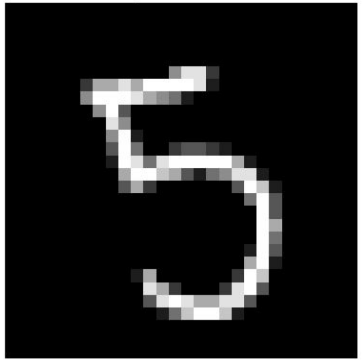
Meeting 3
24.12.2024

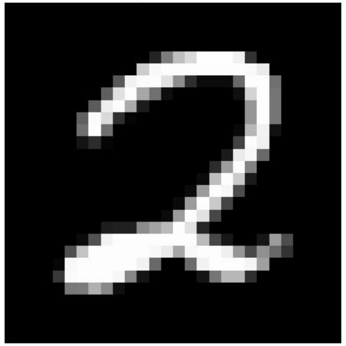
- VAE sample generations
- DDPM
  - Theory
  - Architecture
  - Samples form the paper and local training
  - Latent space information
  - What to do next: Conditional Generation

Here the samples passed into encoder. Encoder calculates their variances and the means. Only the mean vector (e=0) passed into the decoder.

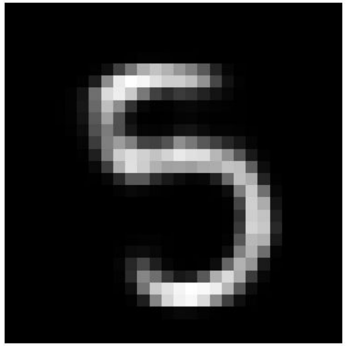
Example '5' from Train



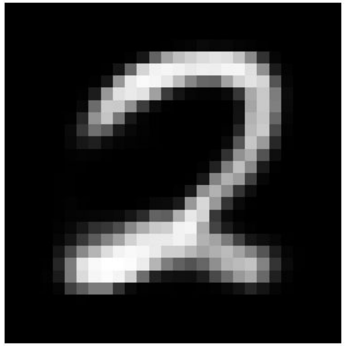
Example '2' from Test

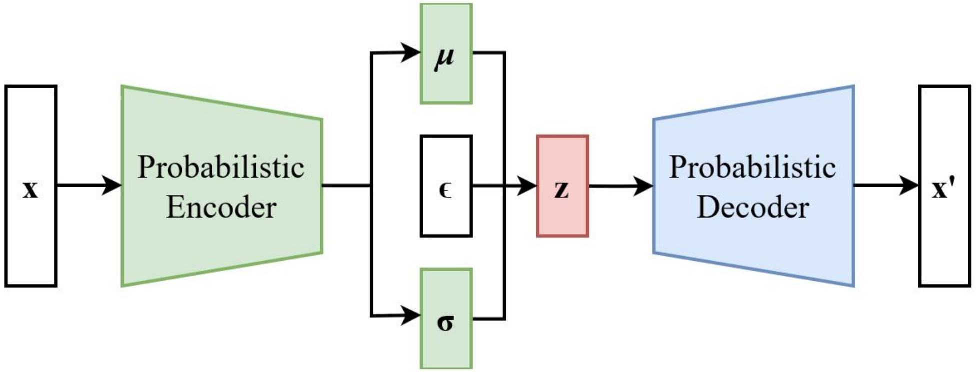


The image reconstructed from the mean of the distribution '5' from Train



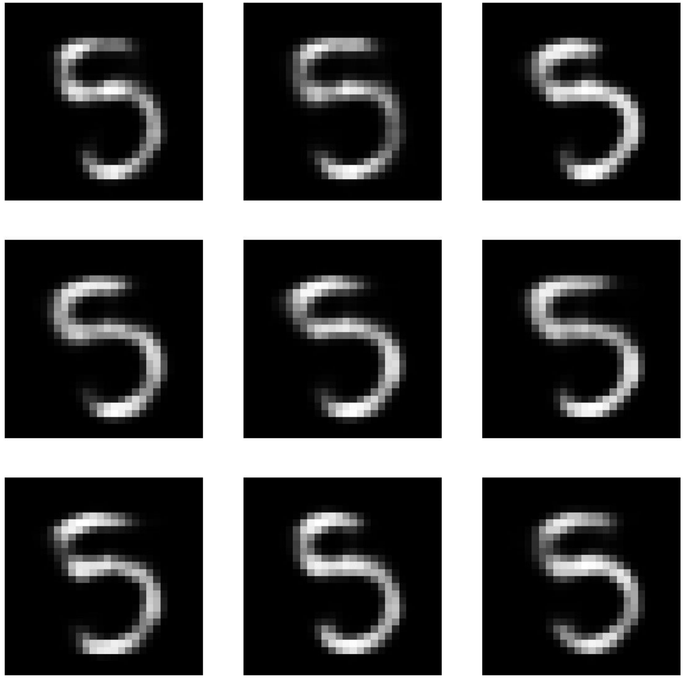
The image reconstructed from the mean of the distribution '2' from Test

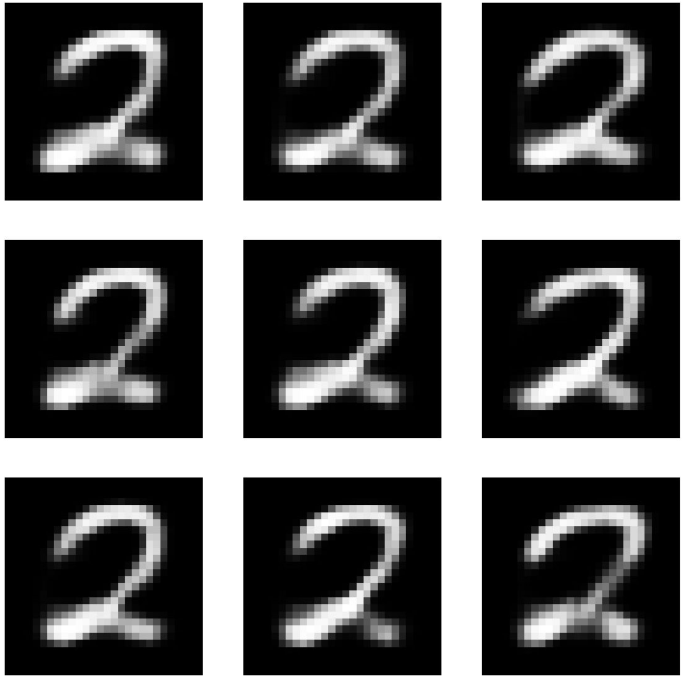




$$z = \mu + \sigma\epsilon, \quad \epsilon \sim \mathcal{N}(0,1)$$

Generated samples using **the mean and the covariance of a specific** sample from the ambient space.


Samples from the “train 5”


Samples from the “test 2”

VAE Hallucinations

For each gif below, an error vector is fixed.
The the latent $\boldsymbol{z=\mu+a*[\varepsilon\odot\sigma]}$goes into the decoder.
a ranges from [0,10] with increment 0.1

DDPM (Denosising Diffusion Probabilistic Models, Ho et al 2020)

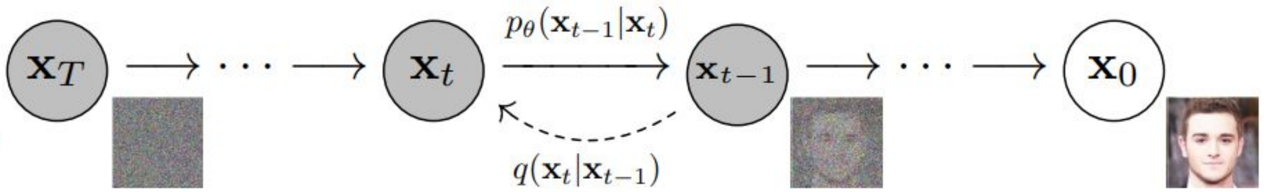

$\mathbf{x}_T$
$\mathbf{x}_t$
$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$\mathbf{x}_{t-1}$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$
$\mathbf{x}_0$

Figure 2: The directed graphical model considered in this work.

Forward/Diffusion process: **(Fixed)**

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

Reverse process: **(Learned)**

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

ELBO and Training

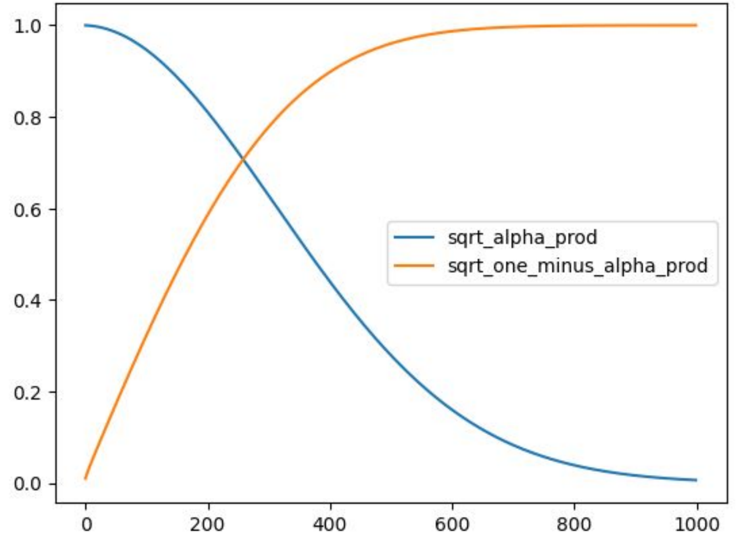
$$\log p(\boldsymbol{x}) \geq \mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\left[\log \frac{p(\boldsymbol{x}_{0:T})}{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}\right]$$

$$= \underbrace{\mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)\right]}_{\text{reconstruction term}} - \underbrace{D_{\mathrm{KL}}(q(\boldsymbol{x}_T|\boldsymbol{x}_0) \parallel p(\boldsymbol{x}_T))}_{\text{prior matching term}} - \sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{\text{denoising matching term}}$$

Thanks to property sum of two gaussian variables equal to another gaussian variable:

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

Plugging this into the ELBO, a closed form loss function achieved.



$\beta_1 = 10^{-4}$ to $\beta_T = 0.02.$

$$\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)) \quad = \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)} \frac{(1-\alpha_t)^2}{(1-\bar{\alpha}_t)\alpha_t} \left[\|\boldsymbol{\epsilon}_0 - \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\|_2^2 \right]$$

$$L_{\mathrm{simple}}(\theta) := \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}} \left[\left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2 \right]$$

Assumptions:

- Nosing parameters **α/β are fixed**. Later work set them as learnable and achieve better results.Nichol et al., (2021)
- **Variance is fixed** the variance to the β^2 (or a constant multiple of β). Above work also addresses this.
- A **simplified loss** is used and the **noise** is predicted. (Mean or the x0 can also be predicted, worse sampling reported.

Algorithm 1 Training

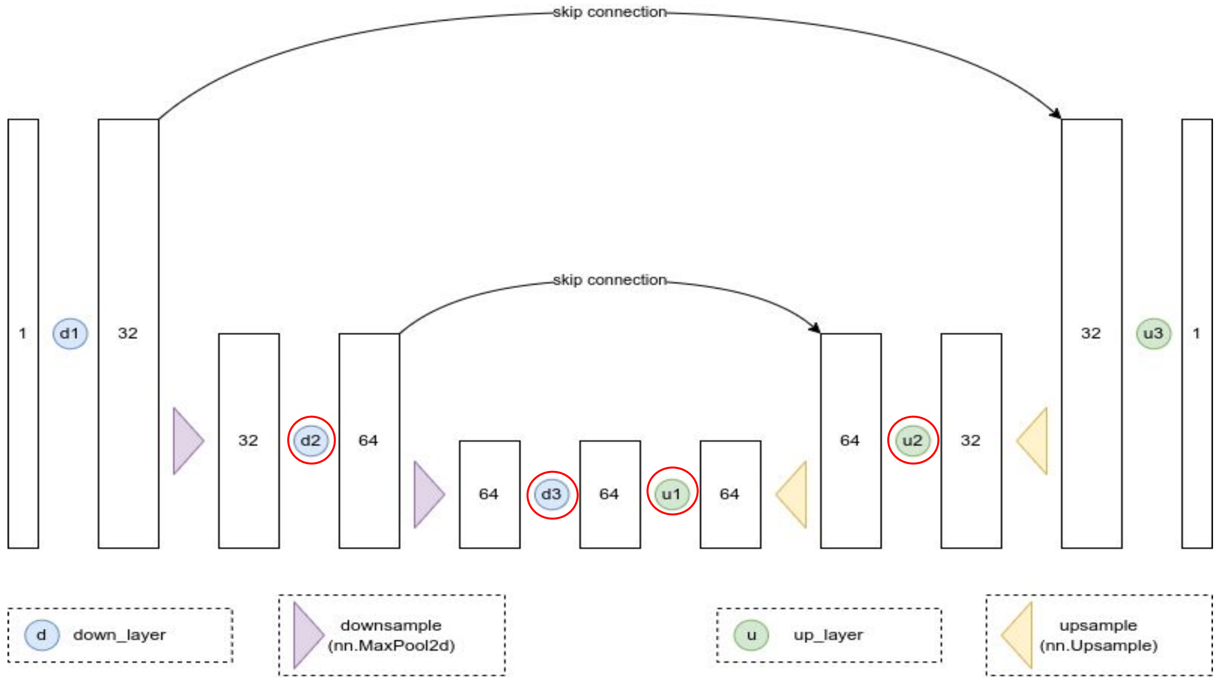
```
1: repeat
2:   $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3:   $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
4:   $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5:   Take gradient descent step on
         $\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0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

Algorithm 2 Sampling

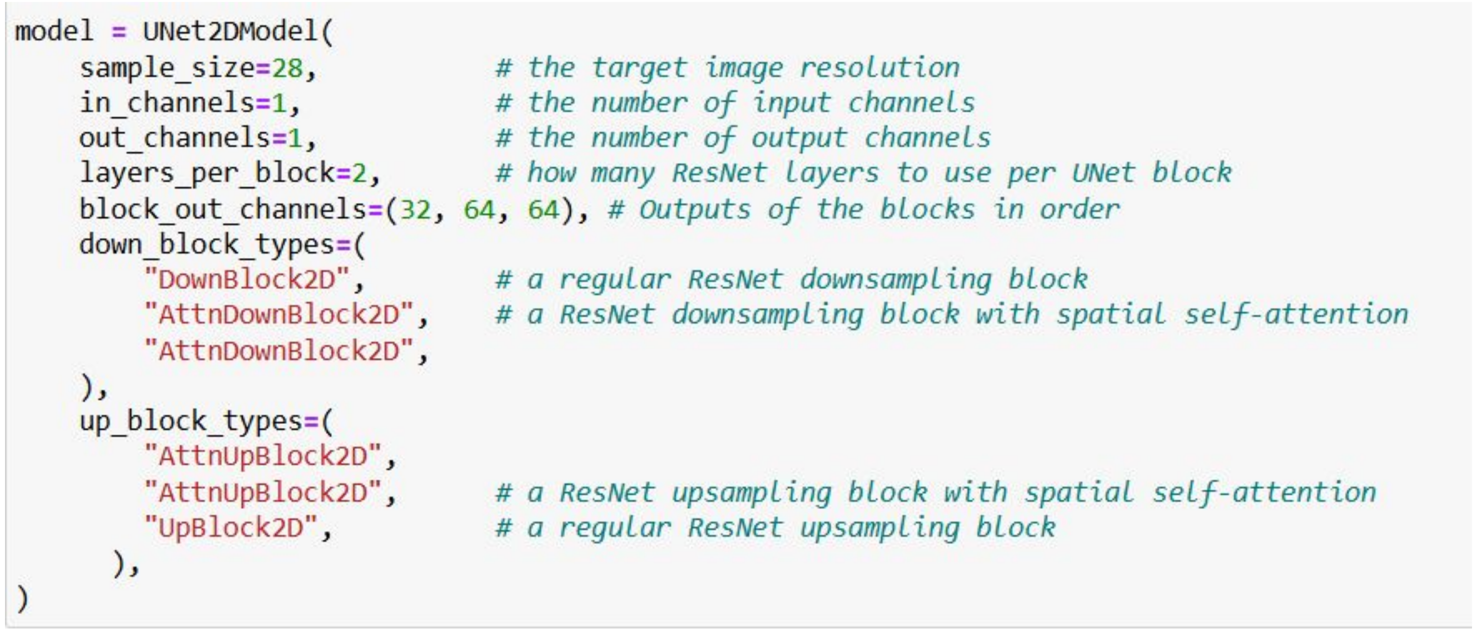
```
1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: for $t = T, \ldots, 1$ do
3:   $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4:   $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}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

Backbone: UNet

The architecture is chosen to be modified version of **UNet**. Fully Convolutional NN. (**Image to Image**). Originally used for medical image segmentation.
Other backbones are possible: **DiT** (Scalable Diffusion Moldes with Transformers) - ADD-IT uses this.
Modification to UNet: **Self-attention** down/upsampling blocks placed at the bottom of the model.
Time step embeddings in the ResNet blocks.

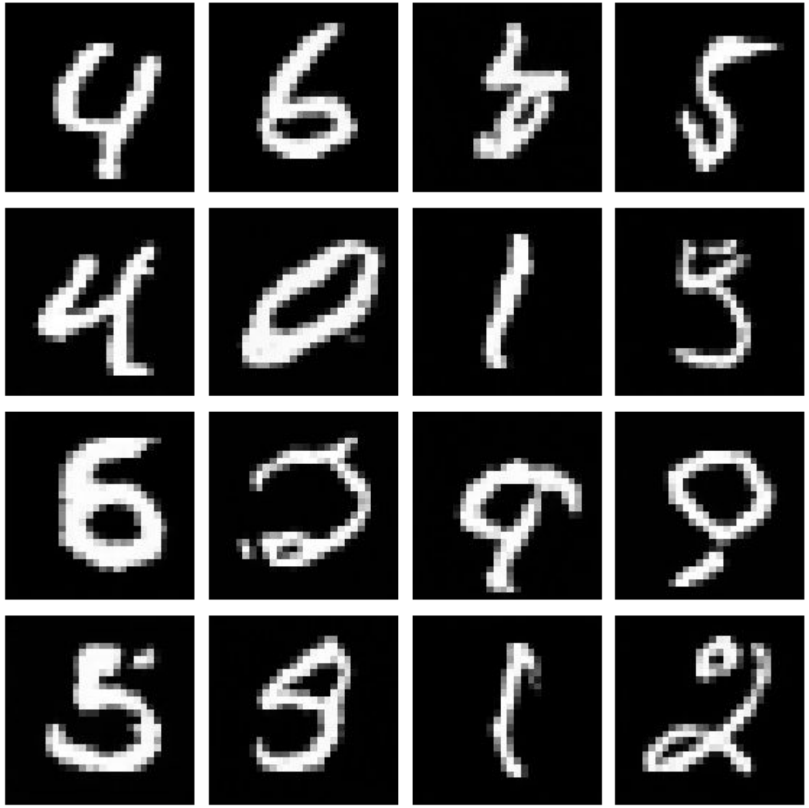


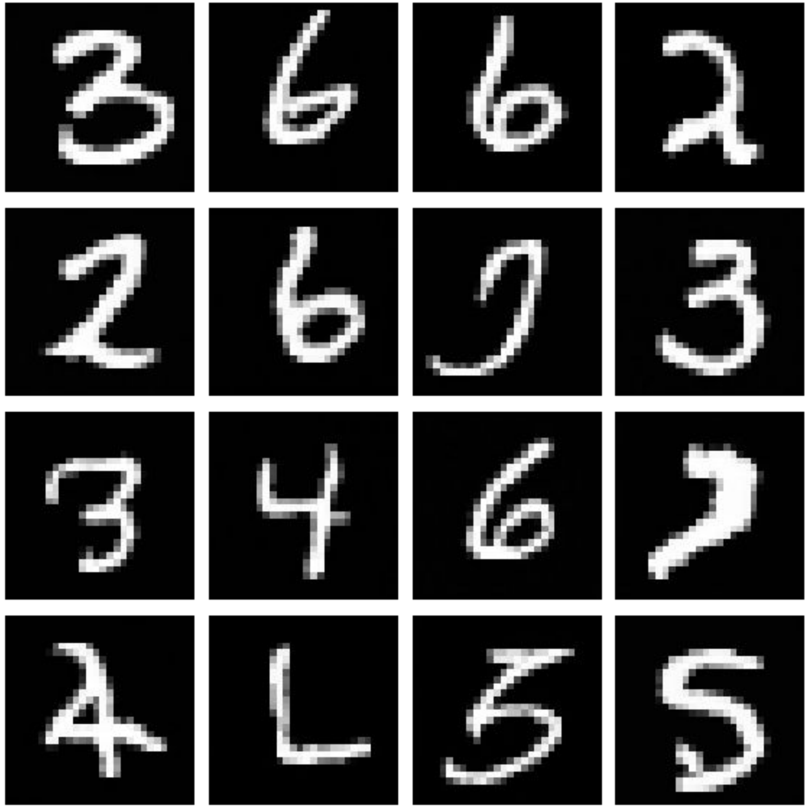
There is a library called **Diffusers**. Work well with PyTorch and make it easy to design UNet models and Time Schedulers.
Here is the model I used for my training on MNIST:



```
model = UNet2DModel(
    sample_size=28,             # the target image resolution
    in_channels=1,              # the number of input channels
    out_channels=1,             # the number of output channels
    layers_per_block=2,         # how many ResNet layers to use per UNet block
    block_out_channels=(32, 64, 64), # Outputs of the blocks in order
    down_block_types=(
        "DownBlock2D",          # a regular ResNet downsampling block
        "AttnDownBlock2D",      # a ResNet downsampling block with spatial self-attention
        "AttnDownBlock2D",
    ),
    up_block_types=(
        "AttnUpBlock2D",
        "AttnUpBlock2D",        # a ResNet upsampling block with spatial self-attention
        "UpBlock2D",            # a regular ResNet upsampling block
      ),
)
```

Generated examples from the model I trained. Left examples from the my local train and the right examples are form the colab train. The images on the correspondent grids generated from the same noise.

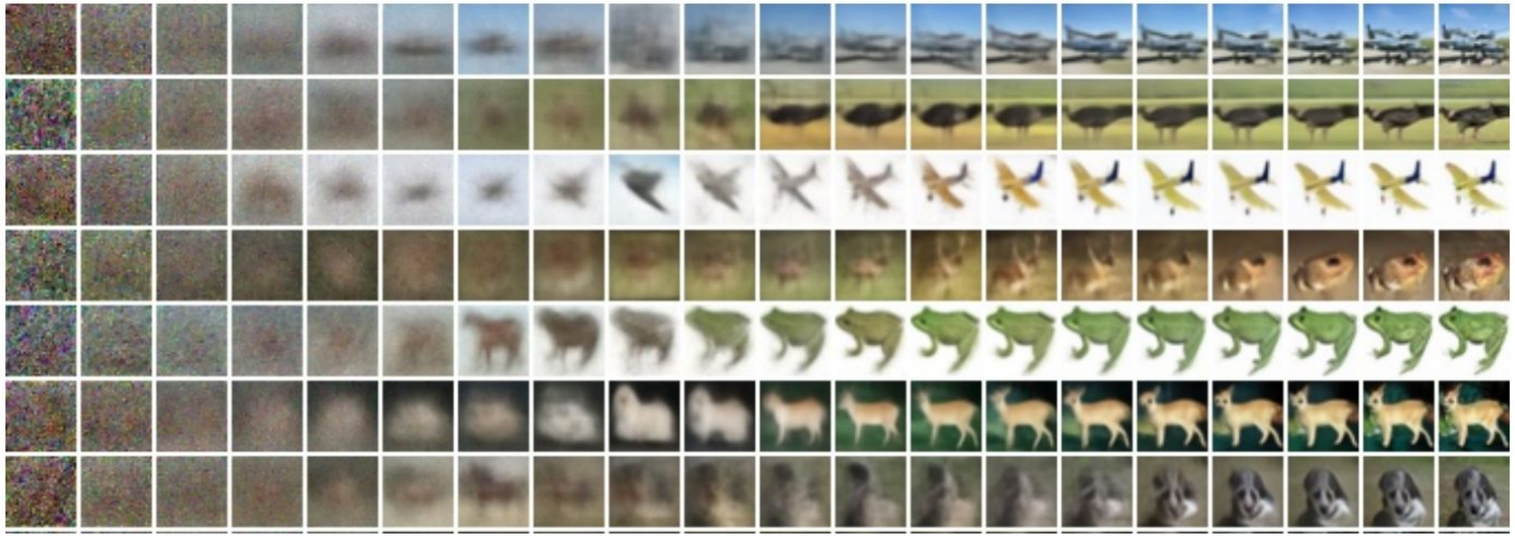

Local training (1.7M param)

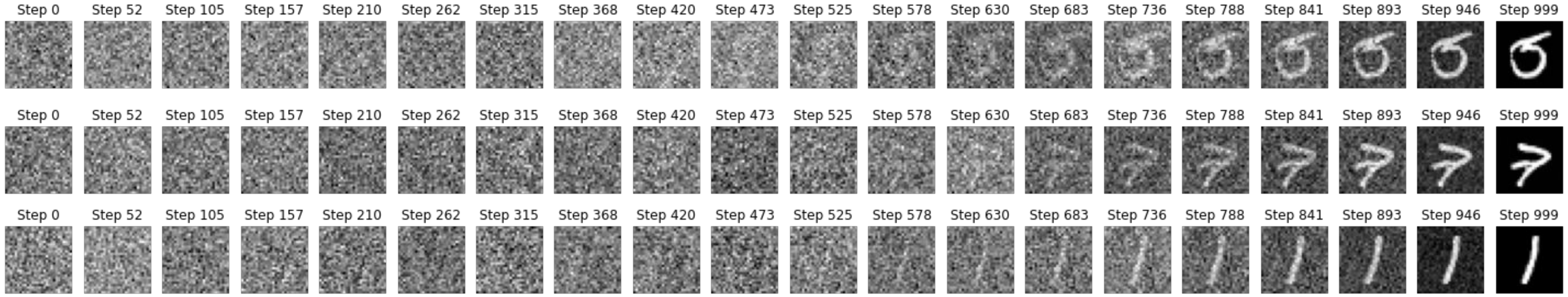

Training on Google Colab (6.5M param)

Progressive Generation

First form the paper, CIFAR10

Second mine, MNIST



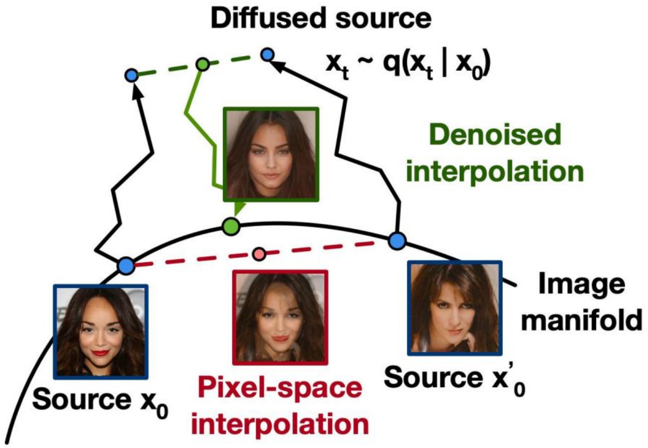


Latent Space Information

Looking and the prior (initial noise) there is no (or clearly understandable) information about the posterior (generated image)
However, intermediate steps do possess information about the posterior. They might be sign of latent space learning.

Latent space interpolation:

$$\bar{\mathbf{x}}_t = (1-\lambda)\mathbf{x}_0 + \lambda\mathbf{x}'_0$$



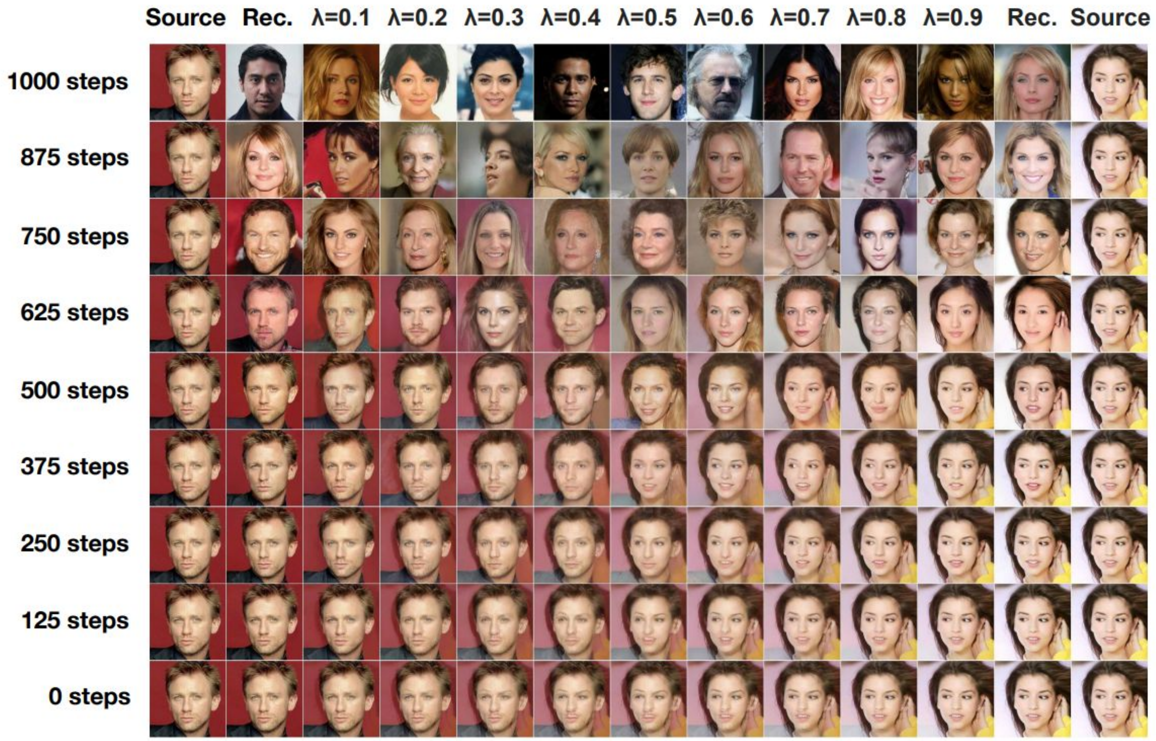
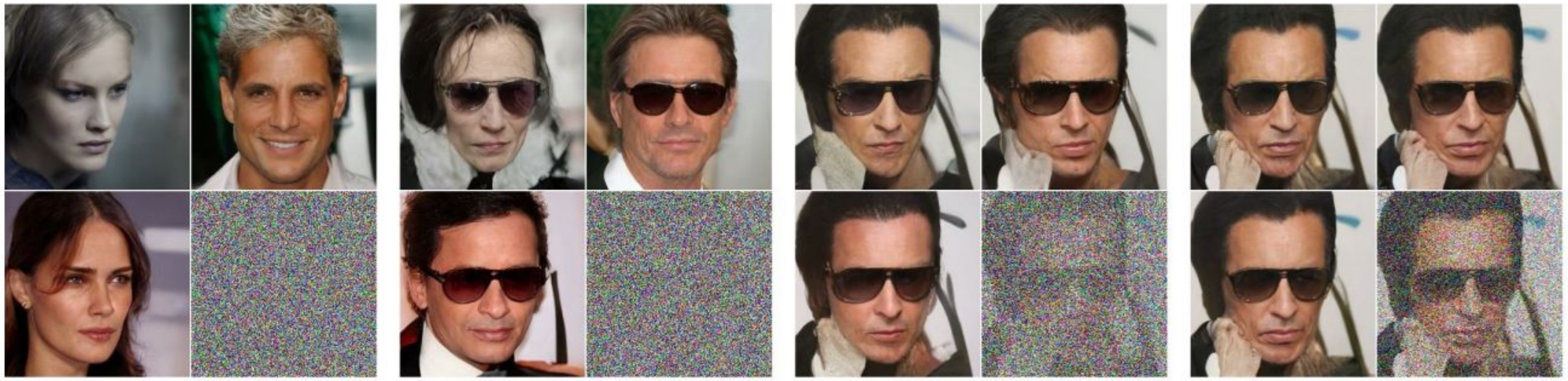


Figure 9: Coarse-to-fine interpolations that vary the number of diffusion steps prior to latent mixing.

Latent Space Information

Another experiment done by the authors of the DDPM paper is freezing a latent in the reverse process and creating branches from that frozen point.

1. Sharing x1000 = Starting from the same noise results in different pictures even in the same model. Since during the sampling the variance taken from a gaussian distribution.
2. Sharing lower (or closer) latents result in some spatial feature resembling.



Share $\mathbf{x}_{1000}$ **Share $\mathbf{x}_{750}$** **Share $\mathbf{x}_{500}$** **Share $\mathbf{x}_{250}$**

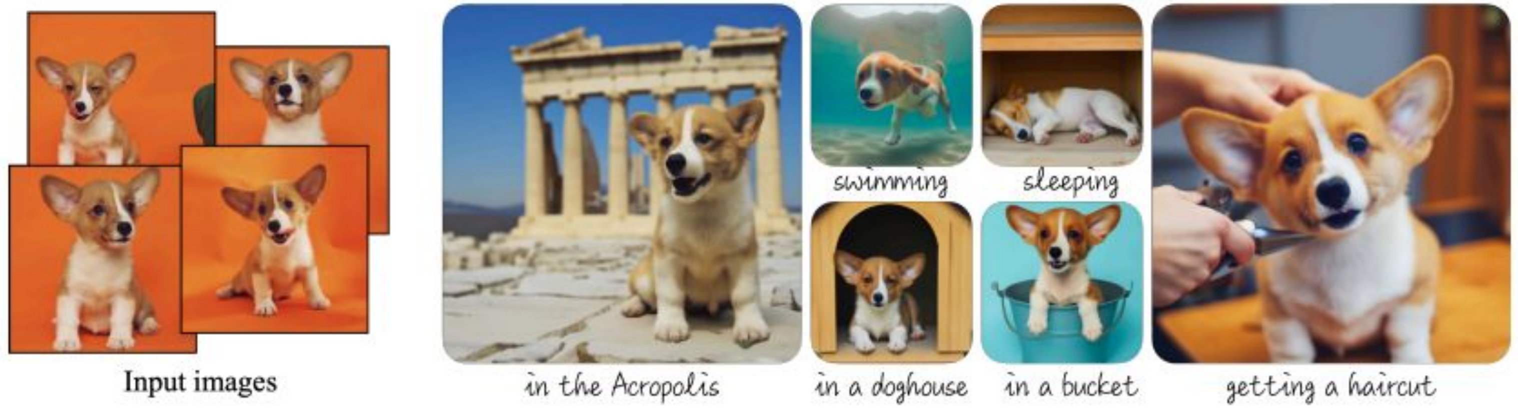
What to do next: Conditional Generation

When learned the reverse process distribution **p(x)** is conditioned on y, the model can be trained with **p(x|y)** to control image generation.

Classifier Guidance: Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis.
Classifier-Free Guidance: Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance

They are based on: Score-based generative modeling through stochastic differential equations

DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation


Input images
in the Acropolis
swimming
sleeping
in a doghouse
in a bucket
getting a haircut