Diffusion Models

Mollaev D.E., 1st year of Phd, MechMath MSU, Differential eometry and Its Applications department

May 17, 2023

Project description

- Datasets - MNIST, CIFAR10
- Model - DDPM (Denoising Diffusion Probabilistics Models) with Unet, IDDPM, DDIM
- Metrics - FID, INCEPTION SCORE
- Classifier guidance, Classifier-free guidance
- Betas schedulers - linear, cosine, sigmoid

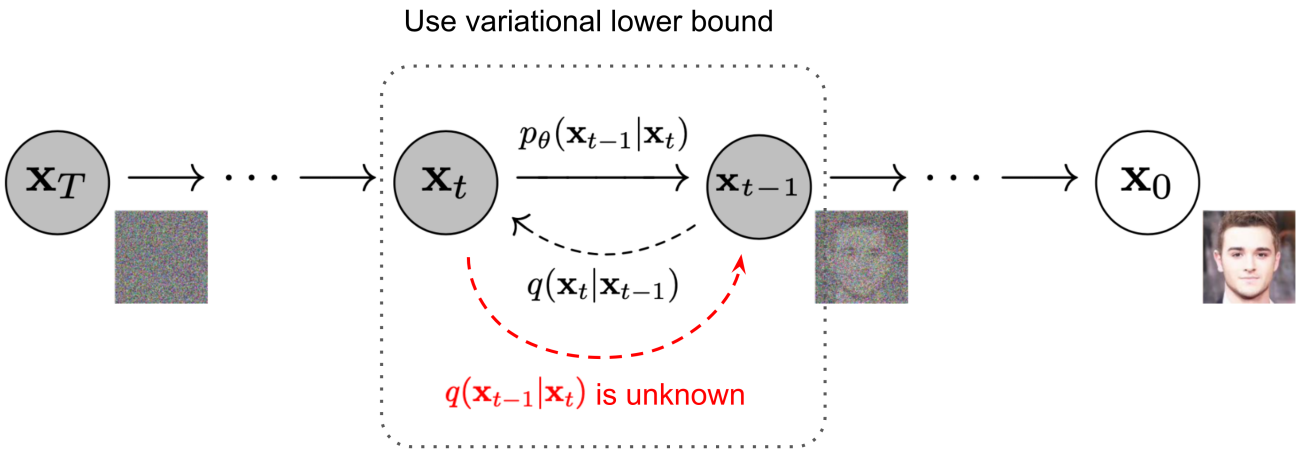
DDPM

Forward diffusion process

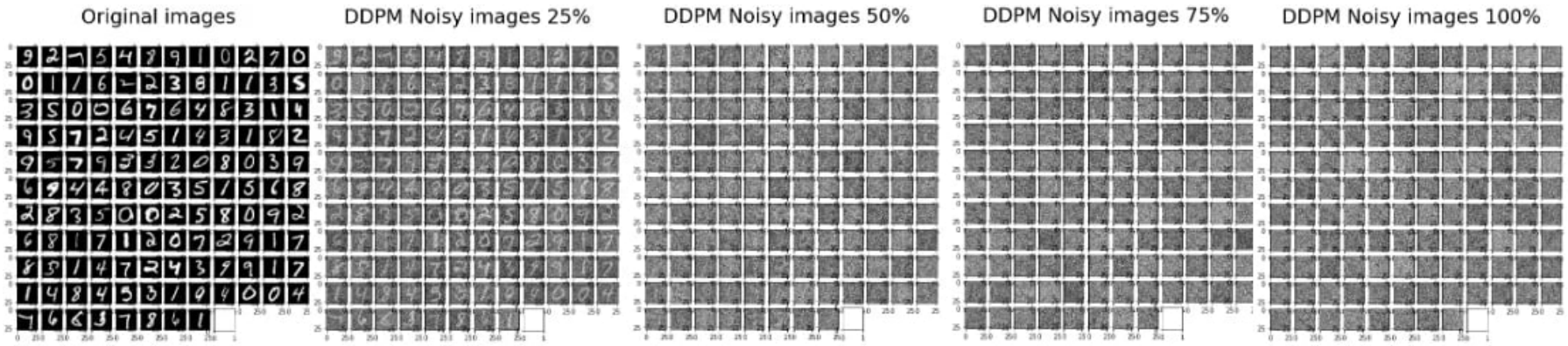
Let $x_0 \sim q(x_0)$, then we add small amount Gaussian noise to the sample in T steps. And we get sequence of noisy samples $x_1, ..., x_T$
Then conditional probability

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

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

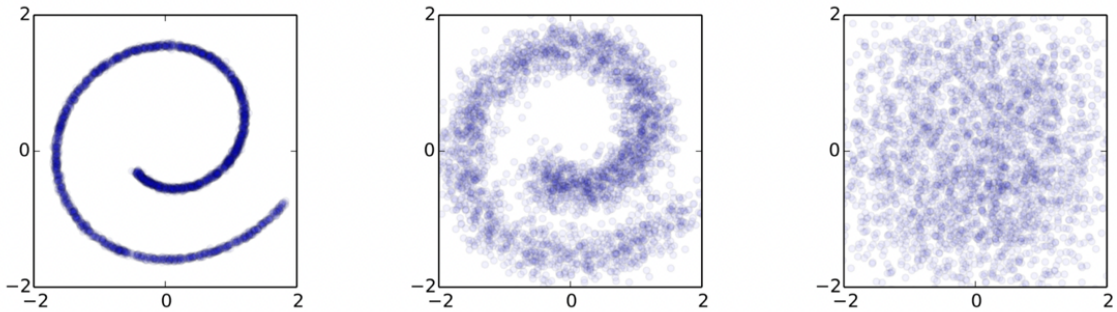


Visualization of forward process



The forward trajectory

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

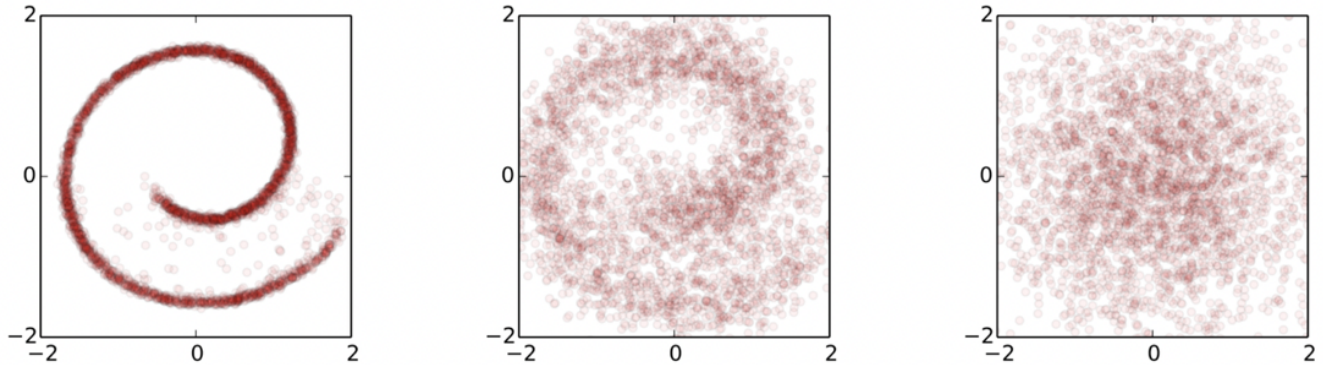


If we can reverse the forward process and sample from $q(x_{t-1}|x_t)$, we will be able to recreate the true sample from a Gaussian noise input, $x_T \sim \mathcal{N}(0, \mathcal{I})$.
But we cannot easily estimate $q(x_{t-1}|x_t)$ because it needs to use the entire dataset and therefore we need to learn 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

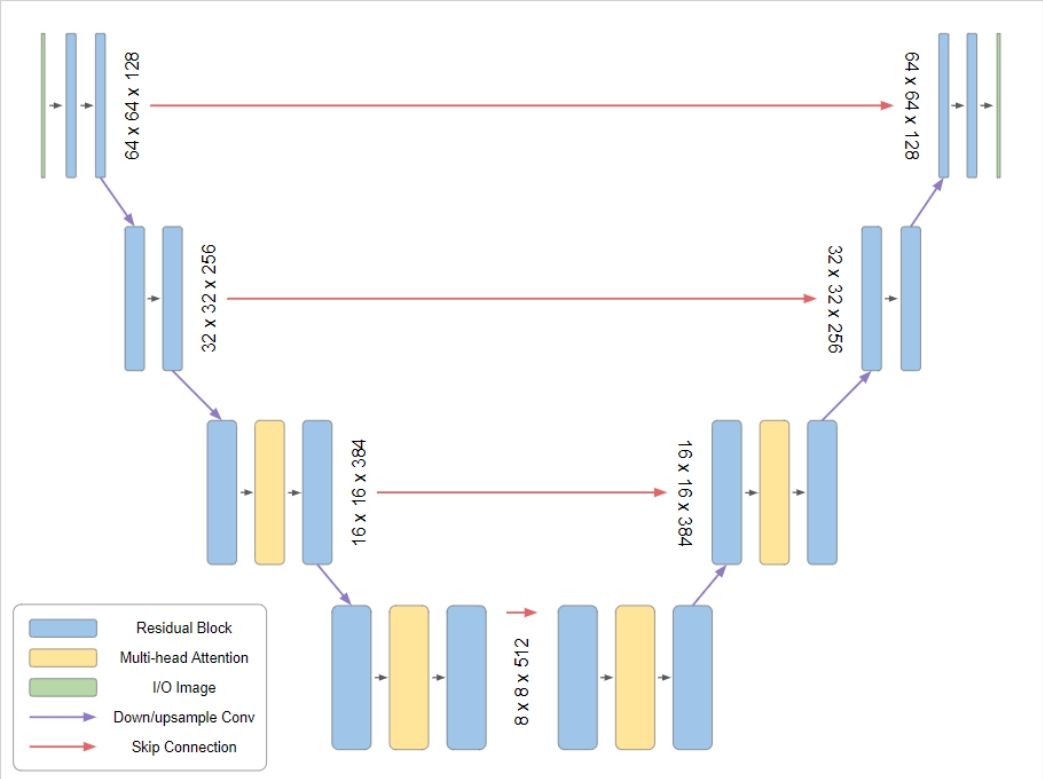
$p_\theta(\mathbf{x}_{0:T})$



Link for visualization

DDPM

Unet


64 x 64 x 128
64 x 64 x 128
32 x 32 x 256
32 x 32 x 256
16 x 16 x 384
16 x 16 x 384
8 x 8 x 512
Residual Block
Multi-head Attention
I/O Image
Down/upsample Conv
Skip Connection

Algorithms

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** $\mathbf{x}_0$

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

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

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 — C, 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

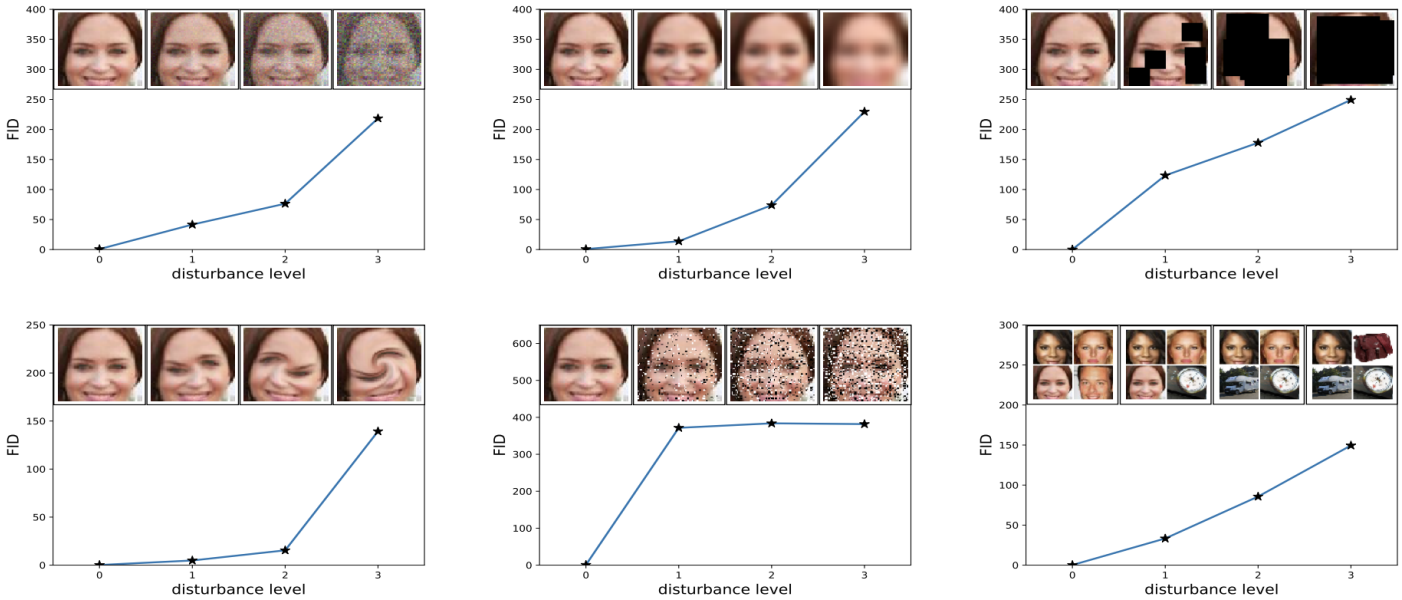


Figure 3: FID is evaluated for **upper left:** Gaussian noise, **upper middle:** Gaussian blur, **upper right:** implanted black rectangles, **lower left:** swirled images, **lower middle:** salt and pepper noise, and **lower right:** CelebA dataset contaminated by ImageNet images. The disturbance level rises from zero and increases to the highest level. The FID captures the disturbance level very well by monotonically increasing.

Paper: GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium

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.

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

- **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.

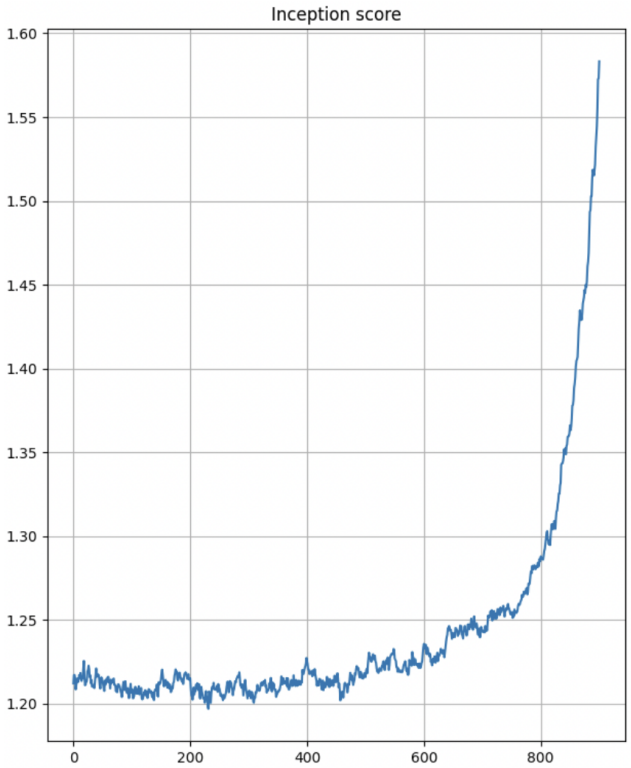
How to calculate IS?

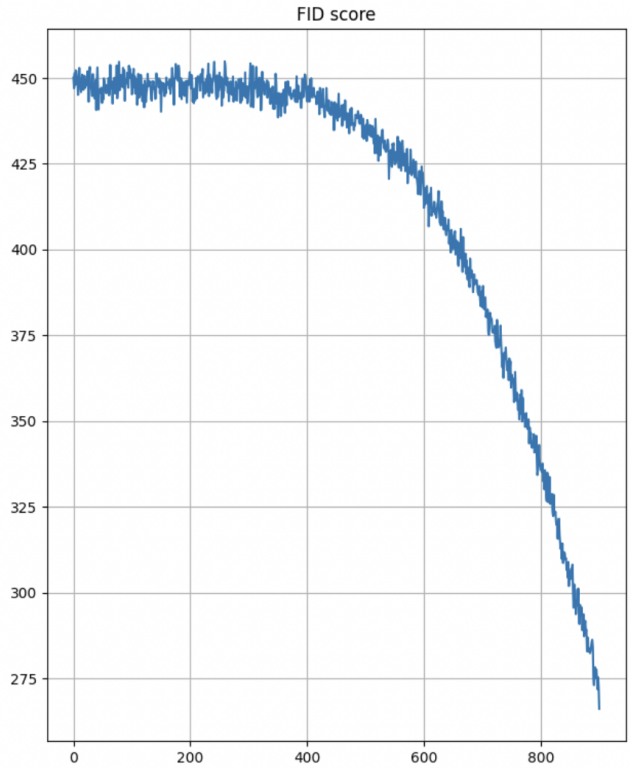
1. Pass the generated images through the Inception model to get the conditional label distribution $p(y|x)$
2. Calculate the marginal probability distribution $p(y)$
3. Calculate the KL Divergence between $p(y)$ and $p(y|x)$
4. Calculate the sum over classes and take the average of outputs over images
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. Here and beyond 100 epochs for training on the CIFAR10 dataset

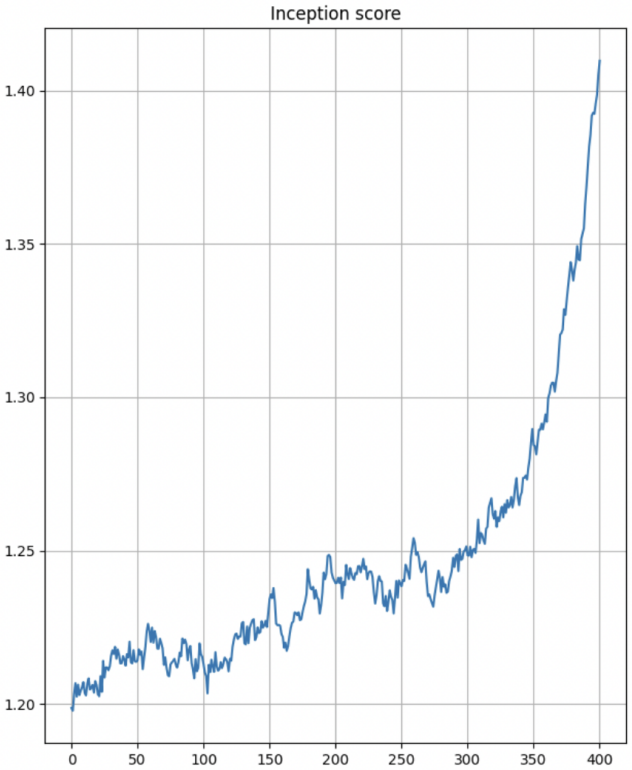
Metrics IS, FID for 1000 steps:

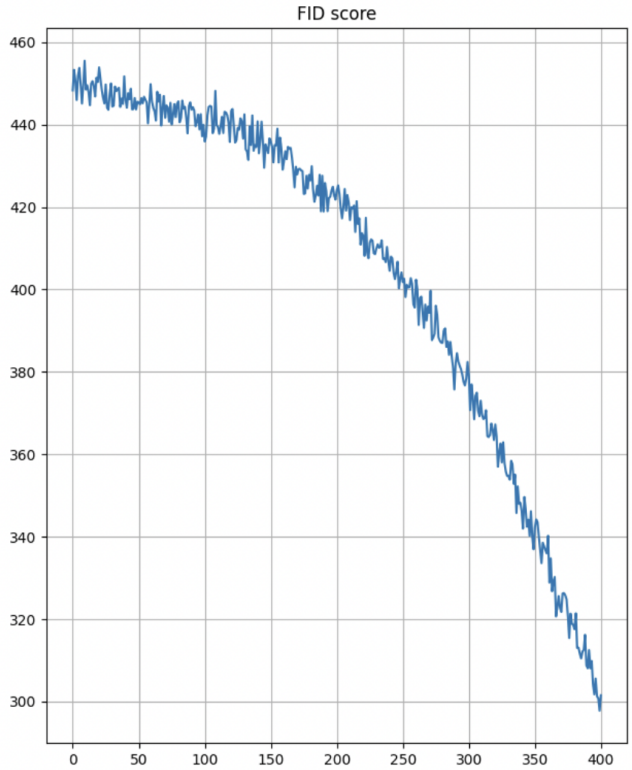




Link for visualization

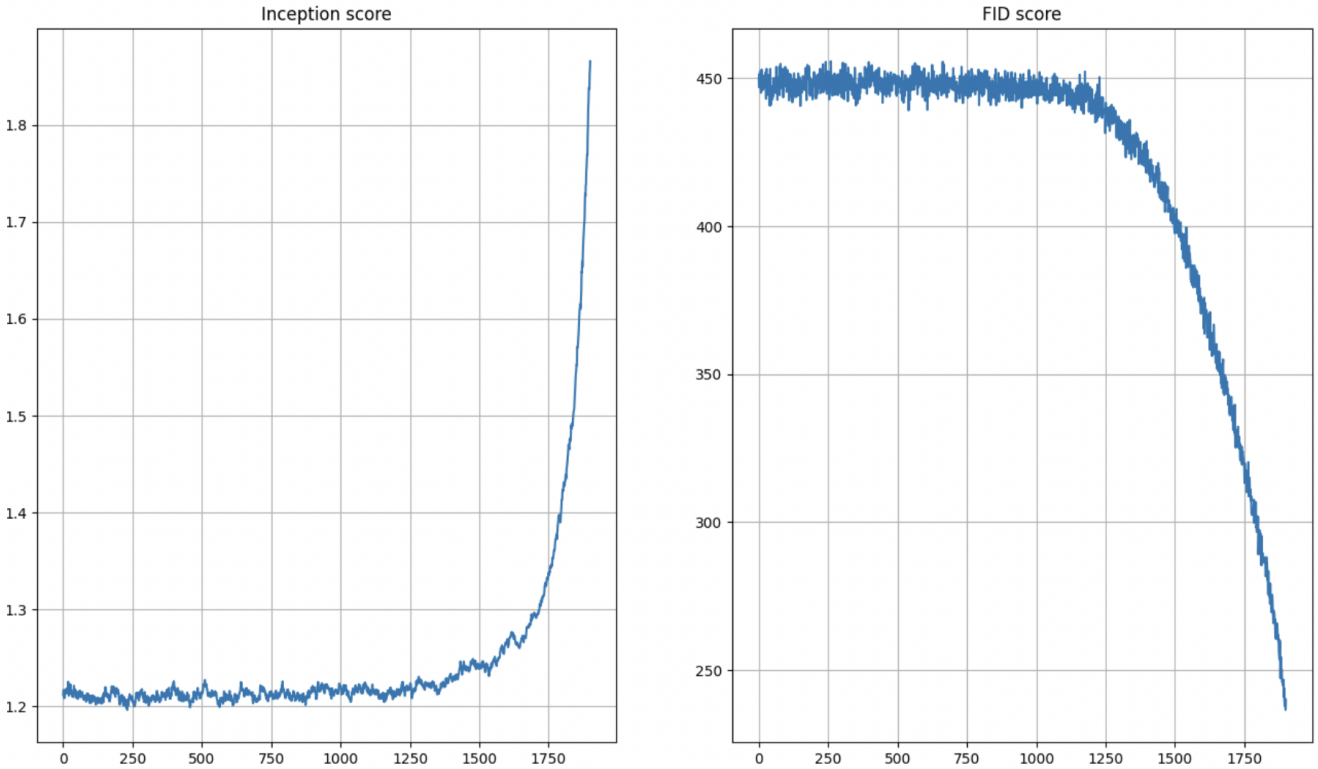
Metrics IS, FID for 500 steps:





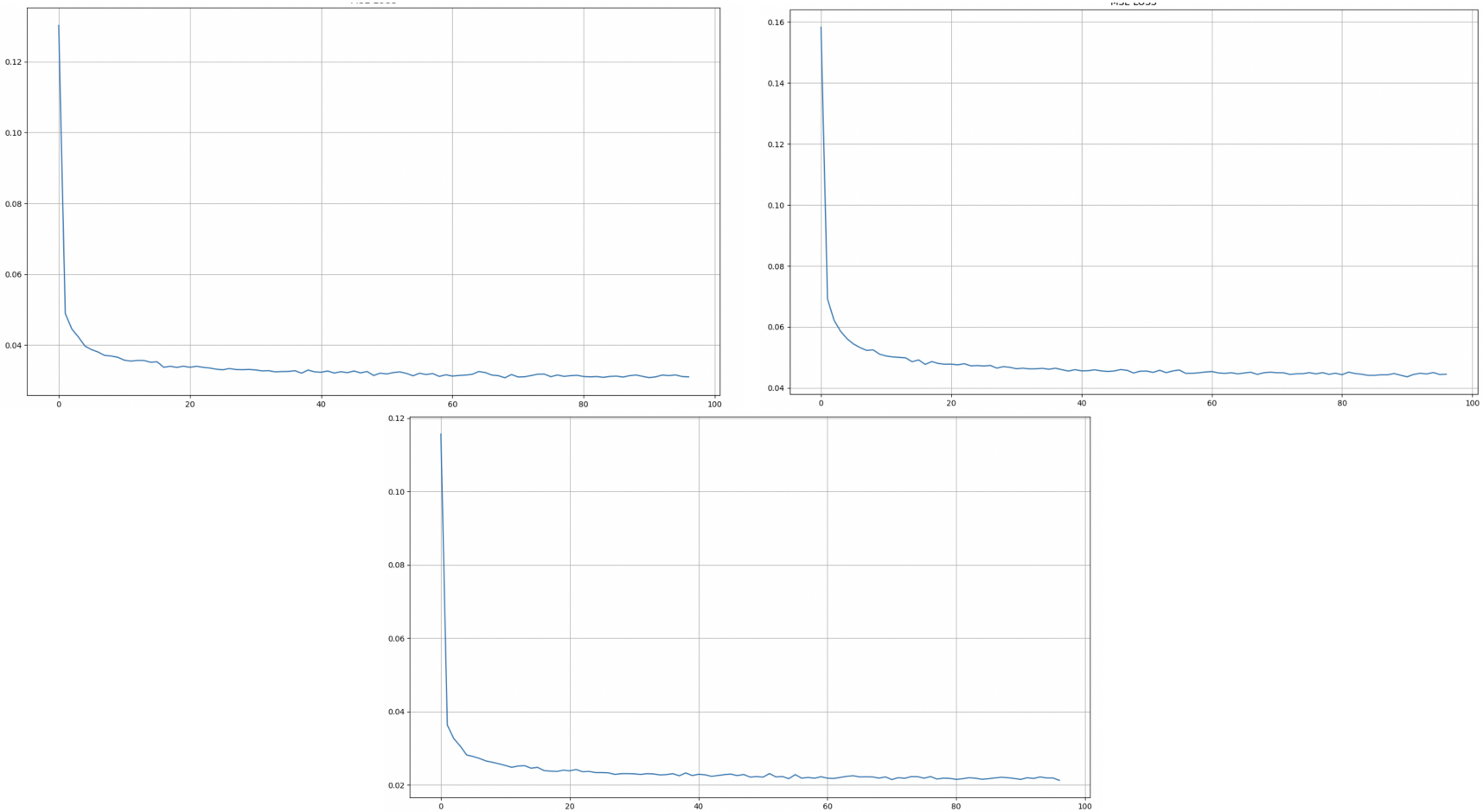
Link for visualization

Metrics IS, FID for 2000 steps:



Link for visualization

ELBO for 1000, 500, 2000 steps



Analysis of results

- Based on the results of the experiments, we have that the more steps in training, the better the metrics IS, FID

500 steps:	**IS = 1.42**	**FID = 300**
1000 steps:	**IS = 1.58**	**FID = 270**
2000 steps:	**IS = 1.88**	**FID = 230**

- With an increase in the number of steps, the generation of images begins to take a lot of time
- The graphs show that the metrics are beginning to improve noticeably in the last steps of the generation

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.

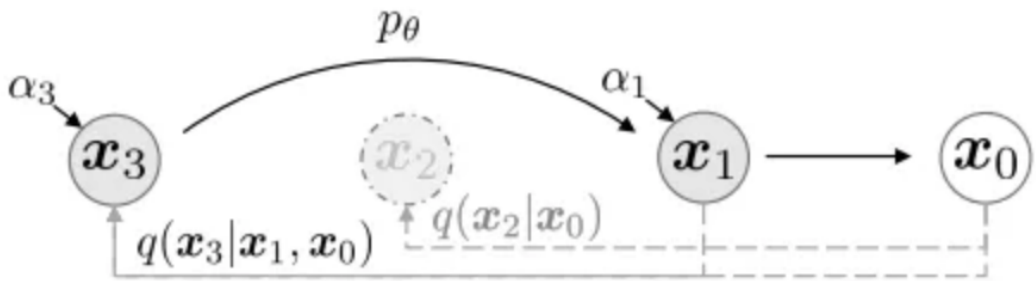
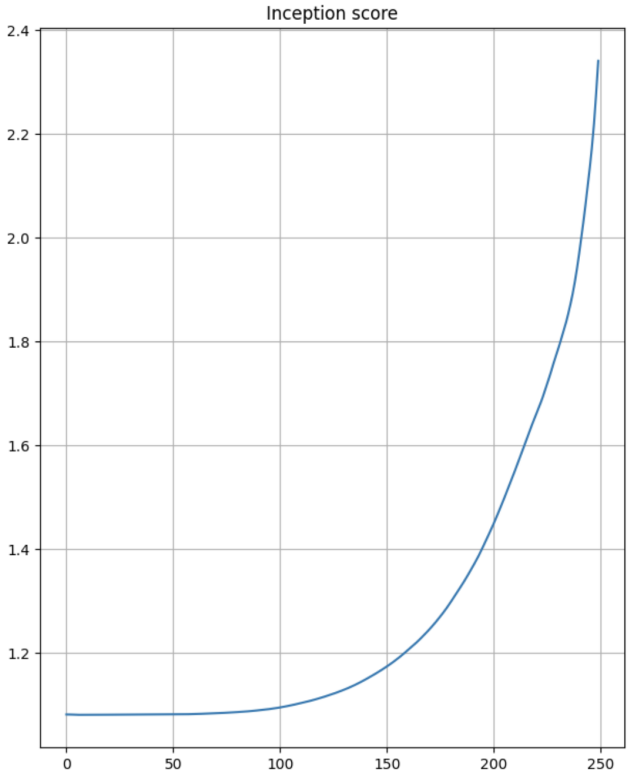
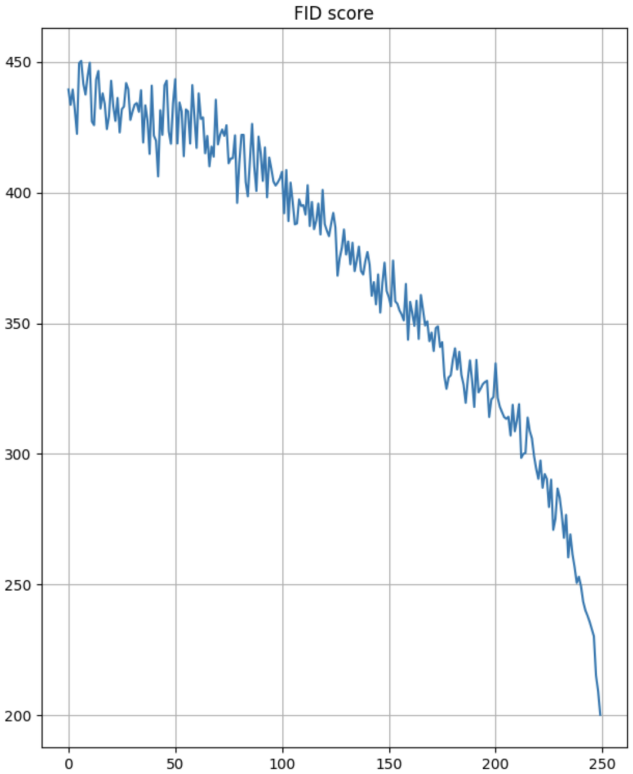


Figure 2: Graphical model for accelerated generation, where $\tau = [1, 3]$.

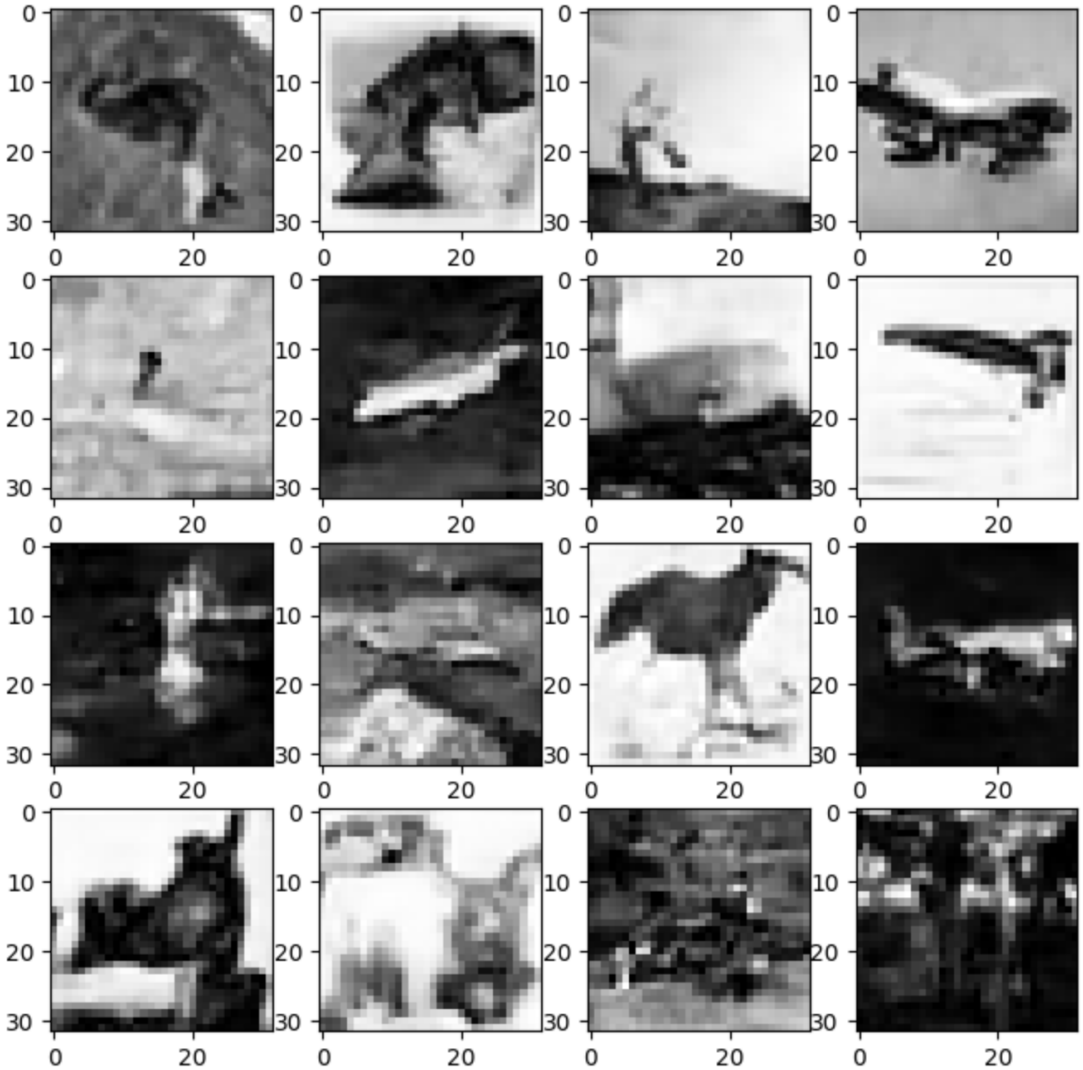
It does so by redefining the diffusion process as a non-Markovian process.

IS, FID for 1000 training steps with 250 inference steps





Visualization



Analysis of results

- Using IDDPM I got metrics better than in DDPM. The authors of the DDPM article wrote that variance learning is unstable. IDDPM solved this problem.

$$\textbf{1000 steps:} \quad \textbf{IS} = 2.38 \quad \textbf{FID} = 200$$

- The graphs of the IS, FID metrics trained at 1000 DDPM steps show that only the last 250 generation steps give a significant increase. Using this heuristic, set the number of steps for DDIM
- The advantage of DDIM is not only in accelerating generation, but also in the fact that you can easily turn a trained DDPM into DDIM by correcting only the generation method

Classifier guidance

Algorithm

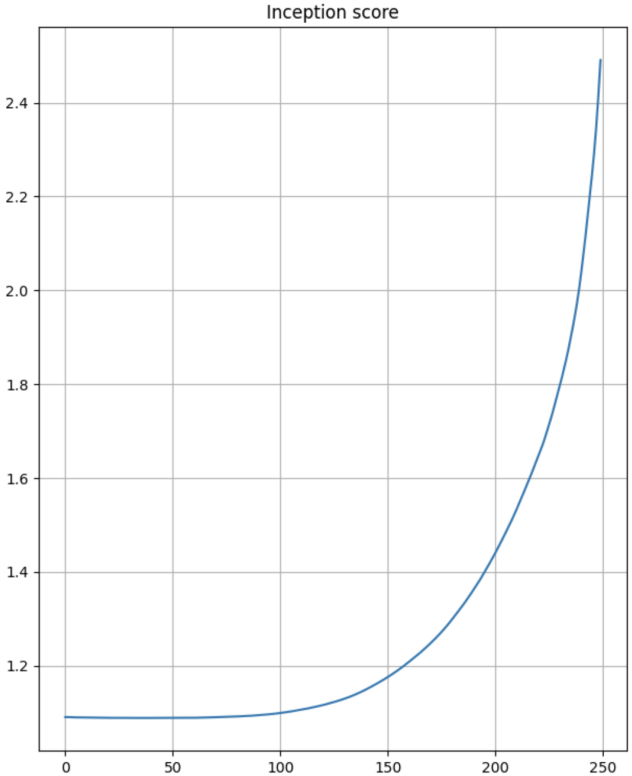
Classifier guidance was introduced in the paper "Diffusion Models Beat GANs on Image Synthesis" and essentially uses a classifier to guide the diffusion model to generate images of a desired class.

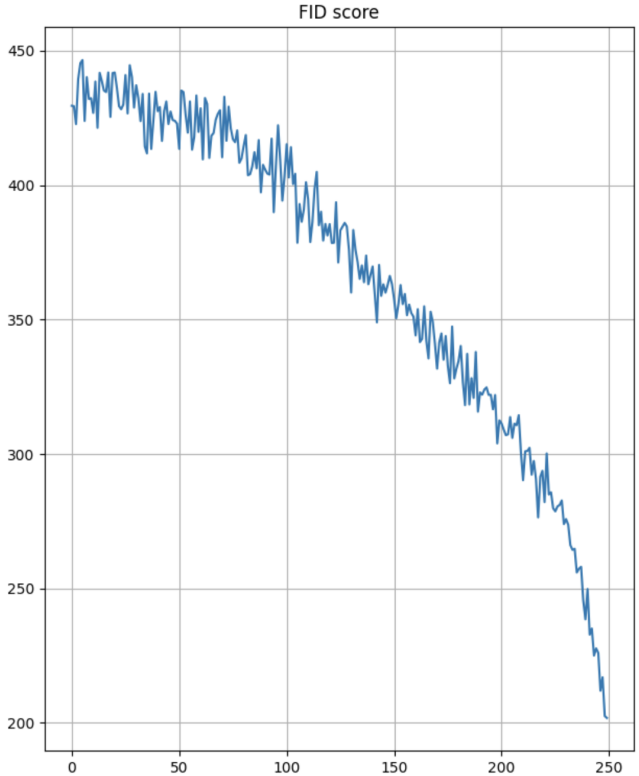
Algorithm 1 Classifier guided diffusion sampling, given a diffusion model $(\mu_\theta(x_t), \Sigma_\theta(x_t))$, classifier $p_\phi(y|x_t)$, and gradient scale s.

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

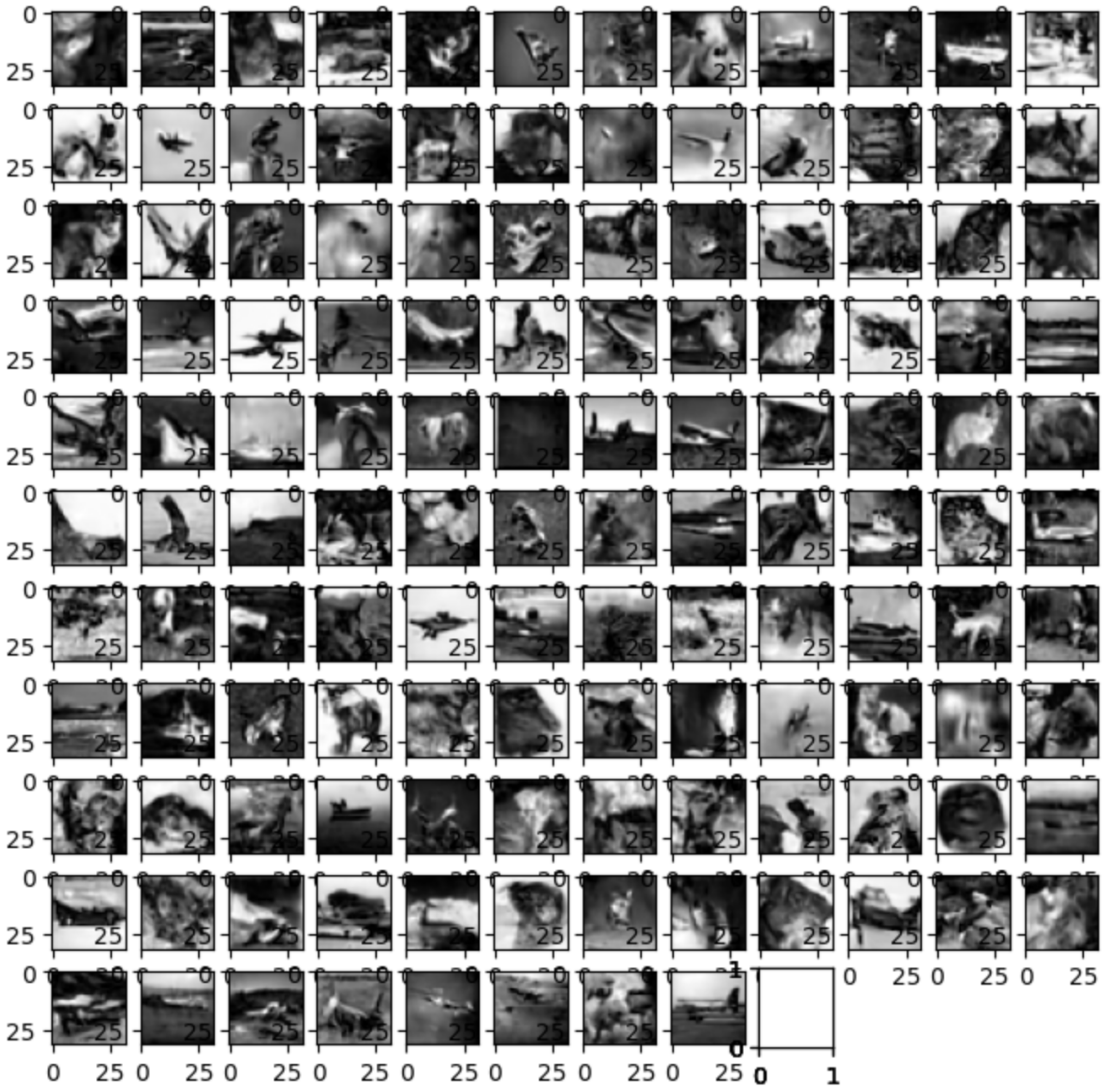
Classifier guidance

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

repeat
  $(\mathbf{x}, \mathbf{c}) \sim p(\mathbf{x}, \mathbf{c})$ ▷ Sample data with conditioning from the dataset
  $\mathbf{c} \leftarrow \varnothing$ with probability p_{uncond} ▷ Randomly discard conditioning to train unconditionally
  $\lambda \sim p(\lambda)$ ▷ Sample log SNR value
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\mathbf{z}_\lambda = \alpha_\lambda \mathbf{x} + \sigma_\lambda \boldsymbol{\epsilon}$ ▷ Corrupt data to the sampled log SNR value
  Take gradient step on $\nabla_\theta \left\| \boldsymbol{\epsilon}_\theta(\mathbf{z}_\lambda, \mathbf{c}) - \boldsymbol{\epsilon} \right\|^2$ ▷ Optimization of denoising model
until converged

Training a diffusion model for classifier-free guidance

Algorithm 2 Conditional sampling with classifier-free guidance

Require: w: guidance strength
Require: $\mathbf{c}$: conditioning information for conditional sampling
Require: $\lambda_1, \ldots, \lambda_T$: increasing log SNR sequence with $\lambda_1 = \lambda_{\min}, \lambda_T = \lambda_{\max}$

$\mathbf{z}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = 1, \ldots, T$ **do**
  ▷ Form the classifier-free guided score at log SNR λ_t
  $\tilde{\boldsymbol{\epsilon}}_t = (1 + w)\boldsymbol{\epsilon}_\theta(\mathbf{z}_t, \mathbf{c}) - w\boldsymbol{\epsilon}_\theta(\mathbf{z}_t)$
  ▷ Sampling step (could be replaced by another sampler, e.g. DDIM)
  $\tilde{\mathbf{x}}_t = (\mathbf{z}_t - \sigma_{\lambda_t} \tilde{\boldsymbol{\epsilon}}_t)/\alpha_{\lambda_t}$
  $\mathbf{z}_{t+1} \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}_{\lambda_{t+1}|\lambda_t}(\mathbf{z}_t, \tilde{\mathbf{x}}_t), (\tilde{\sigma}^2_{\lambda_{t+1}|\lambda_t})^{1-v}(\sigma^2_{\lambda_t|\lambda_{t+1}})^v)$ if $t < T$ else $\mathbf{z}_{t+1} = \tilde{\mathbf{x}}_t$
end for
return $\mathbf{z}_{T+1}$

Sampling with classifier-free guidance

I obtained IS = 3.16, FID = 184

Visualization

Link for MNIST vizualization

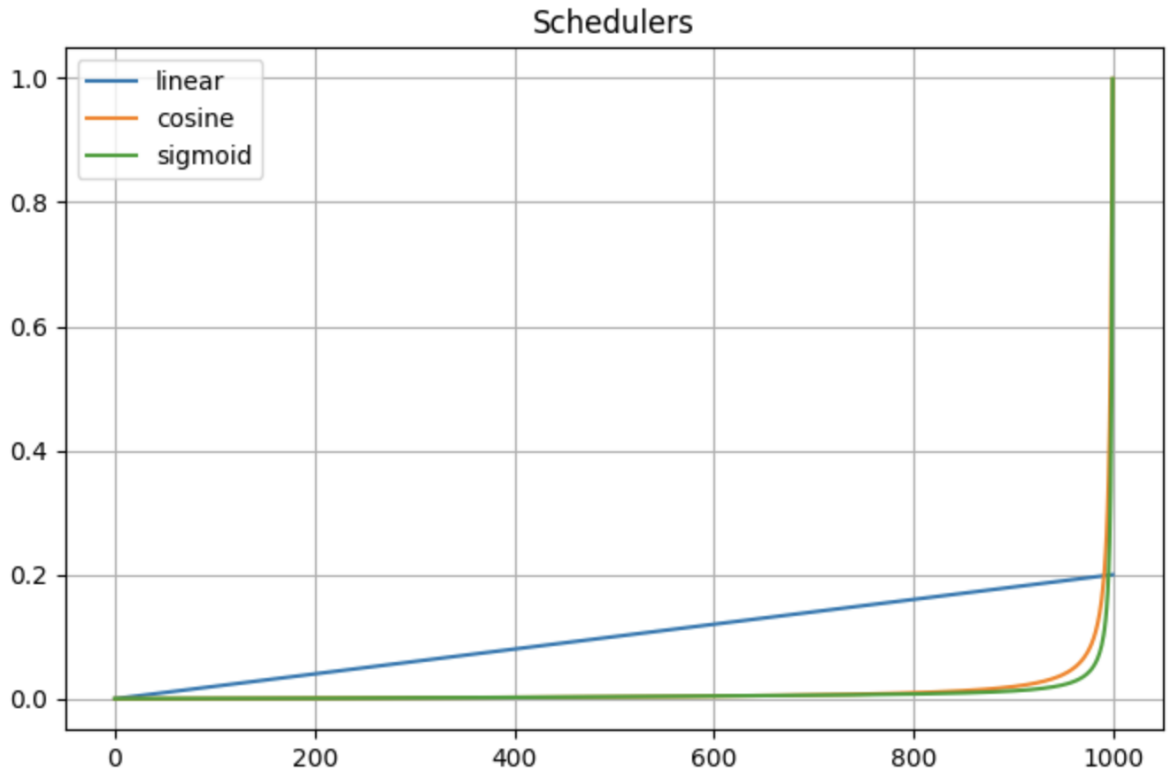
Analysis of results

- When $w = 0$, the model is a normal DDPM with class information.
- When $w > 0$, we utilize classifier-free guidance. The goal is to produce an image of class c. The idea is that the class-informed model will generate an output about the class we want to generate, but the class signal could be stronger.
- I considered w=0, w=0.5 and w=2. And get gif for MNIST. The larger the w, the more quality the digits of a given class are generated.

Betas Schedulers

I have considered the following types of schedulers:

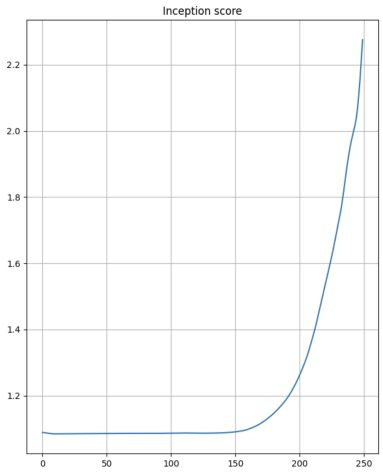
- Linear
- Cosine
- Sigmoid

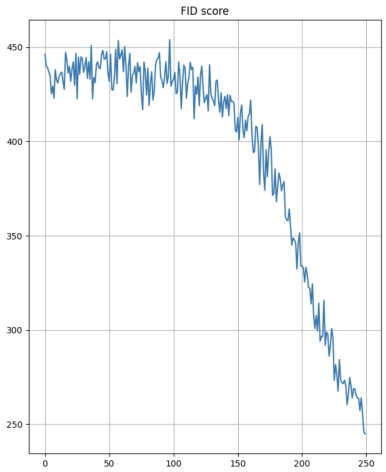


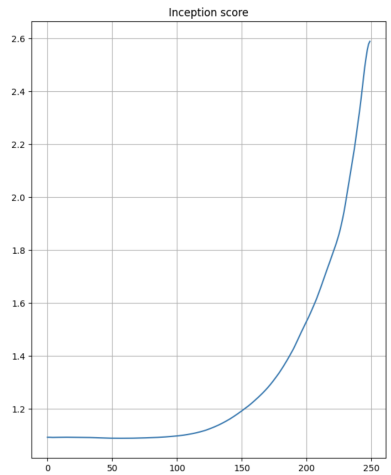
Betas Schedulers

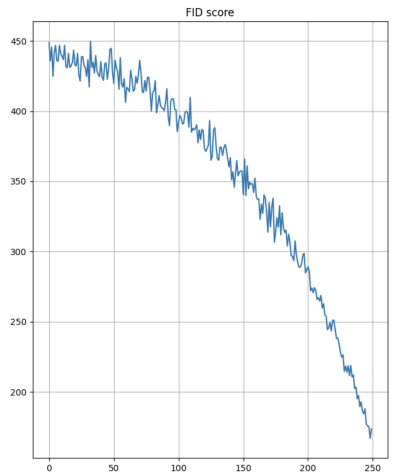
Results

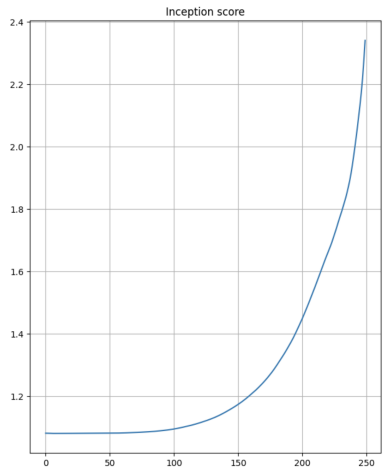
Linear, Cosine and Sigmoid scheduler:

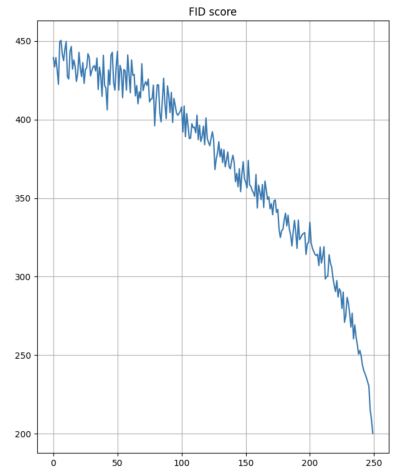






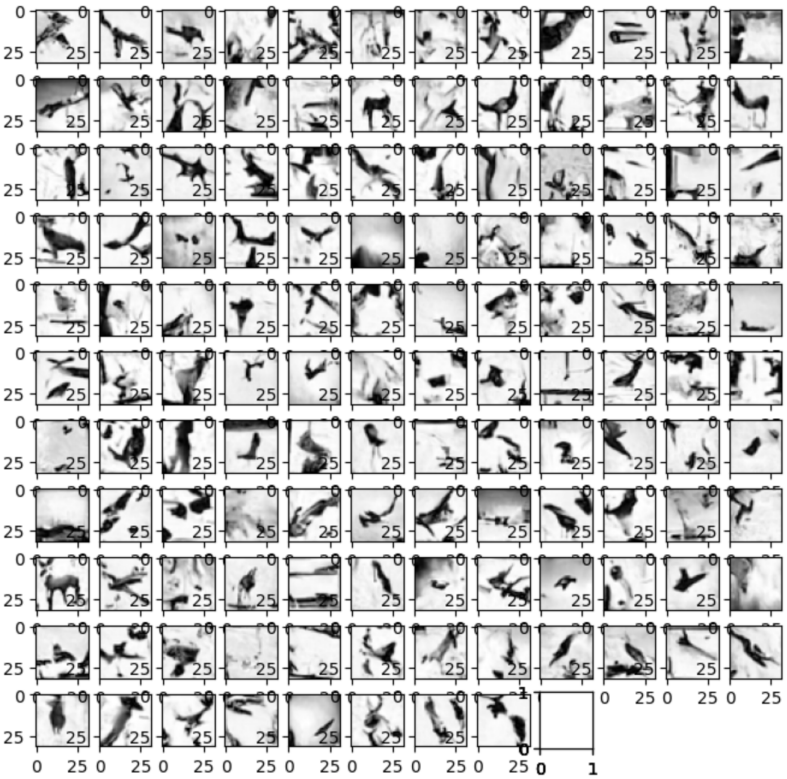


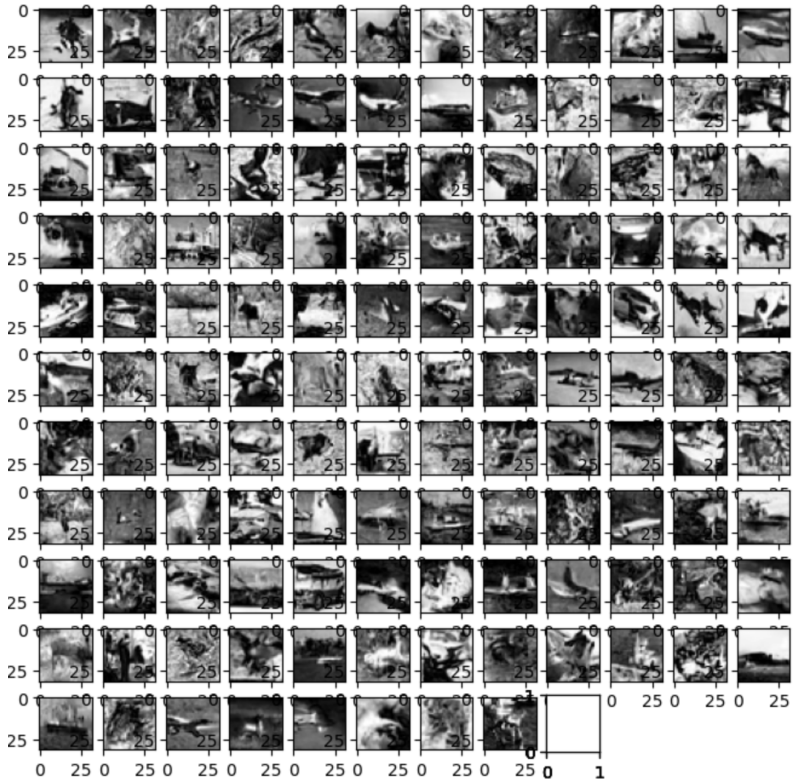




Visualization

Linear, Cosine scheduler:





Analysis of results

- As we can see from the graphs of metrics cosine scheduler wins over the rest of the schedulers.

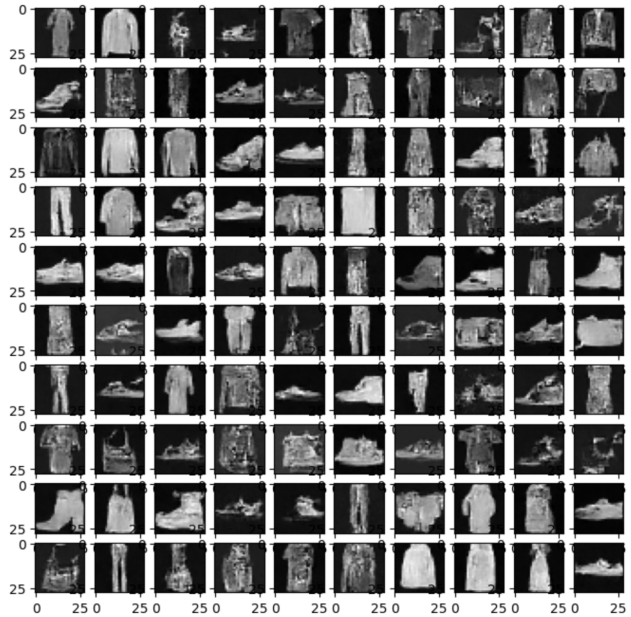
cosine scheduler:	**IS = 2.6**	**FID = 160**
sigmoid scheduler:	**IS = 2.4**	**FID = 200**
linear scheduler:	**IS = 2.3**	**FID = 250**

- The authors DDIM note that "the end of the forward noising process is too noisy, and so doesn't contribute very much to sample quality."
  The main problem with the linear scheduler is for small images. The image is not far from pure Gaussian noise too early in the diffusion process, which may make it hard for the model to learn the reverse process. Essentially, noise is being added too fast. The cosine scheduler adds noise slower to retain image information for later timesteps.

Comparison of datasets

Dataset - MNIST(FASHION MNIST)

- When learning DDPM on the FASHION MNIST dataset for 20 epochs, I received the following pictures:

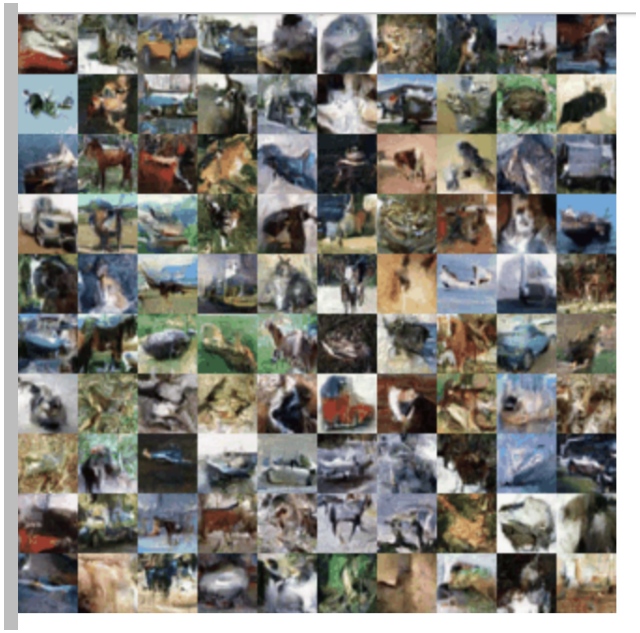


It is difficult to get IS and FID for datasets with one channel, since most classifiers are pre-trained on three channel images, for example ImageNet. But in this case we can evaluate the quality with our eyes

Comparison of datasets

Dataset - CIFAR10

- When learning DDPM on the CIFAR10 dataset for 100 epochs, I received the following pictures:

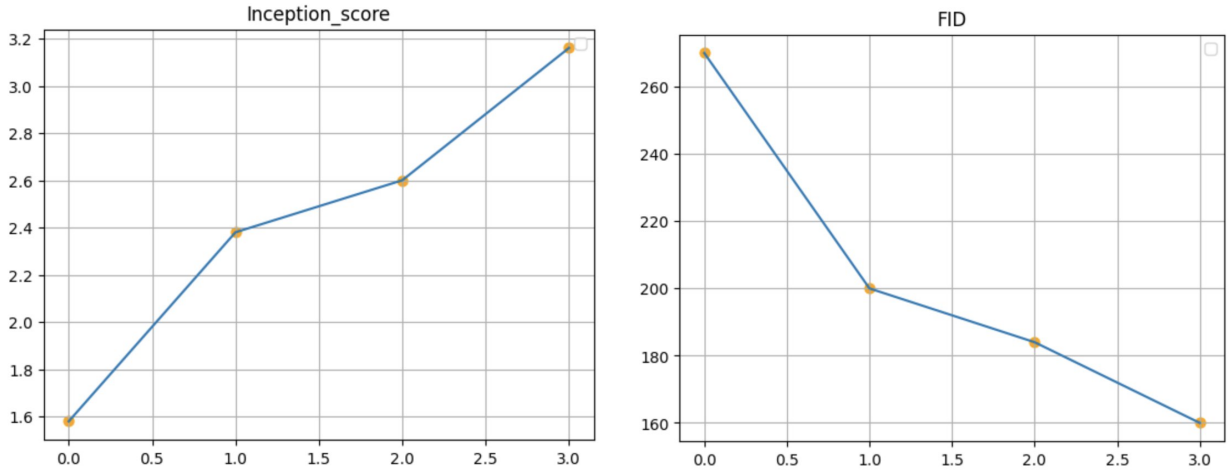


and IS $= 2.38$, FID $= 200$

- Looking at the pictures, we can see that the model is trained on single-channel pictures quite quickly and efficiently.

Resume

Evolution of score:



0 - default DDPM with linear scheduler (1000 steps)

1 - IDDPM(1000 steps) + DDIM(inference - 250 steps) with sigmoid scheduler

2 - IDDPM(1000 steps) + DDIM(inference - 250 steps) with cosine scheduler

3 - Classifier-free guidance + IDDPM(1000 steps) + DDIM(inference - 250 steps) with cosine scheduler - **best configuration**

Sources

Links:

- Variational Diffusion Model
- 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
- Github with code of project