the original reverse distribution $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ as

$$\boxed{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = q(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_q(\mathbf{x}_0, \mathbf{x}_t), \boldsymbol{\Sigma}_q(t))}$$

$$\boxed{\boldsymbol{\Sigma}_q(t) = \frac{(1-\alpha_t)\cdot(1-\bar{\alpha}_{t-1})}{(1-\bar{\alpha}_t)}\mathbb{I}}$$

$$\boxed{\boldsymbol{\mu}_q(\mathbf{x}_t, \mathbf{x}_0) = \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{(1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_t}\mathbf{x}_0}$$

$$\boxed{\boldsymbol{\mu}_q(t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_t\right)}$$

3.5.5 The Loss Function

As discussed before we will follow the same methodology as done in VAEs of learning the approximate reverse distribution $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ by maximizing the expected log-likelihood of the observed data $p_\theta(\mathbf{x}_0)$ for $\mathbf{x}_0 \sim q(\mathbf{x}_0)$

Using Jensen's Inequality over the $\log$ function (convex function), hence the expectation of $\log$ is lesser than equal to the $\log$ of expectation.

$$\begin{aligned}
L &= \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0)} \left[\log p_\theta(\mathbf{x}_0)\right] \\
&= \mathbb{E}_{q(\mathbf{x}_0)} \left[\log \int p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T}\right] \\
&= \mathbb{E}_{q(\mathbf{x}_0)} \left[\log \int q(\mathbf{x}_{1:T}|\mathbf{x}_0) \times \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} d\mathbf{x}_{1:T}\right] \\
&= \mathbb{E}_{q(\mathbf{x}_0)} \left[\log \left(\mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right]\right)\right] \\
&\geq \mathbb{E}_{q(\mathbf{x}_{0:T})} \left[\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right]
\end{aligned}$$

Notice that both these terms are joint probability distributions with $q(\mathbf{x}_{1:T}|\mathbf{x}_0)$ being the actual forward process and $p_\theta(\mathbf{x}_{0:T})$ the approximate reverse process. Expanding these terms out

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

Further we'll condition the forward process on $\mathbf{x}_0$ as it would later allow us to expand terms using Bayes' Rule

$$\begin{aligned}
q(\mathbf{x}_{1:T}|\mathbf{x}_0) &= \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \\
&= q(\mathbf{x}_1|\mathbf{x}_0) \cdot \prod_{t=2}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0) \\
&= q(\mathbf{x}_1|\mathbf{x}_0) \cdot \prod_{t=2}^{T} \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \cdot q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)} \\
&= q(\mathbf{x}_T|\mathbf{x}_0) \cdot \prod_{t=2}^{T} q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)
\end{aligned}$$

Substituting it all for the new lower bound loss

$$
\begin{aligned}
L_N &= \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] \\
&= \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log \frac{p_\theta(\mathbf{x}_T)\cdot\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_1|\mathbf{x}_0)\cdot\prod_{t=2}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{x}_0)}\right] \\
&= \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log \frac{p_\theta(\mathbf{x}_T)\cdot\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_1|\mathbf{x}_0)\cdot\prod_{t=2}^{T} \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\cdot q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}}\right] \\
&= \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log \frac{p_\theta(\mathbf{x}_T)\cdot\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_T|\mathbf{x}_0)\cdot\prod_{t=2}^{T} q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}\right] \\
&= \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log \frac{p_\theta(\mathbf{x}_T)}{q(\mathbf{x}_T|\mathbf{x}_0)} + \log p_\theta(\mathbf{x}_0|\mathbf{x}_1) + \sum_{t=2}^{T}\log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}\right] \\
&= \log p_\theta(\mathbf{x}_0|\mathbf{x}_1) + \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\sum_{t=2}^{T}\log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}\right] \\
L_N &= \sum_{t=2}^{T} D_{KL}\left(p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \parallel q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\right) + \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)
\end{aligned}
$$

3.5.6 Reparametrization of the Loss Function

The above loss function aims to bring the actual reverse $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ and approximated reverse distributions $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ as close as possible by the means of the $T-1$ KL Divergence terms. Since we approximate the reverse process using a neural network, the Divergence terms would imply that we want their means to be as close as possible

$$
q(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \left(\frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{(1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_t}\mathbf{x}_0\right), \frac{(1-\alpha_t)\cdot(1-\bar{\alpha}_{t-1})}{(1-\bar{\alpha}_t)}\mathbb{I})
$$

As stated in the paper, take both their covariances to be same, hence each term of the loss function L_t can be written as

$$
\boxed{L_t = \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{1}{2\|\boldsymbol{\Sigma}_\theta(\mathbf{x}_t,t)\|_2^2}\|\boldsymbol{\mu}_\theta(\mathbf{x}_t,t) - \boldsymbol{\mu}_t(\mathbf{x}_t,\mathbf{x}_0)\|^2\right]}
$$

The authors however define this in terms of the noise prediction. Since $\boldsymbol{\mu}_q(\mathbf{x}_t,t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_t\right)$, hence for the approximate reverse process distribution, we may write

$$
\boldsymbol{\mu}_\theta(\mathbf{x}_t,t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t)\right)
$$

Hence the objective can be re-written as

$$
\begin{aligned}
L_t &= \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{1}{2\|\boldsymbol{\Sigma}_\theta\|_2^2}\|\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t)\right) - \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_t\right)\|^2\right] \\
&= \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t)\|\boldsymbol{\Sigma}_\theta\|_2^2}\|\boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t) - \boldsymbol{\epsilon}_t\|^2\right]
\end{aligned}
$$

$$\boxed{L_t = \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t)\|\boldsymbol{\Sigma}_\theta\|_2^2}\|\boldsymbol{\epsilon}_\theta((\sqrt{\bar{\alpha}_t})\mathbf{x}_0 + (\sqrt{1-\bar{\alpha}_t})\boldsymbol{\epsilon}_t, t) - \boldsymbol{\epsilon}_t\|^2\right]}$$

Notice how beautifully it wraps down to just making the model to learn to approximate the noising $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ process over any $\mathbf{x}_t$ to the actual noise $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbb{I})$. Hence this quite so weird learning process lets us learn the denoising reverse distribution.

3.5.7 Modified Objective

The authors further found that training works better by dropping off the constant term entirely, so the final objective is

$$\boxed{L_t^{\text{Simple}} = \mathbb{E}_{t\sim[1,T],\mathbf{x}_0,\boldsymbol{\epsilon}_t}\left[\|\boldsymbol{\epsilon}_\theta((\sqrt{\bar{\alpha}_t})\mathbf{x}_0 + (\sqrt{1-\bar{\alpha}_t})\boldsymbol{\epsilon}_t, t) - \boldsymbol{\epsilon}_t\|^2\right]}$$

Algorithm 1 Training

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

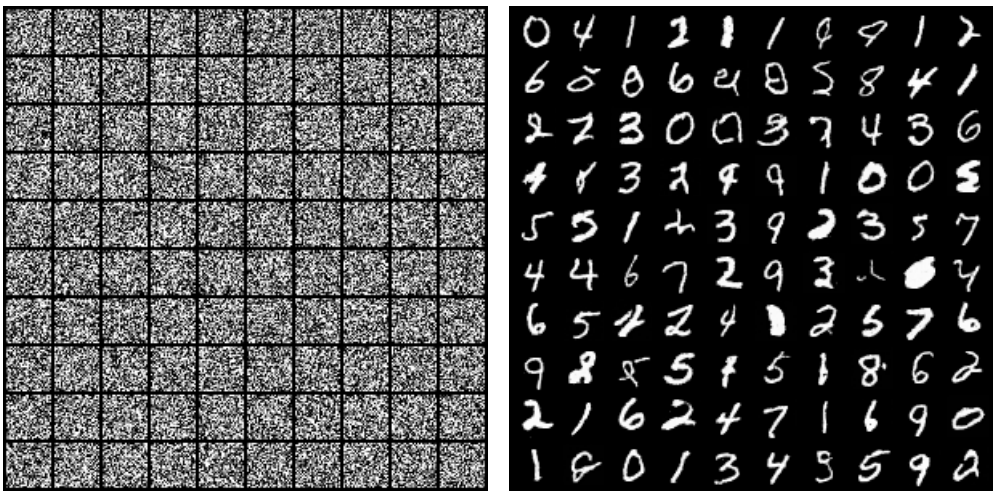
Algorithm 2 Sampling

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

Figure 3.11: Training and Sampling Algorithms for DDPMs. Source [3]

3.6 Implementing DPMs

Following the UNet architecture using ResNet [10] and Self-Attention Blocks, we obtained the following denoising results:



(a) Random Noise (b) Denoised Image

Figure 3.12: Reverse Denoising Process over MNIST Digits

3.7 Implementing Latent Diffusion Models

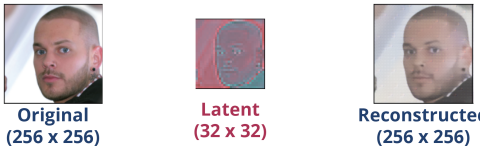


Figure 3.13: Idea of Latent Representation of the Image

3.7.1 Training AutoEncoder

Using the Perceptual Loss [8], Adversarial Loss, L2 Loss, we trained an AutoEncoder (VQVAE here [11]) to get a good quality reconstruction from the smaller latent space.

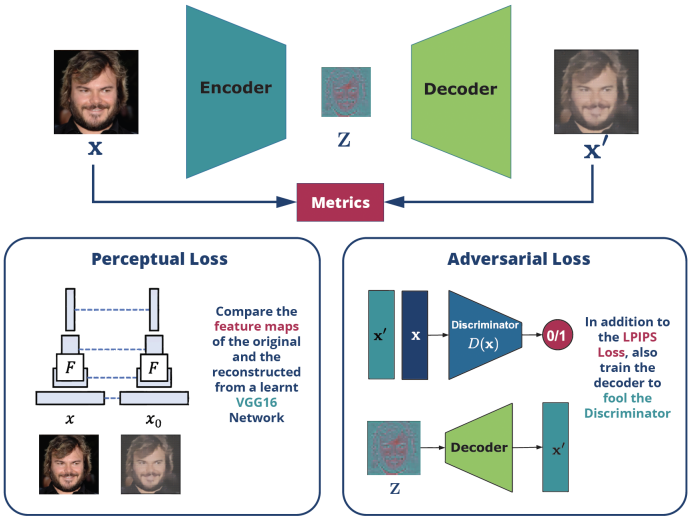


Figure 3.14: Training the AutoEncoder (VQVAE)

3.7.2 Results from LDM reverse sampling

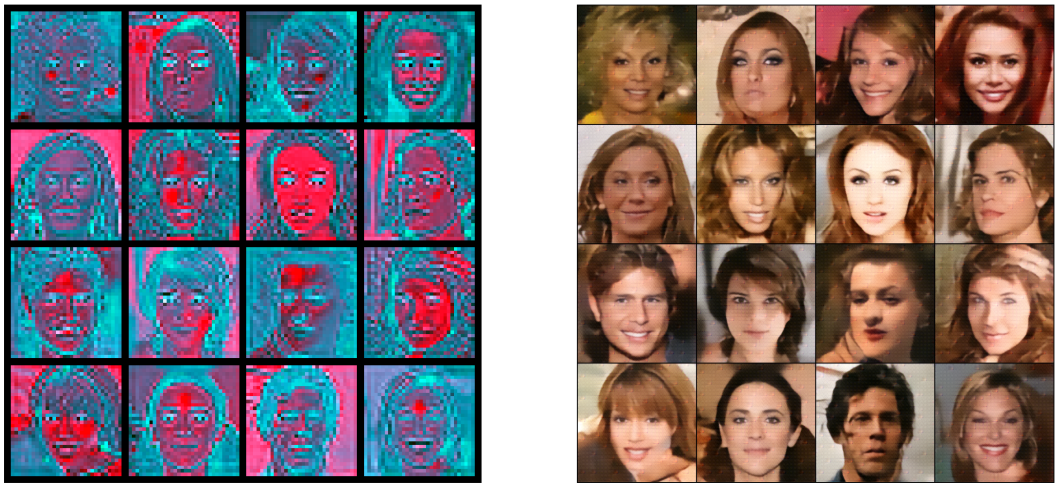


Figure 3.15: Final Denoised Latent Space Images decoded to Pixel Space using trained Decoder

3.8 Image InPainting using Latent Diffusion Models

Figure 3.16: Image InPainting Task

3.8.1 The Implementation

Without specifically training the LDM for this task we can still achieve the inpainting [6] of the masked region by feeding in the information of the unmasked regions through the forward process sampling and combining it with the masked regions of the reverse process denoised image.

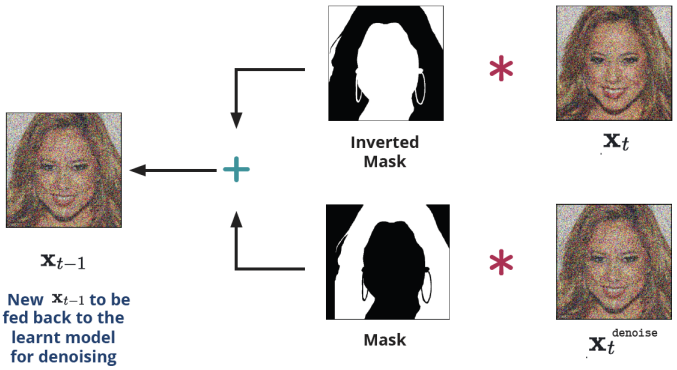


Figure 3.17: Changes in the reverse process

The only catch we do all this over the latent space of smaller resolution rather than the actual image space.

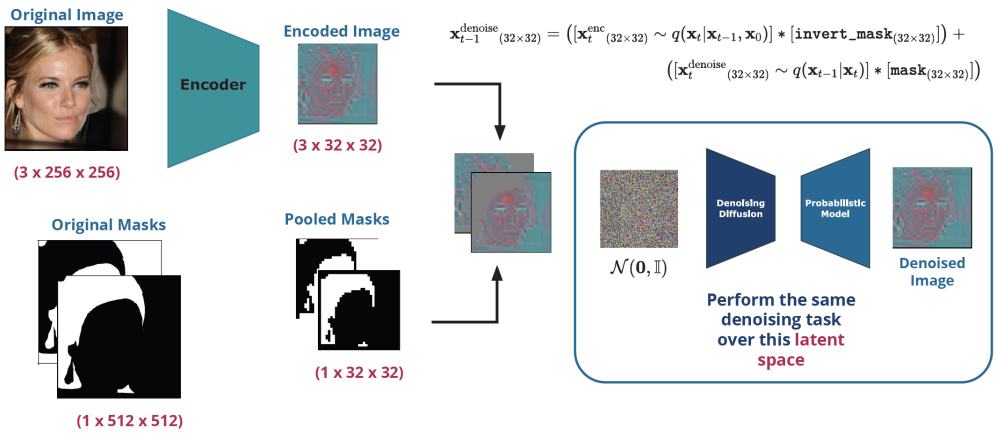


Figure 3.18: The whole inpainting process over the latent space

3.8.2 Results

(a) Original Images (b) Denoised Images (c) Final Mask Manipulation

Figure 3.19: Final InPainting Results

Chapter 4

Blogs and Presentation

4.1 Blog Page

All the progress throughout the summer was kept engaging by writing personal blogs and implementing the model architectures using rendered Jupyter Notebooks.

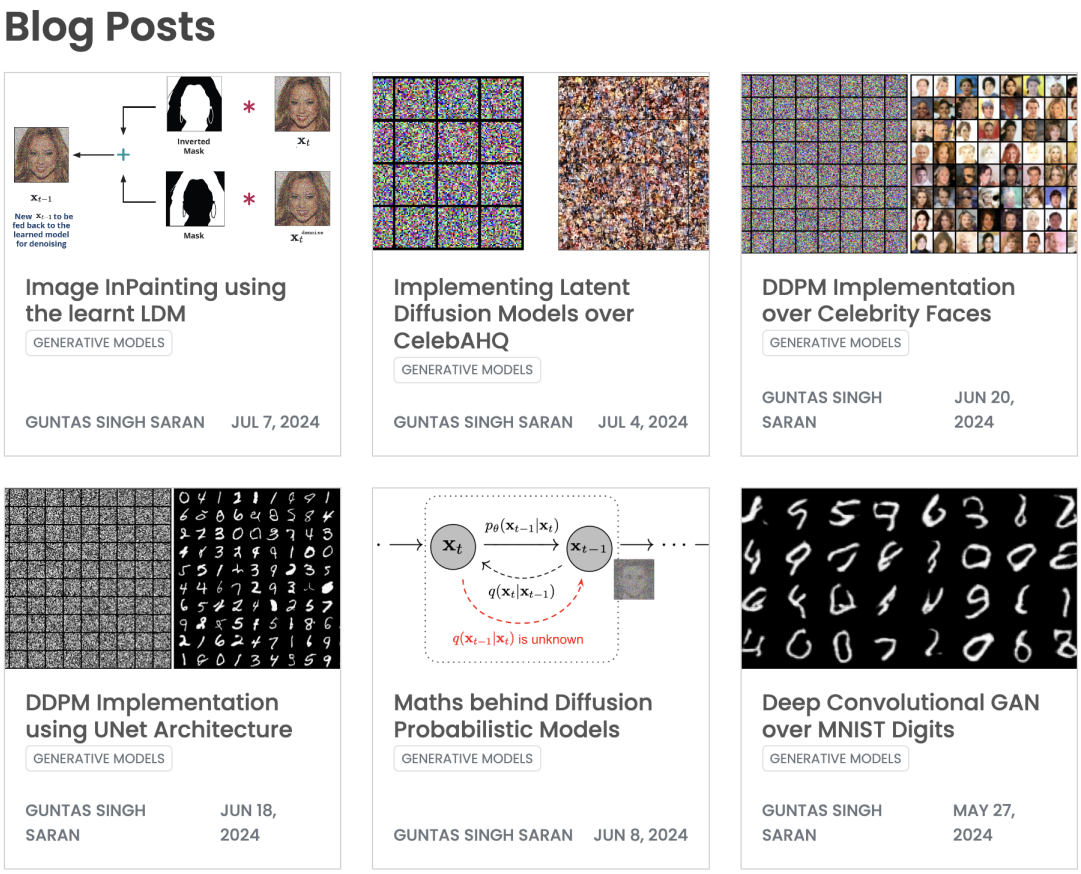


Figure 4.1: The Blog Page

4.2 Final Poster Presentation

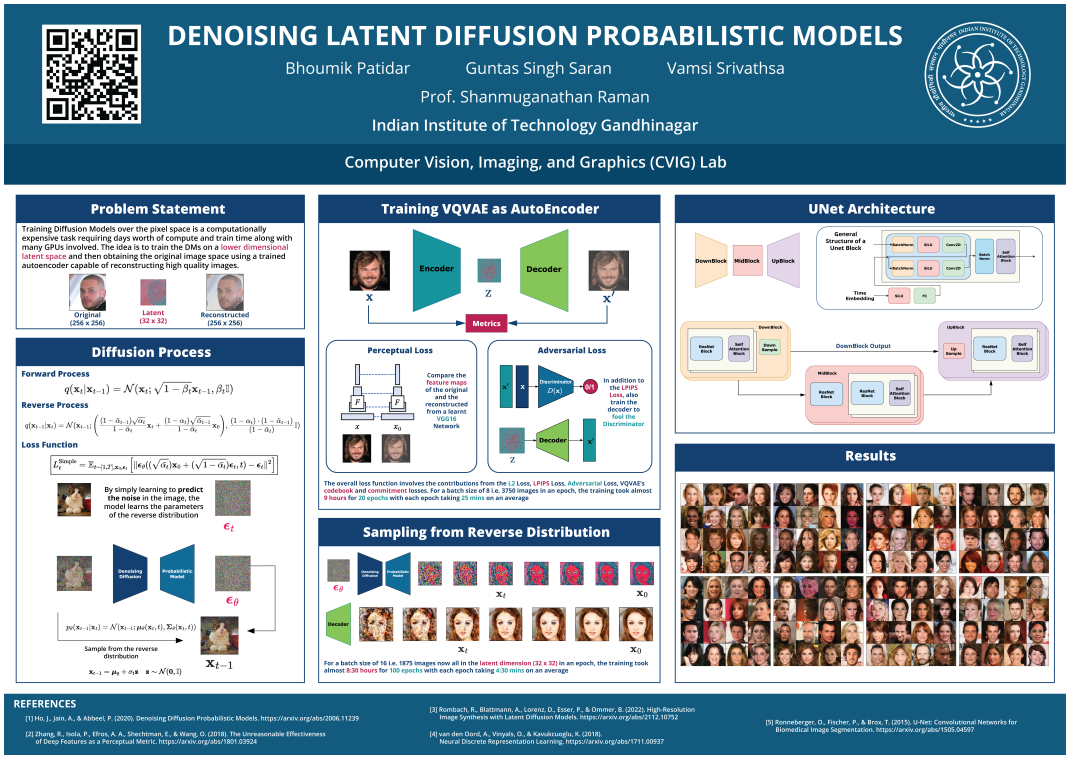


Figure 4.2: Latent Diffusion Model Poster

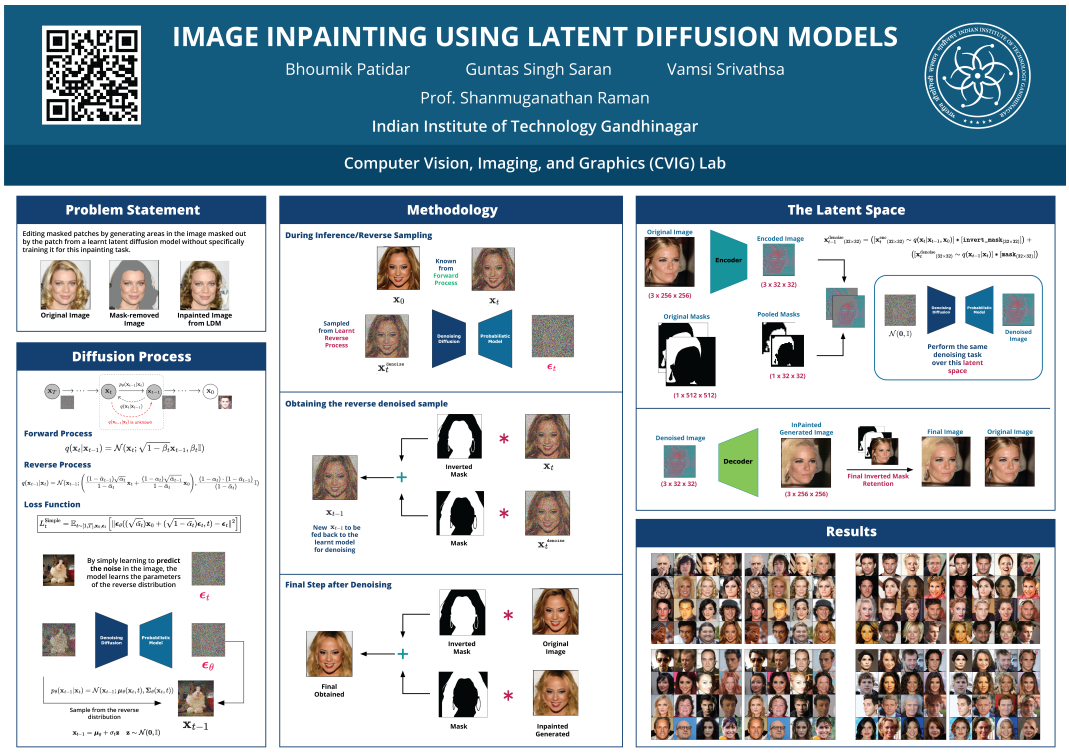


Figure 4.3: InPainting Poster

Bibliography

[1] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2022.

[2] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[3] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," 2020.

[4] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2016.

[5] L. Weng, "What are diffusion models?," *lilianweng.github.io*, Jul 2021.

[6] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," 2022.

[7] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015.

[8] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," 2018.

[9] J. Zhu, Y. Shen, D. Zhao, and B. Zhou, "In-domain gan inversion for real image editing," 2020.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[11] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," 2018.

[12] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," 2019.

[13] W. Xia, Y. Zhang, Y. Yang, J.-H. Xue, B. Zhou, and M.-H. Yang, "Gan inversion: A survey," 2022.

[14] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014.

[15] R. Mokady, A. Hertz, K. Aberman, Y. Pritch, and D. Cohen-Or, "Null-text inversion for editing real images using guided diffusion models," 2022.

[16] S. Li, T. Hu, F. S. Khan, L. Li, S. Yang, Y. Wang, M.-M. Cheng, and J. Yang, "Faster diffusion: Rethinking the role of unet encoder in diffusion models," 2023.

[17] H. Jin, Y. Li, F. Luan, Y. Xiangli, S. Bi, K. Zhang, Z. Xu, J. Sun, and N. Snavely, "Neural gaffer: Relighting any object via diffusion," 2024.

[18] R. Sajnani, J. Vanbaar, J. Min, K. Katyal, and S. Sridhar, "Geodiffuser: Geometry-based image editing with diffusion models," 2024.

[19] P. Nakkiran, A. Bradley, H. Zhou, and M. Advani, "Step-by-step diffusion: An elementary tutorial," 2024.

[20] A. Melnik, M. Miasayedzenkau, D. Makarovets, D. Pirshtuk, E. Akbulut, D. Holzmann, T. Renusch, G. Reichert, and H. Ritter, "Face generation and editing with stylegan: A survey," 2023.

[21] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Deep image homography estimation," 2016.

[22] F. Lu, S. Dong, L. Zhang, B. Liu, X. Lan, D. Jiang, and C. Yuan, "Deep homography estimation for visual place recognition," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, p. 10341–10349, Mar. 2024.

[23] M. Shao, T. Tasdizen, and S. Joshi, "Analyzing the domain shift immunity of deep homography estimation," 2023.

[24] R. Abdal, Y. Qin, and P. Wonka, "Image2stylegan: How to embed images into the stylegan latent space?," 2019.

[25] R. Gao, A. Holynski, P. Henzler, A. Brussee, R. Martin-Brualla, P. Srinivasan, J. T. Barron, and B. Poole, "Cat3d: Create anything in 3d with multi-view diffusion models," 2024.

[26] T. Karras, M. Aittala, T. Aila, and S. Laine, "Elucidating the design space of diffusion-based generative models," 2022.

[27] S. Khanna, P. Liu, L. Zhou, C. Meng, R. Rombach, M. Burke, D. Lobell, and S. Ermon, "Diffusionsat: A generative foundation model for satellite imagery," 2024.