

DIFFUSION MODELS

Mollaev D.E.

May 16, 2023

PROJECT DESCRIPTION

- Datasets - MNIST, CIFAR10
- Model - DDPM (Denoising Diffusion Probabilistic Models) with Unet, IDDPMP, DDIM
- Metrics - FID, INCEPTION SCORE

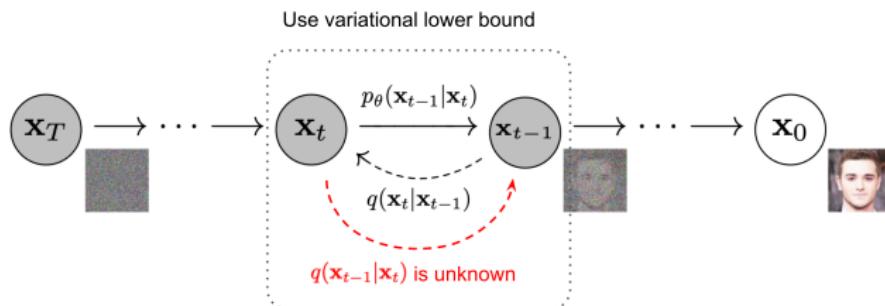
DDPM

FORWARD DIFFUSION PROCESS

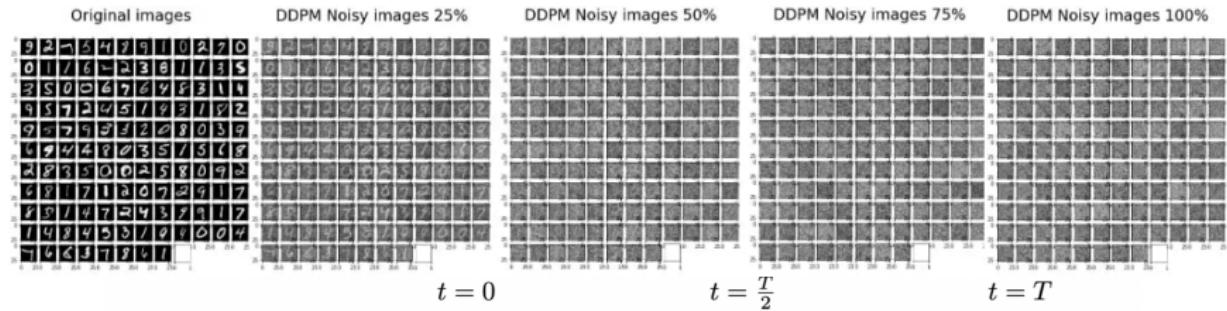
Let $x_0 \sim q(x_0)$, then we add small amount Gaussian noise to the sample in T steps. And we get sequence of noisy samples x_1, \dots, x_T
Then conditional probability

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

Eventually when $T \rightarrow \infty$, x_T is equivalent to an isotropic Gaussian distribution

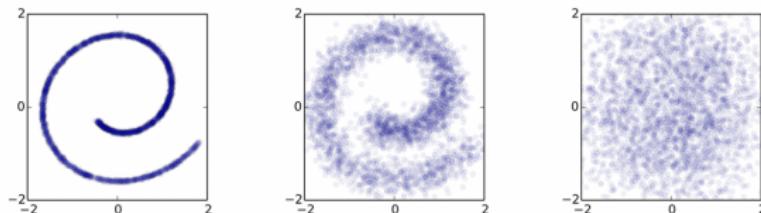


VISUALIZATION OF FORWARD PROCESS



The forward trajectory

$$q(\mathbf{x}_{0:T})$$



DDPM

REVERSE DIFFUSION PROCESS

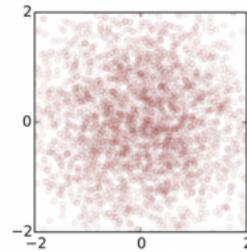
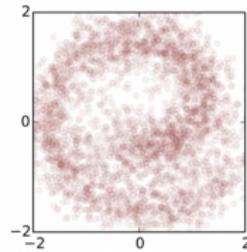
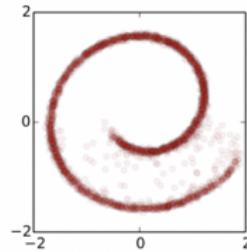
If we can reverse the forward process and sample from $q(x_{t-1}|x_t)$, we will be able to recreate the true sample from a Gaussian noise input, $x_T \sim \mathcal{N}(0, \mathcal{I})$.

But we cannot easily estimate $q(x_{t-1}|x_t)$ because it needs to use the entire dataset and therefore we need to learn a model(Unet) p_θ to approximate these conditional probabilities in order to run the reverse diffusion process.

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

The reverse trajectory

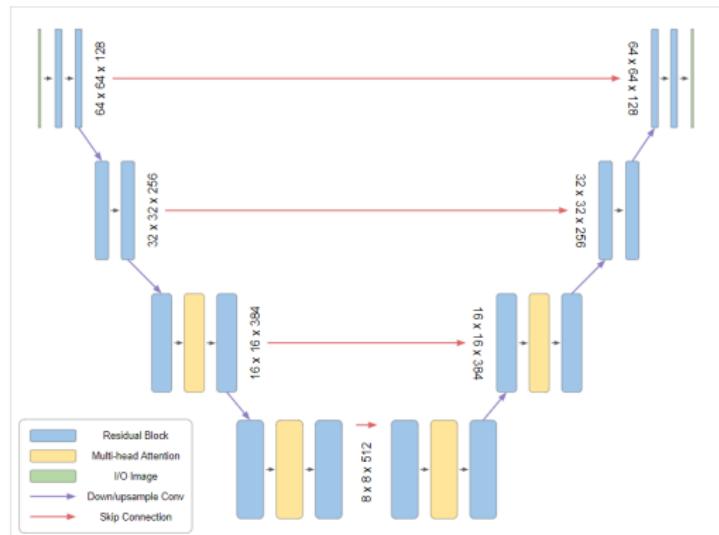
$$p_\theta(\mathbf{x}_{0:T})$$



[Link for visualization](#)

DDPM

UNET



ALGORITHMS

Algorithm 1 Training

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

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:   
$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\bar{\alpha}_t}} \left( \mathbf{x}_t - \frac{1 - \bar{\alpha}_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_{\theta}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$$

5: end for
6: return  $\mathbf{x}_0$ 
```

METRICS

FID

FID(Frechlet Inception Distance) is a performance metric that calculates the distance between the feature vectors of real images and the feature vectors of generate images

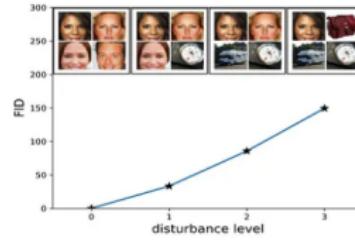
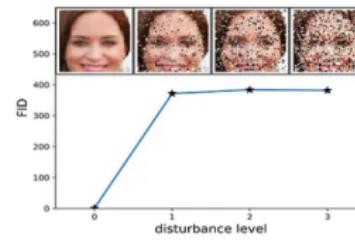
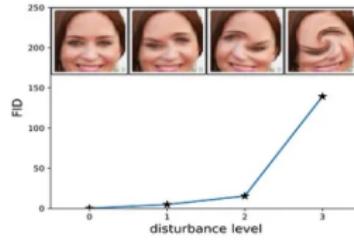
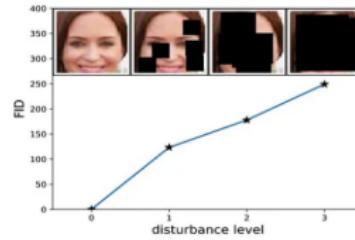
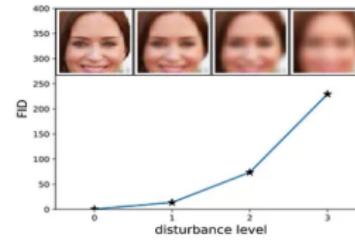
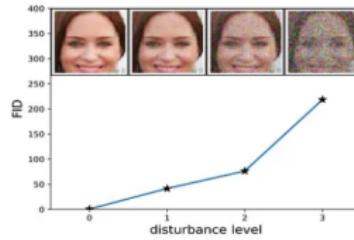
$$d^2((\mathbf{m}, \mathbf{C}), (\mathbf{m}_w, \mathbf{C}_w)) = \|\mathbf{m} - \mathbf{m}_w\|_2^2 + \text{Tr}(\mathbf{C} + \mathbf{C}_w - 2(\mathbf{C}\mathbf{C}_w)^{1/2})$$

How to calculate FID?

- 1 Use the Inception V2 pre-trained model to extract the feature vectors of real images and generated images by the generator
- 2 Calculate the feature-wise mean of the feature vectors generated in step 1
- 3 Generate the covariance matrices of the feature vectors — \mathbf{C}, \mathbf{C}_w
- 4 Calculate trace
- 5 Calculate the squared difference of the mean vectors calculated in step 2
- 6 Finally, add the output of step 4 and step 5

METRICS

FID



METRICS

INCEPTION SCORE

The Inception Score (IS) is an objective performance metric, used to evaluate the quality of generated images or synthetic images. It measures how realistic and diverse the output images are.

It measures two things:

- **Diversity** (Variety) — How diverse the generated images are — The entropy of the overall distribution should be high.
- **Quality** (Goodness) — How good the generated images are — Low entropy with high predictability is required.

METRICS

INCEPTION SCORE

$$\text{IS}(G) = \exp \left(\mathbb{E}_{\mathbf{x} \sim p_a} D_{KL}(p(y|\mathbf{x}) \| p(y)) \right)$$

- **Conditional Probability Distribution** — $p(y|x)$. It should be highly predictable and with low entropy. Here y is the set of labels and x is the image.
- **Marginal Probability Distribution** — $p(y)$

$$\int_z p(y|x = G(z)) dz$$

Here, $G(z)$ is the generated image by the generator model when provided with a latent vector. If the data distribution for y is uniform with high entropy, then the synthetic images will be diverse.

METRICS

INCEPTION SCORE

How to calculate IS?

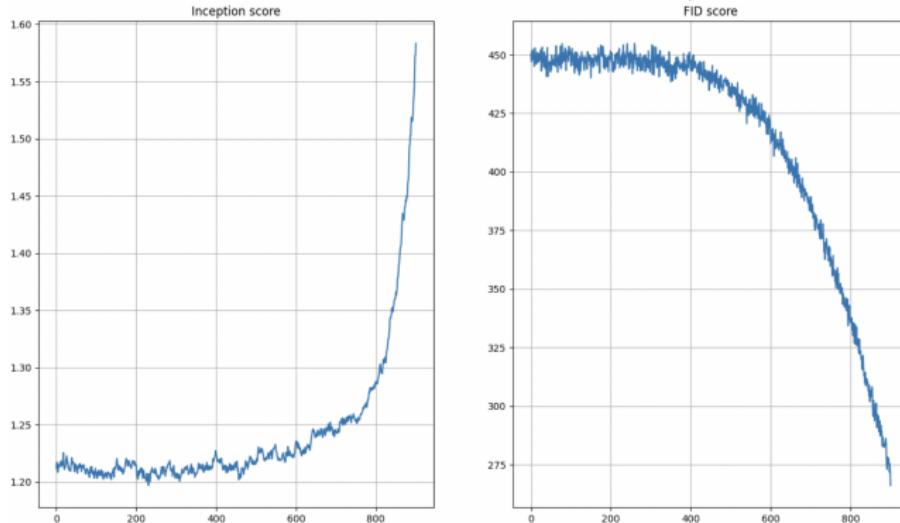
- 1 Pass the generated images through the Inception model to get the conditional label distribution $p(y|x)$
- 2 Calculate the marginal probability distribution $p(y)$
- 3 Calculate the KL Divergence between $p(y)$ and $p(y|x)$
- 4 Calculate the sum over classes and take the average of outputs over images
- 5 Finally, take the exponential of the averaged value.

RESULTS

DDPM, IS, FID

I trained three generators with 500, 1000, 2000 steps on dataset CIFAR10.

Metrics IS, FID for 1000 steps:

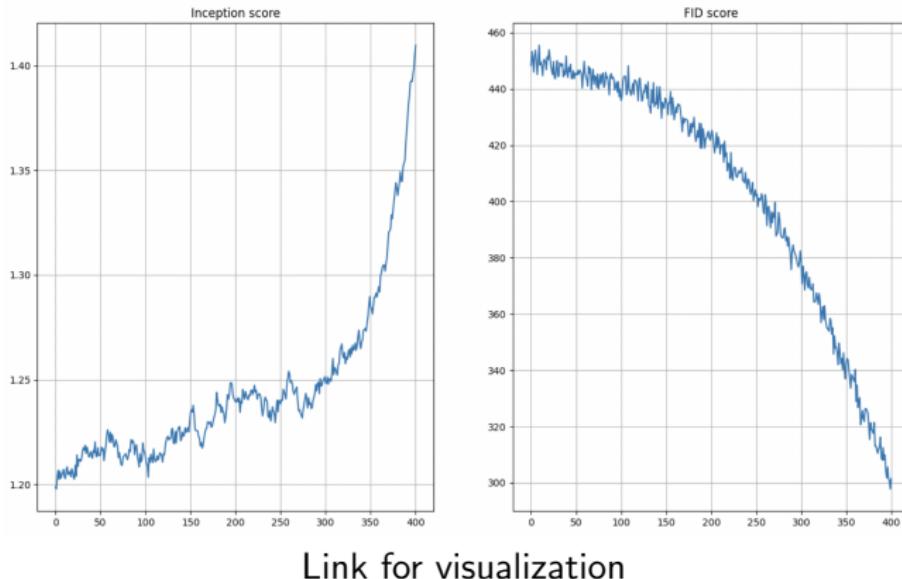


[Link for visualization](#)

RESULTS

DDPM, IS, FID

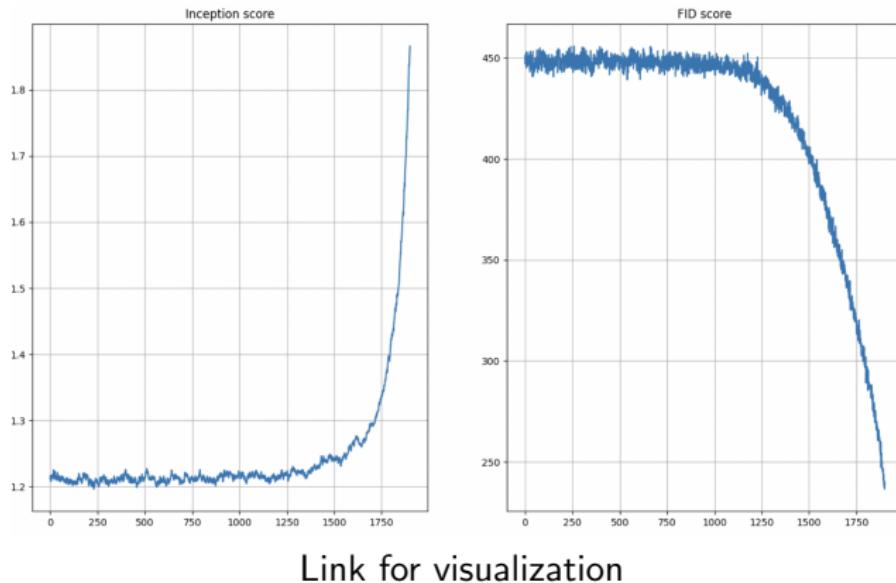
Metrics IS, FID for 500 steps:



RESULTS

DDPM, IS, FID

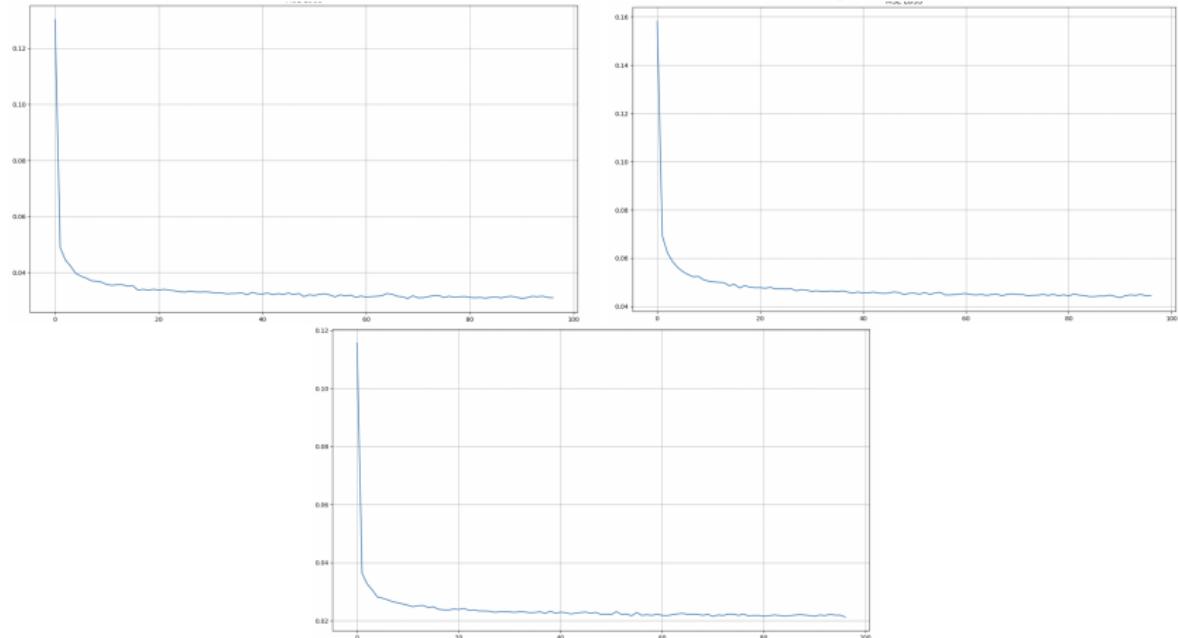
Metrics IS, FID for 2000 steps:



RESULTS

DDPM, NLL - VARIATIONAL LOWER BOUND

ELBO for 1000, 500, 2000 steps



IDDPM

The improving DDPMs paper had a couple of methods to improve the score:

- 1 Learn $\Sigma_\theta(x_t)$, the variance of the predicted normal distribution instead of keeping it fixed at β_t .
- 2 Change the learning rate scheduler defined as a linear β_t interpolation between 10^{-4} and 0.02 to a cosine $\bar{\alpha}_t$ interpolation.

DDIM

One problem with the DDPM process is the speed of generating an image after training. The DDIM paper introduces a way to speed up image generation with little image quality tradeoff.

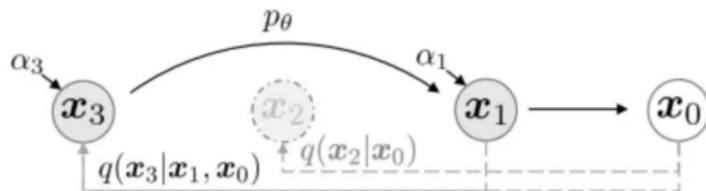


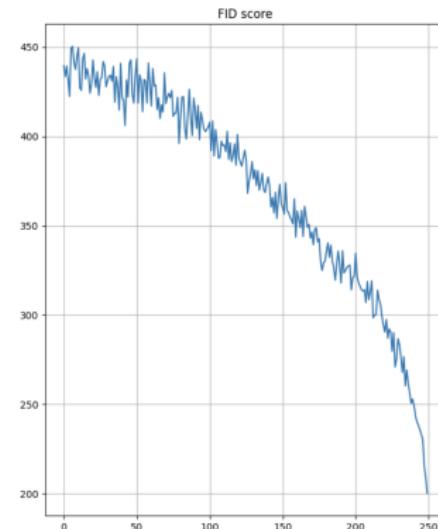
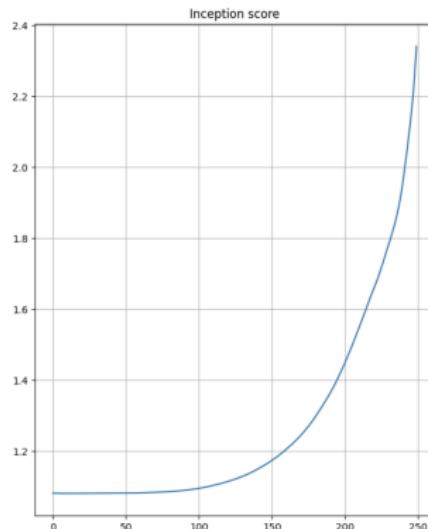
Figure 2: Graphical model for accelerated generation, where $\tau = [1, 3]$.

It does so by redefining the diffusion process as a non-Markovian process.

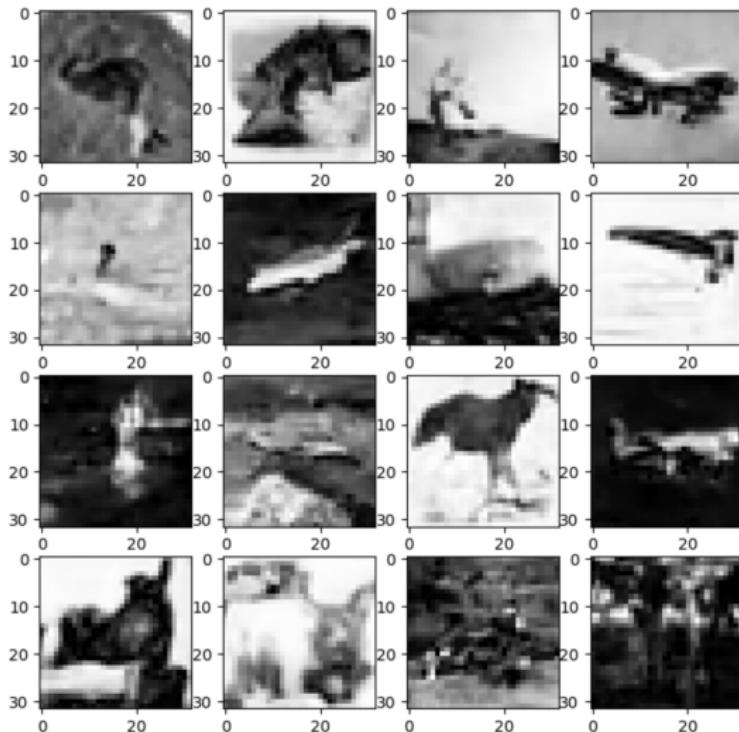
RESULTS

IDDPM

IS, FID for 1000 training steps with 250 inference steps



VISUALIZATION



CLASSIFIER GUIDANCE

ALGORITHM

Classifier guidance was introduced in the paper “Diffusion Models Beat GANs on Image Synthesis” and essentially uses a classifier to guide the diffusion model to generate images of a desired class.

Algorithm 1 Classifier guided diffusion sampling, given a diffusion model $(\mu_\theta(x_t), \Sigma_\theta(x_t))$, classifier $p_\phi(y|x_t)$, and gradient scale s .

Input: class label y , gradient scale s

$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$

for all t from T to 1 **do**

$\mu, \Sigma \leftarrow \mu_\theta(x_t), \Sigma_\theta(x_t)$

$x_{t-1} \leftarrow$ sample from $\mathcal{N}(\mu + s\Sigma \nabla_{x_t} \log p_\phi(y|x_t), \Sigma)$

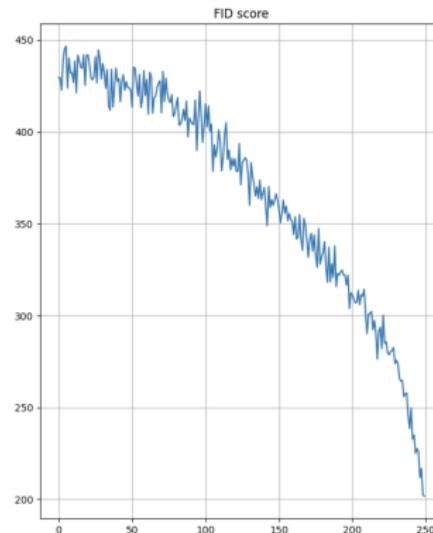
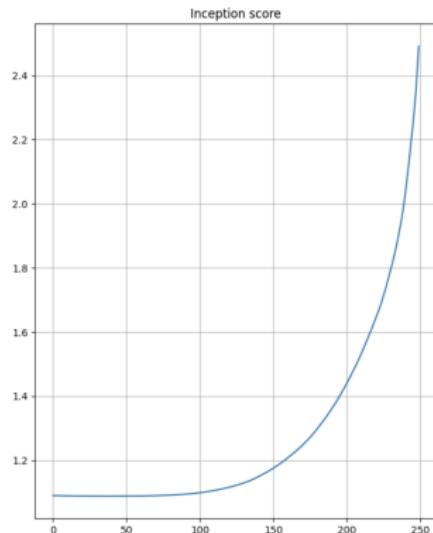
end for

return x_0

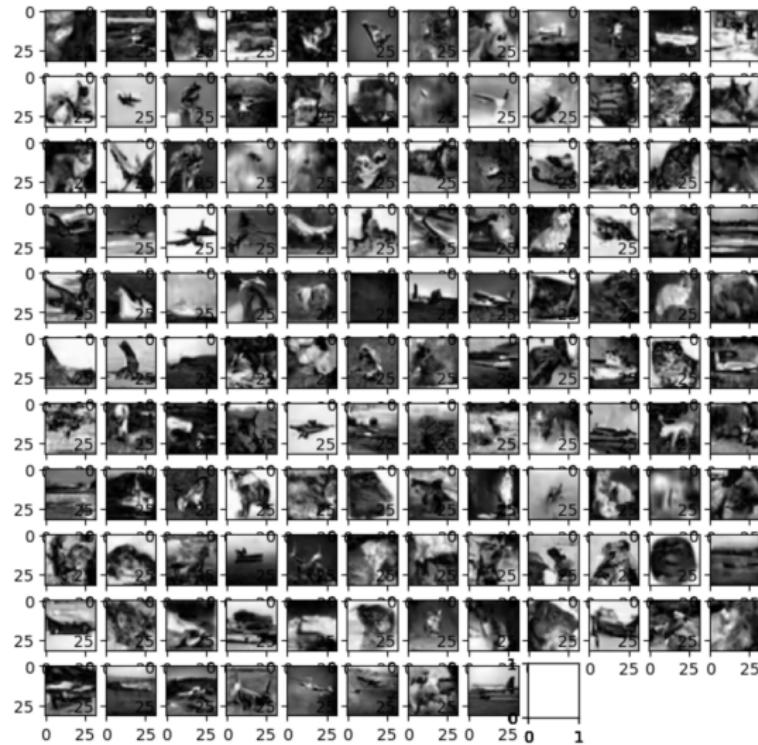


CLASSIFIER GUIDANCE

RESULTS - IS, FID



VISUALIZATION



CLASSIFIER-FREE GUIDANCE

Classifier-Free Guidance improves classifier guidance by eliminating the classifier while still providing class guidance to the model.

Algorithm 1 Joint training a diffusion model with classifier-free guidance

Require: p_{uncond} : probability of unconditional training

1: repeat

2: $(\mathbf{x}, \mathbf{c}) \sim p(\mathbf{x}, \mathbf{c})$ ▷ Sample data with conditioning from the dataset
 3: $\mathbf{c} \leftarrow \emptyset$ with probability p_{uncond} . ▷ Randomly discard conditioning to train unconditionally

$\lambda \sim p(\lambda)$ ▶ Sample log SNR value

$$5: \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

$$6: \quad z_\lambda = \alpha_\lambda x +$$

7: Take gradient step on $\nabla_{\theta} \|\epsilon_{\theta}(z_1, c) - \hat{\epsilon}\|^2$ ▷ Optimization of denoising model

8: until converged

— 5 —

Training a diffusion model for classifier-free guidance

I obtained IS = 3.16, FID = 184

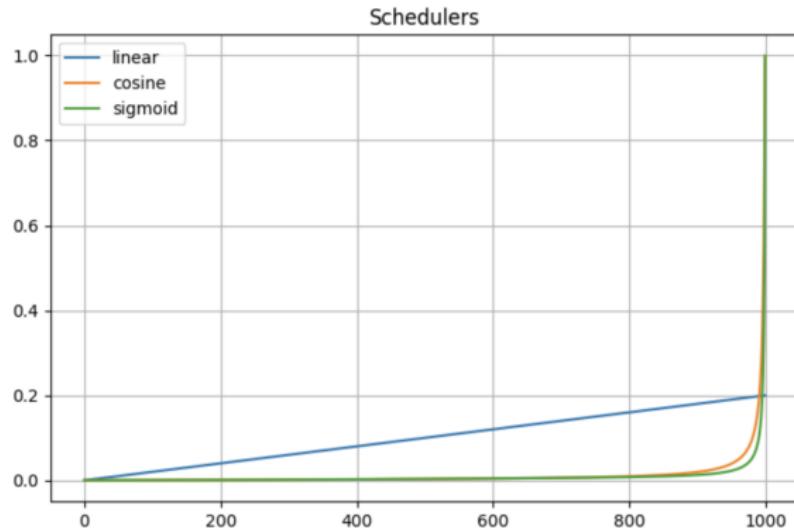
VISUALIZATION



Link for MNIST vizualization

BETAS SCHEDULERS

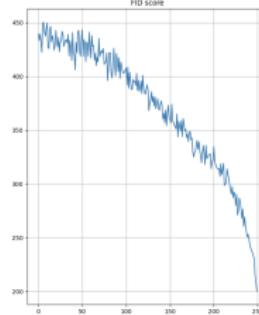
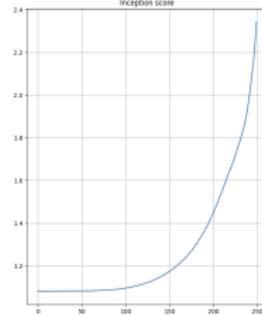
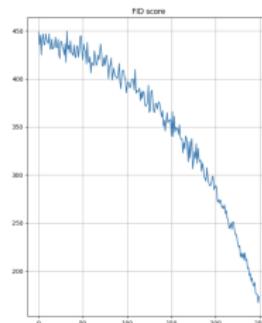
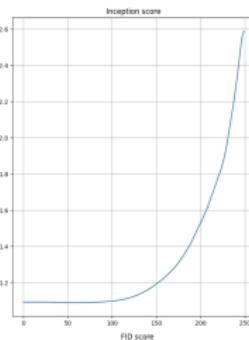
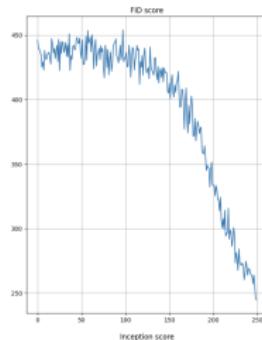
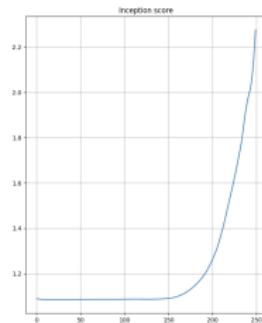
- Linear
- Cosine
- Sigmoid



BETAS SCHEDULERS

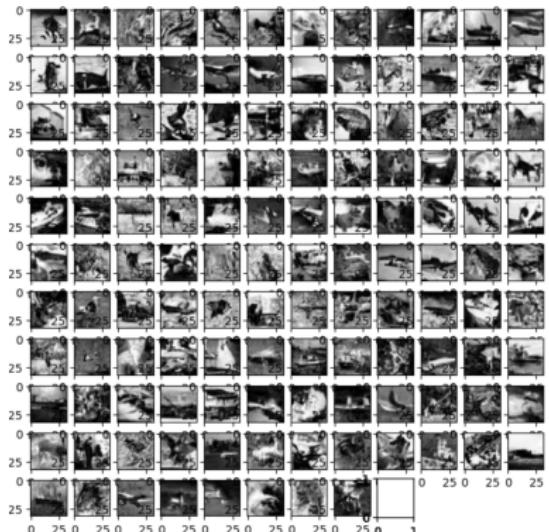
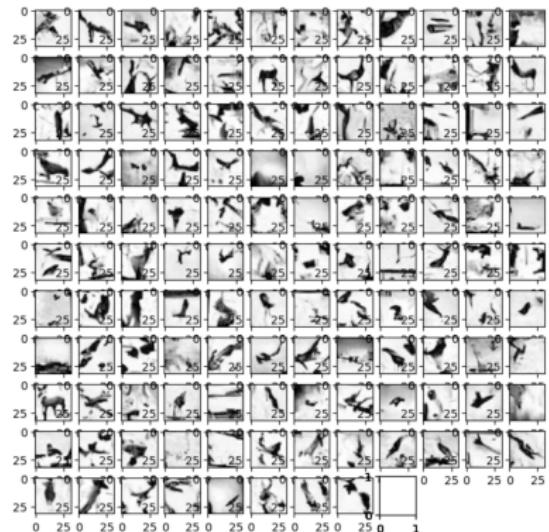
RESULTS

Linear, Cosine and Sigmoid scheduler:



VISUALIZATION

Linear, Cosine scheduler:



SOURCES

Links:

- Denoising Diffusion Probabilistic Models
- Denoising Diffusion Implicit Models
- Improved Denoising Diffusion Probabilistic Models
- Lil'Log What are Diffusion Models?
- Medium: Generating images with DDPMs: A PyTorch Implementation
- Medium: A Very Short Introduction to Inception Score(IS)
- Medium: A Very Short Introduction to Frechlet Inception Distance(FID)
- Special course on Mechmath MSU: Introduction to Machine and Deep Learning Theory