Diffusion Models

Himesh Kumar Anant
SR No. 23775
himeshkumar@iisc.ac.in

Jithendra Rao Kasibhatla
SR No. 23646
jithendrarao@iisc.ac.in

Keval Pithadiya
SR No. 23756
kevalp@iisc.ac.in

Om Prakash Choudhary
SR No. 23618
omprakashc@iisc.ac.in

Abstract

Diffusion models are a class of generative models that learn to transform Gaussian noise into realistic samples that lie on the data manifold. There are three well-studied and mathematically equivalent training formulations: noise prediction, clean image prediction, and score-based learning. In this work, we train and empirically evaluate DDPMs based on the work of Ho et. al. [1] over three datasets to learn both noise prediction and clean image prediction and also look DDIMs introduced by Song et. al. [2] for a faster reverse process and evaluate them over different hyperparameter configurations. We also briefly explore score-based learning models.

1 Brief Introduction of Diffusion

The problem of generating realistic images is immensely challenging as it involves sampling meaningful data embedded in tiny clusters within a very high-dimensional image space. Clearly, developing an approach to solve this problem will require a range of clever ideas to make this problem tractable. In the DDPM approach, we model the problem as one of iteratively denoising an image in small steps, starting from pure noise, to obtain a meaningful image. Refer to A.

1.1 Denoising Diffusion Probabilistic Models

A DDPM model is a latent variable model with latents $\mathbf{x}_1, ..., \mathbf{x}_T$ of the same dimensionality as the data $\mathbf{x}_T$. The reverse process is defined as a Markov chain with learned Gaussian transitions $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$. The forward (diffusion) process is a fixed Markov chain which adds Guassian noise to the data such that $p(\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_t; \mathbf{0}, \mathbf{I})$.

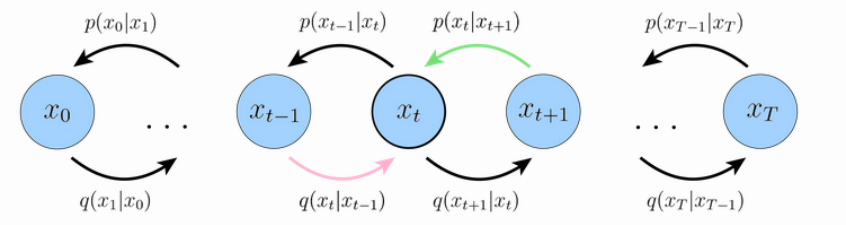


Figure 1: DDPM, adapted from [3]

Preprint. Under review.

- **Forward Process.** The forward process is fixed as $q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t;\ \sqrt{\alpha_t}\mathbf{x}_{t-1}, (1-\alpha_t)\mathbf{I})$ which admits a closed form $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\overline{\alpha_t}}\mathbf{x}_0, (1-\sqrt{\overline{\alpha_t}})\mathbf{I})$ allowing us to complete the forward process in one step.
- **Backward Process.** The ground truth denoising distribution $q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_q(\mathbf{x}_t, \mathbf{x}_0), \sigma_q^2(t)\mathbf{I})$.

1.1.1 Simplified Loss and Objectives

There are three different but equivalent objectives for training DDPM (Refer Appendix B):

- **Clean Image Prediction Interpretation** Network is trained to predict the original clean image $\mathbf{x}_0$ from a noisy version $\mathbf{x}_t$.
$$\arg\min_{\boldsymbol{\theta}} \|\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - \mathbf{x}_0\|^2$$
- **Noise Prediction Interpretation** Network learns to recover the exact noise $\boldsymbol{\epsilon}_0$ that was added to generate $\mathbf{x}_t$.
$$\arg\min_{\boldsymbol{\theta}} \|\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - \boldsymbol{\epsilon}_0\|^2$$
- **Score Interpretation** Network learns the direction to move in data space to increase its likelihood, guiding the denoising process.
$$\arg\min_{\boldsymbol{\theta}} \|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t \mid \mathbf{x}_0)\|^2$$

1.2 DDIM

DDIMs (Denoising Diffusion Implicit Models) are a family of diffusion models with non-Markovian forward and reverse processes.

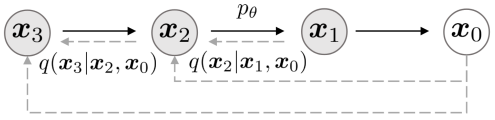


Figure 2: Denoising in DDIM, adapted from [2]

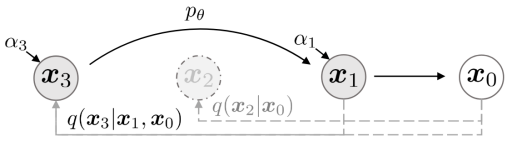


Figure 3: Subset sampling, adapted from [2]

- **Forward Process:** $q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t}\,\mathbf{x}_0, (1-\bar{\alpha}_t)\,\mathbf{I})$
- **Backward Process:** The reverse process is chosen to have this specific form:
$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}\left(\sqrt{\bar{\alpha}_{t-1}}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\cdot\frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0}{\sqrt{1-\bar{\alpha}_t}}, \sigma_t^2\,\mathbf{I}\right)$$
Under this assumption, $q(\mathbf{x}_t|\mathbf{x}_0) \sim \mathcal{N}(\mathbf{x}_t|\sqrt{\overline{\alpha_t}}\mathbf{x}_0, (1-\overline{\alpha_t})I)\ \forall\sigma_t$ (family) which is same as in DDPM.

Main Results

- **Unified Training** All models in the DDIM family (varying σ_t) share the **same** training objective—no retraining is required for different σ_t.
- **Subsampled Sampling** The training objective remains valid even if we use only a **subset of time steps** during sampling. This enables fast generation by **skipping steps**.
- **DDPM as a Special Case** DDPM can be recovered from DDIM by choosing a specific noise schedule. In practice, we set $\sigma_t^2 = \eta\frac{(1-\alpha_t)(1-\overline{\alpha}_{t-1})}{1-\overline{\alpha}_t}$ for a hyperparameter η, where $\eta = 1$ corresponds to the standard DDPM reverse process. This allows us to use a pretrained DDPM model (Markovian) and sample using DDIM, which is significantly **faster**.

2 Methodology

Our goal is to empirically compare the three mathematically equivalent formulations. To ensure a fair comparison, we keep the model architecture and training pipeline consistent across all experiments and modify only the training objective and sampling strategy. We used the Hugging Face `diffusers` library for training and sampling. All models are based on a standard UNet architecture, with minor changes to adapt to different loss functions.

2.1 Datasets and Image Resolutions

We trained models on the following datasets at various resolutions using an NVIDIA Tesla P100 GPU on Kaggle. All models were trained with 1000 diffusion time steps.

CIFAR-10 (32×32) [4] Trained with all three objectives on 50,000 train images: noise (80 epochs, ∼4 hours), clean (80 epochs, ∼4 hours), and score-based (30 epochs, ∼7 hours).

CelebA (32×32) [5] Trained with both noise and clean image objectives on 20,000 images (260 epochs, ∼8 hours).

CelebA (128×128) [5] Trained with both noise and clean image objectives on 20,000 images (30 epochs, ∼8 hours).

MNIST (28×28) [6] Trained with both noise and clean image objectives on 60,000 images (15 epochs, ∼30 minutes)

2.2 Evaluation Metrics

We compute the **Frechét Inception Distance** [7] which compares the distributions of the true and generated samples (lower is better) and **Inception Score** [8] which seeks to evaluate the image quality and diversity (higher is better). Refer to Appendix D for details on sampling and scoring.

3 Experiments

3.1 Comparison of DDPM, DDIM and The Three Training Objectives

We present our findings in table below, where for DDIM the stochasticity parameter $\eta = 0$. For CIFAR10, we see that noise prediction with DDPM works best while for CelebA, noise prediction with DDIM outperforms DDPM (especially for 128x128). It is also clear that clean image prediction consistently performs worse than noise prediction.

For MNIST, the FID and IS are in disagreement. This can be attributed to the fact that hand-written digits form a very structured data for which FID is not a suitable metric (See Appendix D). From looking at the samples however (Appendix E.4), it is evident that both clean image prediction and noise prediction with DDPM produce similar results while DDIM produces very noisy images.

A **Score Based Model** on CIFAR-10 gave **FID** = 126.02* and **IS** = 4.752*.

Sampling Type	CIFAR-10 (32x32)		CelebA (32x32)		CelebA (128x128)		MNIST (32x32)	
	FID	IS	FID	IS	FID	IS	FID	IS
DDPM Clean	73.56	5.0878	241.49	NA	146.80*	NA	63.54	9.2897
DDPM Noise	52.10	6.1290	163.27	NA	99.43*	NA	58.29	9.2236
DDIM Noise	63.92	6.1057	162.72	NA	76.69	NA	33.04	8.7317

Table 1: FID and IS scores for different sampling methods across datasets.

3.2 Comparison over DDIM Hyperparameters

We also measure the performance by varying the DDIM sampling parameters: inference steps n and stochasticity η and calculating FID and IS scores on CIFAR10. For sampling with lesser inference steps, DDIM dominates DDPM performance but as we increase the number of inference steps, DDPM outperforms DDIM along with a general improvement in image quality.

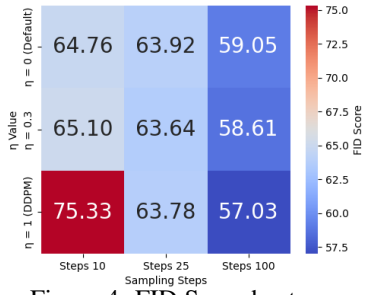


Figure 4: FID Score heatmap

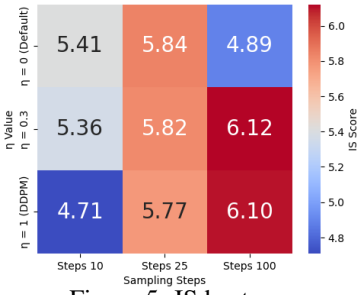


Figure 5: IS heatmap

Following samples are generated from DDPM and DDIM trained on CelebA and CIFAR-10 for qualitative analysis.

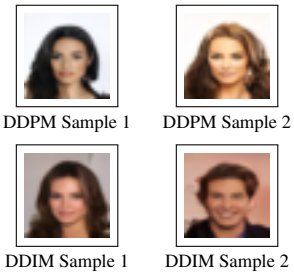


Figure 6: CelebA (32x32)

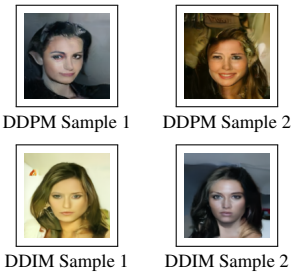


Figure 7: CelebA (128x128)

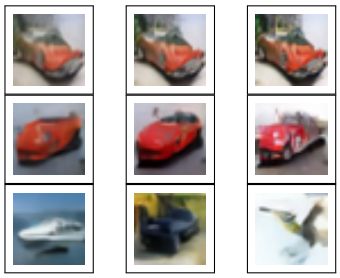


Figure 8: CIFAR-10 ($\eta = 0, 0.3, 1$)

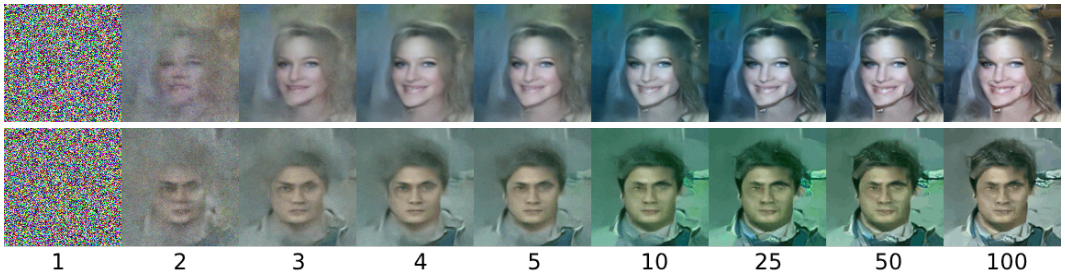


Figure 9: Sample Quality Progression with DDIM Inference Steps

3.3 Interpolation in Latent Space

We were also able to recreate latent interpolation to generate images with smooth interpolation between features such as pose, skin tone, hairstyle, expression and background. We linearly interpolate two image tensors $\mathbf{x}^{(0)}$ and $\mathbf{x}^{(1)}$ to get $\mathbf{x}^* = (1-s)\mathbf{x}^{(0)} + s\mathbf{x}^{(1)}$ where $s \in [0, 1]$.

(i) Interpolating $\mathbf{x}_0^{(0)}$ and $\mathbf{x}_0^{(1)}$ to obtain $\mathbf{x}_0^*$, adding noise to get $\mathbf{x}_{500}^*$, and then denoising to get $\hat{\mathbf{x}}_0^*$.

(ii) First we add noise to obtain $\mathbf{x}_{500}^{(0)}$ and $\mathbf{x}_{500}^{(1)}$, interpolate to get $\mathbf{x}_{500}^*$ and then denoise to get $\hat{\mathbf{x}}_0^*$.

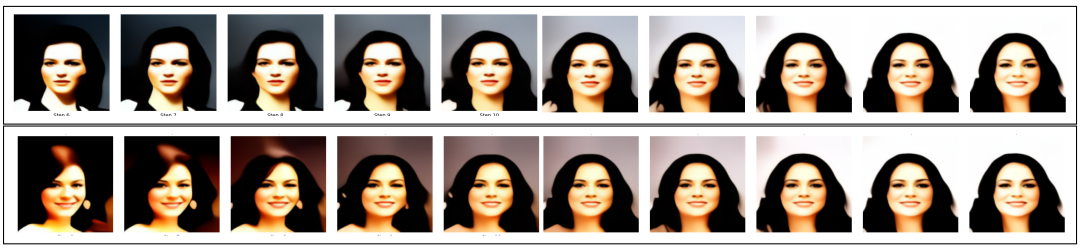


Figure 10: Interpolation on CelebA (128x128) using DDIM Sampling: (i) Above (ii) Below

4 Conclusion

Our experiments show that **noise learning** performs significantly better than **clean image learning**. This is not a mere coincidence—noise prediction is closely related to the *score-based interpretation* of diffusion models. In fact, it can be shown that $\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t) = -\frac{1}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_0$. Therefore, noise learning implicitly estimates the gradient of the data likelihood (up to a constant scaling factor). As a result, the reverse process can be interpreted as following the gradient ascent direction to move toward higher likelihood regions of the data distribution. Refer to Appendix C for more details.

The effectiveness of DDIM is evident from Figures 4 and 5. DDIM allows us to sample images much faster by only computing a fraction of the total inference steps without a drastic loss in quality. Furthermore, it affords us the freedom of making a reliable compromise between sample speed and sample quality by choosing the number of inference steps.

Acknowledgments

We are grateful to our mentor *Adit Vishnu* for his guidance and encouragement which motivated us to pursue this project with fervor. We also thank *Prof. Chiranjib Bhattacharyya* and *Prof. Shishir N. Y. Kolathaya* for providing us with this opportunity.

We appreciate the HuggingFace community for the `diffusers` library which was used for building our models and the Kaggle platform for their provision of high-end GPU accelerators to facilitate our model training.

Source Code

The source code used for building and training diffusion models as well as sampling images is available in the below linked GitHub repository. Download links to our trained models are available in the repository. It also includes the code used to evaluate IS for CIFAR10 and MNIST datasets.

`https://github.com/kevalpithadiya/Diffusion-Models`

References

[1] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising Diffusion Probabilistic Models". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 6840–6851. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

[2] Jiaming Song, Chenlin Meng, and Stefano Ermon. *Denoising Diffusion Implicit Models*. 2022. arXiv: 2010.02502 [cs.LG]. URL: https://arxiv.org/abs/2010.02502.

[3] Calvin Luo. *Understanding Diffusion Models: A Unified Perspective*. 2022. arXiv: 2208.11970 [cs.LG]. URL: https://arxiv.org/abs/2208.11970.

[4] Alex Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. Tech. rep. 2009. URL: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[5] Ziwei Liu et al. "Deep Learning Face Attributes in the Wild". In: *Proceedings of International Conference on Computer Vision (ICCV)*. Dec. 2015.

[6] Li Deng. "The mnist database of handwritten digit images for machine learning research". In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[7] Martin Heusel et al. *GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium*. 2018. arXiv: 1706.08500 [cs.LG]. URL: https://arxiv.org/abs/1706.08500.

[8] Tim Salimans et al. *Improved Techniques for Training GANs*. 2016. arXiv: 1606.03498 [cs.LG]. URL: https://arxiv.org/abs/1606.03498.

[9] Stanley H. Chan. *Tutorial on Diffusion Models for Imaging and Vision*. 2025. arXiv: 2403.18103 [cs.LG]. URL: https://arxiv.org/abs/2403.18103.

[10] Yang Song and Stefano Ermon. *Generative Modeling by Estimating Gradients of the Data Distribution*. 2020. arXiv: 1907.05600 [cs.LG]. URL: https://arxiv.org/abs/1907.05600.

[11] Maximilian Seitzer. *pytorch-fid: FID Score for PyTorch*. https://github.com/mseitzer/pytorch-fid. Version 0.3.0. Aug. 2020.

A Motivating Latent Spaces and Diffusion

A.1 Latent Space and Diffusion

Our objective is to sample images from the distribution from which our train data belongs. Diffusion achieves this by performing a process similar to the VAE one repeatedly. VAEs encode images in some other space, known as latent space (which is usually lower dimensional). Thus in order to achieve this, two models are simultaneously trained, one to convert an image from the image space into latent space and called the encoder and another to do the reverse called the decoder. Diffusion Models do something very similar but, as the name suggests we make small changes at each step to go from latent space (which is of the same dimension in this case). However, unlike in a traditional VAE, the forward process is fixed to make it such that smaller changes are made to the clearer image while larger changes are made to a noisier image. And thus only the decoder network has to be trained. The term diffusion comes from the fact that like diffusion, smaller changes happen when we reach the final time steps as compared to the initial denoising time steps.

A.2 ELBO

Since we want to maximize the log likelihood:-

$$\log(p(\boldsymbol{x})) = \log\left(\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\frac{p(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z}\mid\boldsymbol{x})}\right]\right)$$
$$\log(p(\boldsymbol{x})) \geq \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left(\log\left[\frac{p(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z}\mid\boldsymbol{x})}\right]\right)$$

The second term is called the ELBO (Evidence Lower Bound) and we minimize this since the first one is intractable.

The expression of ELBO for Variational Diffusion Models [9] is as follows:-

$$\mathrm{ELBO}_{\phi,\theta}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{x}_1|\mathbf{x}_0)}\left[\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\right] \underbrace{-\mathrm{D}_{\mathrm{KL}}\left(q_\phi(\mathbf{x}_T|\mathbf{x}_0)\parallel p(\mathbf{x}_T)\right)}_{\text{Nothing to train}}$$
$$-\sum_{t=2}^{T}\mathbb{E}_{q_\phi(\mathbf{x}_t|\mathbf{x}_0)}\left[\mathrm{D}_{\mathrm{KL}}\left(q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)\right)\right].$$

Using Bayes theorem it can be shown that

$$q_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}\mid\mu_q(\mathbf{x}_t,\mathbf{x}_0),\Sigma_q(t)) \quad \text{Where}$$

$$\mu_q(\mathbf{x}_t,\mathbf{x}_0) = \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{(1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_t}\mathbf{x}_0,$$
$$\Sigma_q(t) = \frac{(1-\alpha_t)(1-\sqrt{\bar{\alpha}_{t-1}})}{1-\bar{\alpha}_t}\mathbf{I} \stackrel{\text{def}}{=} \sigma_q^2(t)\mathbf{I},$$

The authors choose the form of $p_\theta(x_{t-1}|x_t)$ to be the following [9]

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}\left(\mathbf{x}_{t-1}\mid\underbrace{\mu_\theta(\mathbf{x}_t)}_{\substack{\text{neural}\\\text{network}}},\,\sigma_q^2(t)\mathbf{I}\right),$$

Using the formula of KL divergence between two gaussian we get the following ELBO

$$\begin{aligned}\text{ELBO}_\theta(\mathbf{x}) = \mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)}\left[\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\right] \\ - \sum_{t=2}^{T} \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[\frac{1}{2\sigma_q^2(t)}\left\|\mu_q(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t)\right\|^2\right],\end{aligned}$$

To get the ELBO for DDPM make the following substitutions

$$\mu_\theta(\mathbf{x}_t) \stackrel{\text{def}}{=} \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{(1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_t}\underbrace{\hat{\mathbf{x}}_\theta(\mathbf{x}_t)}_{\text{network}}.$$

we can simplify the reconstruction term to get

$$\log p_\theta(\mathbf{x}_0|\mathbf{x}_1) = -\frac{\|\mathbf{x}_0 - \mu_\theta(\mathbf{x}_1)\|^2}{2\sigma_q^2(1)} - \frac{d}{2}\log\left(2\pi\sigma_q^2(1)\right).$$

Now ELBO for DDPM is given by

$$\text{ELBO}_\theta(\mathbf{x}) = -\sum_{t=1}^{T}\frac{1}{2\sigma_q^2(t)}\cdot\frac{(1-\alpha_t)^2\bar{\alpha}_{t-1}}{(1-\bar{\alpha}_t)^2}\,\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[\|\hat{\mathbf{x}}_\theta(\mathbf{x}_t) - \mathbf{x}_0\|^2\right].$$

B Derivation of Equivalent Training Objectives

Objective 1: Prediction of $\mathbf{x}_0$

In the DDPM paper, the authors show that minimizing the variational bound leads to the following optimization objective:

$$\arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)} \left\|\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q\right\|_2^2 \tag{1}$$

From the DDPM reverse process, the true posterior mean is given by:

$$\boldsymbol{\mu}_q(x_t, x_0) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})x_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)x_0}{1-\bar{\alpha}_t} \tag{2}$$

Since $\boldsymbol{\mu}_{\boldsymbol{\theta}}(x_t, t)$ also conditions on x_t, we approximate it by:

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(x_t, t) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})x_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\hat{x}_{\boldsymbol{\theta}}(x_t, t)}{1-\bar{\alpha}_t} \tag{3}$$

Minimizing the squared error between $\boldsymbol{\mu}_{\boldsymbol{\theta}}$ and $\boldsymbol{\mu}_q$, we get:

$$\begin{aligned} &\arg\min_{\theta} \frac{1}{2\sigma_q^2(t)} \left\| \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)}{1-\bar{\alpha}_t} \left(\hat{x}_\theta(x_t, t) - x_0\right) \right\|_2^2 \\ &= \arg\min_{\theta} \frac{1}{2\sigma_q^2(t)} \frac{\bar{\alpha}_{t-1}(1-\alpha_t)^2}{(1-\bar{\alpha}_t)^2} \left\|\hat{x}_\theta(x_t, t) - x_0\right\|_2^2 \end{aligned} \tag{4}$$

Objective 2: Prediction of Noise $\boldsymbol{\epsilon}_0$

We use the reparameterization trick to express x_0 in terms of x_t and the noise:

$$x_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_0}{\sqrt{\bar{\alpha}_t}} \tag{5}$$

Substituting this into the mean expression gives:

$$\boldsymbol{\mu}_{\boldsymbol{q}}(x_t, x_0) = \frac{1}{\sqrt{\alpha_t}} x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}} \boldsymbol{\epsilon} \tag{6}$$

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(x_t, t) = \frac{1}{\sqrt{\alpha_t}} x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}} \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(x_t, t) \tag{7}$$

The resulting training objective becomes:

$$\begin{aligned} &\operatorname*{argmin}_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)} \left\| \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}} \left(\boldsymbol{\epsilon}_0 - \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(x_t, t)\right) \right\|_2^2 \\ &= \operatorname*{argmin}_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)} \frac{(1-\alpha_t)^2}{(1-\bar{\alpha}_t)\alpha_t} \left\|\boldsymbol{\epsilon}_0 - \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(x_t, t)\right\|_2^2 \end{aligned} \tag{8}$$

Objective 3: Score Matching

According to Tweedie's formula, the posterior mean of x_0 is:

$$\sqrt{\bar{\alpha}_t}x_0 = x_t + (1-\bar{\alpha}_t)\nabla \log p(x_t) \tag{9}$$

Solving for x_0:

$$x_0 = \frac{x_t + (1-\bar{\alpha}_t)\nabla \log p(x_t)}{\sqrt{\bar{\alpha}_t}} \tag{10}$$

Substituting into $\boldsymbol{\mu}_q(x_t, x_0)$, we derive:

$$\boldsymbol{\mu}_q(x_t, x_0) = \frac{1}{\sqrt{\alpha_t}}x_t + \frac{1-\alpha_t}{\sqrt{\alpha_t}}\nabla \log p(x_t) \tag{11}$$

Therefore, we can model the reverse mean as:

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(x_t, t) = \frac{1}{\sqrt{\alpha_t}}x_t + \frac{1-\alpha_t}{\sqrt{\alpha_t}}\, s_{\boldsymbol{\theta}}(x_t, t) \tag{12}$$

This gives the training objective:

$$\begin{aligned} &\underset{\boldsymbol{\theta}}{\operatorname{argmin}} \frac{1}{2\sigma_q^2(t)} \left\| \frac{1-\alpha_t}{\sqrt{\alpha_t}} \left(s_{\boldsymbol{\theta}}(x_t, t) - \nabla \log p(x_t)\right) \right\|_2^2 \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \frac{1}{2\sigma_q^2(t)} \frac{(1-\alpha_t)^2}{\alpha_t} \left\| s_{\boldsymbol{\theta}}(x_t, t) - \nabla \log p(x_t) \right\|_2^2 \end{aligned} \tag{13}$$

Note that $\nabla \log p(x_t)$ is the gradient taken in the vector space of the images and is thus intractable as we do not know it. It is approximated using the following [9]:-

$$q(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{h} K\left(\frac{\mathbf{x} - \mathbf{x_i}}{h}\right)$$

Where h is a parameter and K is a Kernel function which peaks at 0 and lesser at all other values. Essentially, each data point contributes to a peak in the estimated probability distribution and h just controls how smooth this distribution is.

These three objectives correspond respectively to predicting:

1. The clean image x_0,
2. The noise $\boldsymbol{\epsilon}_0$,
3. The score function $\nabla \log p(x_t)$.

Each provides a valid way of training a denoising diffusion model.

C More details on the Different Objectives

Noise prediction is the go to objective for most usecases. This is because, it has 2 major advantages over the clean image objective and the score function objective. As explained in [3], all three methods are equivalent (Loss functions imply each other), however, the noise objective is very similar to Langevin Sampling, which is equivalent to sampling from the underlying distributions as we will see. While, learning the clean image is probably worse since it leads to overfitting, the mechanism behind this is not entirely clear, however this hypothesis is empirically supported by both the authors of [1] since the clean objective generates worse images for a more complex dataset in CIFAR-10 as seen by the authors and reproduced by us. But for a simpler dataset in MNIST, the clean objective although had a higher FID indicative of less crisp images, but it had a slightly higher IS. This is probably because of overfitting working in its favor as discussed.

Now to understand why learning the noise is better than learning the score function. Let us first look at Langevin sampling. The problem of generating similar images can be boiled down to sampling from the underlying image distribution. As this distribution is very complicated, it is intractable. Langevin sampling is an algorithm for sampling from such complicated distribution, consider the following from [9]:-

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \tau \nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}_t) + \sqrt{2\tau}\boldsymbol{z}, \qquad \boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I}),$$

In this we are only using the score function (logarithm of the gradient) which can be trained. We train the logarithm of the gradient rather than the gradient itself for stability reasons, as explained in [9]. Intuitively, we start with $\boldsymbol{x}_0$ as Gaussian noise and move towards regions of higher probability along with some random noise. And this is equivalent to sampling from the underlying distribution given sufficient time steps. This is clear from the following figure:-

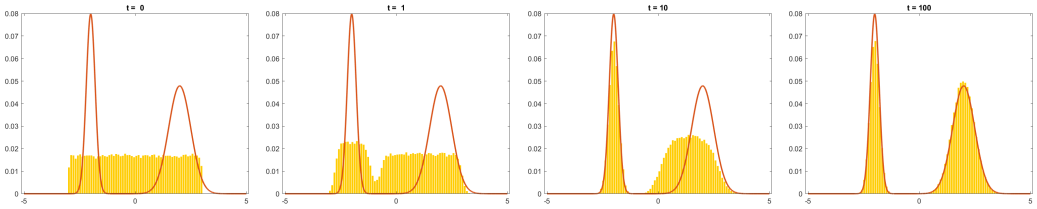


Figure 11: Adapted from [9]

Thus, clearly, starting from random noise, the samples mimic the underlying distribution after 100 time steps.

Now, the biggest issue with this is that the score loss function (in B) requires us to estimate the underlying distribution at each step in learning, which is very computationally intense. Thus, although it is exact, it is not preferred due to the computational cost. Other implicit methods also exist as discussed in [10], such as Sliced Score matching, however, these too suffer from similar computational costs, although they are an improvement.

Let us now see, how noise learning mimics the score objective. This was first discussed by the authors in [1]. Consider the reverse step in the noise approach:-

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}, \; where \; \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

Thus, clearly it resembles the Langevin sampling equation (note that t's order is reversed due to the definition). Hence, the noise objective is preferred as it is very similar to Langevin sampling (seen to be exact) without the extreme computational cost associated with it.

D Sampling and Scores

D.1 Sampling Methods

All DDIM sampling was performed with $\eta = 0.0$ and 50 inference steps unless stated otherwise. All scores were computed using equal sample size (5760) for both the true and generated images, except for the scores marked with * for which only 640 generated samples were used due to resource and time constraints.

D.2 Frechét Inception Distance

The Frechét Inception Distance[7] aims to compare the distributions of the true and generated images by extracting features from a pre-trained InceptionV3 model and computing the Frechét or Wasserstein-2 distance between them

$$\text{FID} = \|\mu_r - \mu_g\|^2 + \text{Tr}\left(\Sigma_r + \Sigma_g - 2\left(\Sigma_r\Sigma_g\right)^{\frac{1}{2}}\right)$$

The computation was done using `pytorch_fid` [11].

D.3 Inception Score

The Inception Score (IS) [8] is another useful metric to evaluate image quality. Unlike FID, which gives a measure of how close the generated image distribution is to the trained image distribution, IS gives a quantitative measure of how meaningful the images are (for images from labeled datasets such as CIFAR-10 or MNIST). Consider the following equation:-

$$\text{IS} = \exp\left(\mathbb{E}_{x\sim p_g} D_{KL}(p(y|\mathbf{x}) \,||\, p(y))\right) \approx \exp\left(\frac{1}{N}\sum_{i=1}^{N} D_{KL}(p(y|\mathbf{x_i}) \,||\, p(y))\right)$$

Where p_g refers to the probability distribution of the sampled images and $p(y|x)$ is the label distribution of some image x and $p(y)$ is the averaged out class distribution (Which are both estimated by a pre-trained classifier). This quantity is higher if a larger number of samples are classifiable, thus a higher IS is preferrable. Consider the image in 12, if the generated images are similar to this, they might have a relatively higher (worse) FID score due to the blurry nature of the image, however the images are more classifiable, hence will have a higher (better) IS score. Similarly for image in 13, the opposite holds, as it is not a meaningful digit although it is a crisp image.

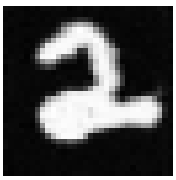


Figure 12: Bad Quality, Good Classifiability

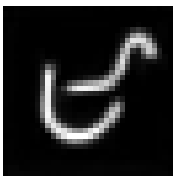
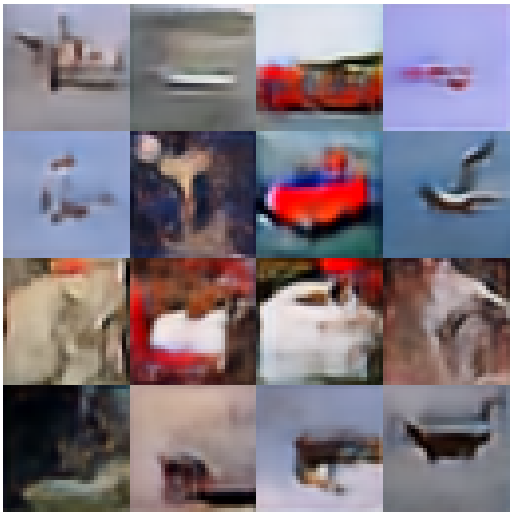


Figure 13: Good Quality, Bad Classifiability

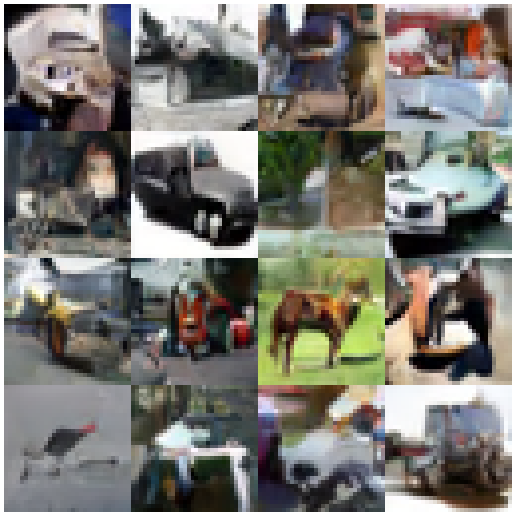
Thus, FID, in conjunction with IS comprehensively evaluates the quality of images generated by a model by covering both the actual crispness of the images along with their meaningfulness. For calculating IS for MNIST, it was trivial since training a classifier with 99+% accuracy was easy using a simple CNN and this was used to calculate it. However for CIFAR-10, since we compressed the images to 32×32, a deeper NN with accuracy of $\sim 95\%$ accuracy had to be used since the accuracy seemed to plateau at this value for more epochs and more parameters. Thus, this classifier was used for the calculation of IS. All the implementation details can be found in the code.

E Samples from All Models

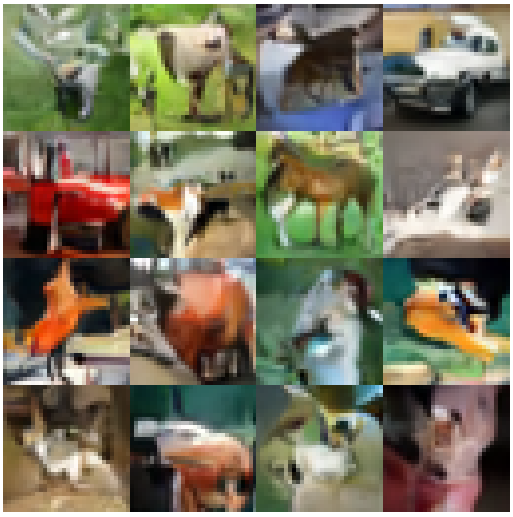
E.1 CIFAR10 (32x32)



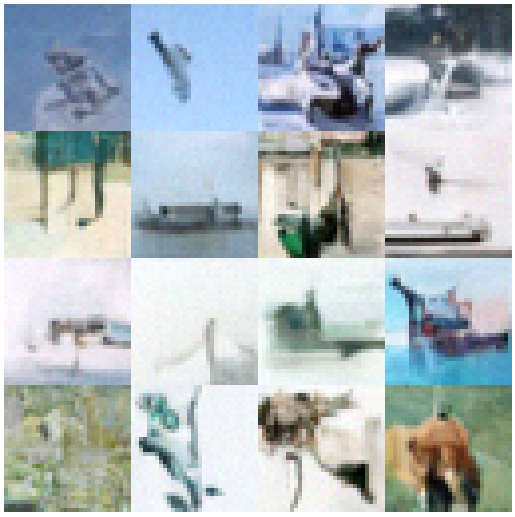
(a) Clean Image Prediction Model, DDPM Sampling



(b) Noise Prediction Model, DDPM Sampling

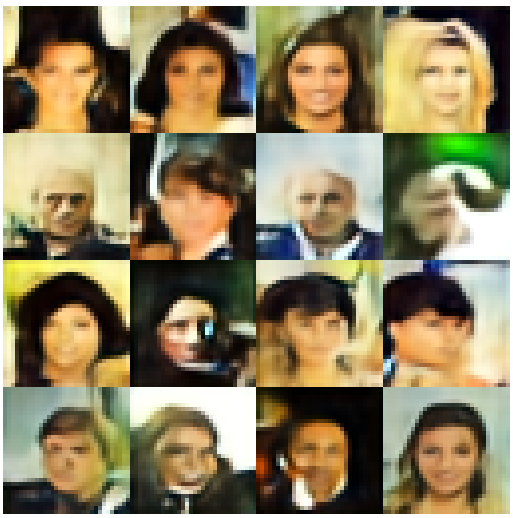


(c) Noise Prediction Model, DDIM Sampling

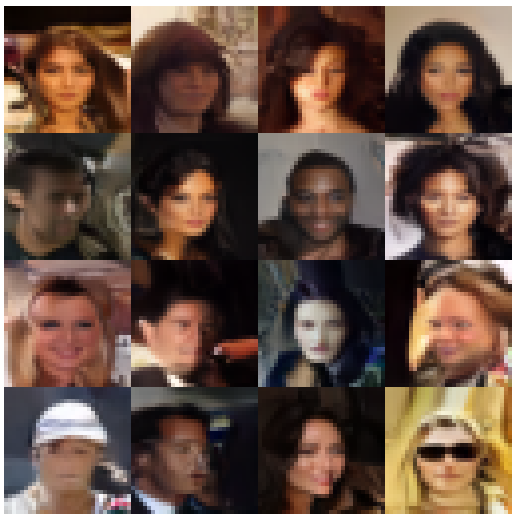


(d) Score-based Model and Sampling

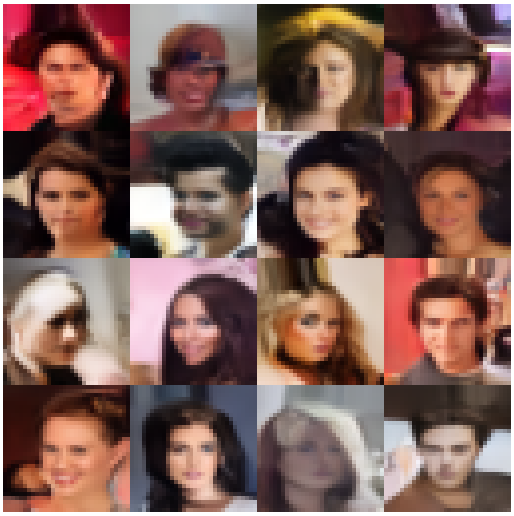
E.2 CelebA (32x32)



(a) Clean Image Prediction Model, DDPM Sampling



(b) Noise Prediction Model, DDPM Sampling



(c) Noise Prediction Model, DDIM Sampling

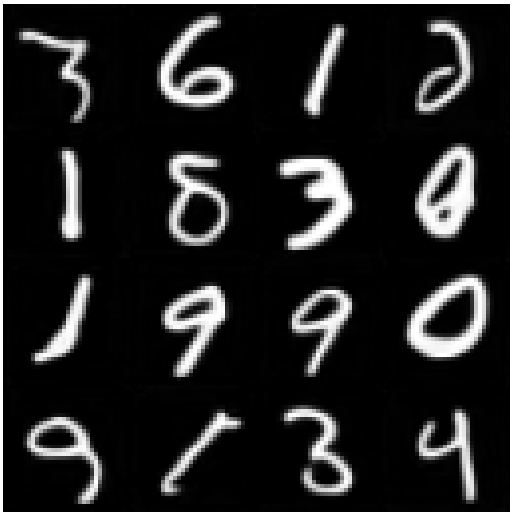
E.3 CelebA (128x128)

(a) Clean Image Prediction Model, DDPM Sampling

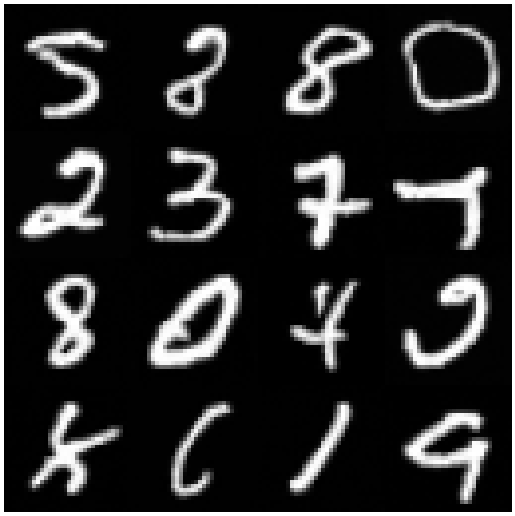
(b) Noise Prediction Model, DDPM Sampling

(c) Noise Prediction Model, DDIM Sampling

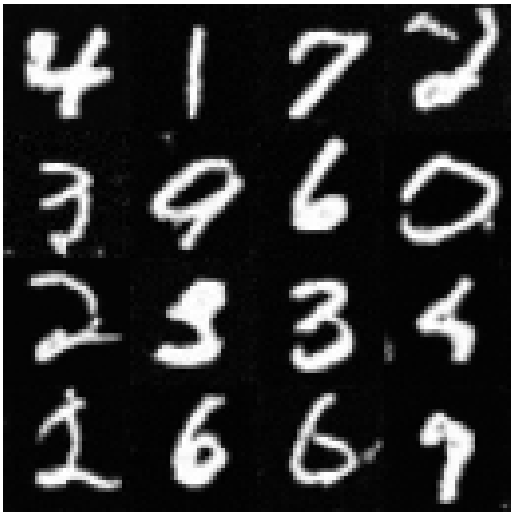
E.4 MNIST (32x32)



(a) Clean Image Prediction Model, DDPM Sampling



(b) Noise Prediction Model, DDPM Sampling



(c) Noise Prediction Model, DDIM Sampling

F Contributions

Shared Efforts
Covering the Literature, Running Model Training and Sampling

Om Prakash Choudhary
Modified Training Code to Learn Clean Image Interpretation and Score-based Model.
Modified Sampling Code for Reconstruction.

Keval Pithadiya
Sample Aggregation, FID Score Evaluation, Building Uniform Report

Himesh Kumar Anant
Implemented Interpolation, Sample Aggregation

Jithendra Rao Kasibhatla
IS Evaluation, Assessment of the Models