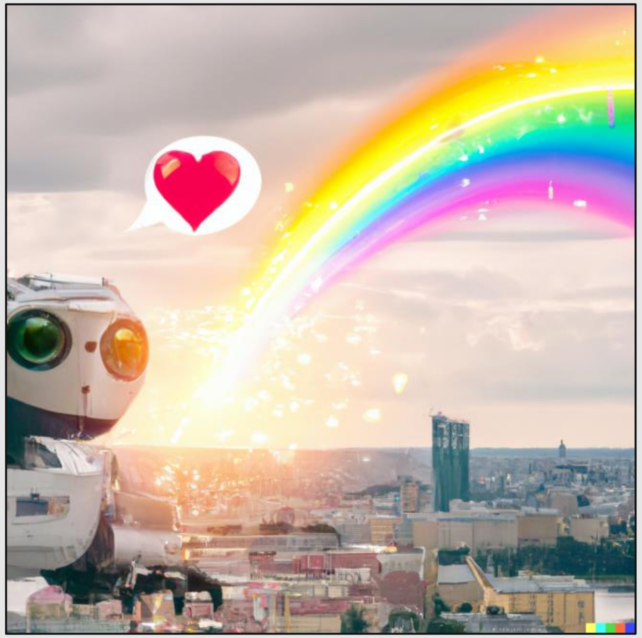
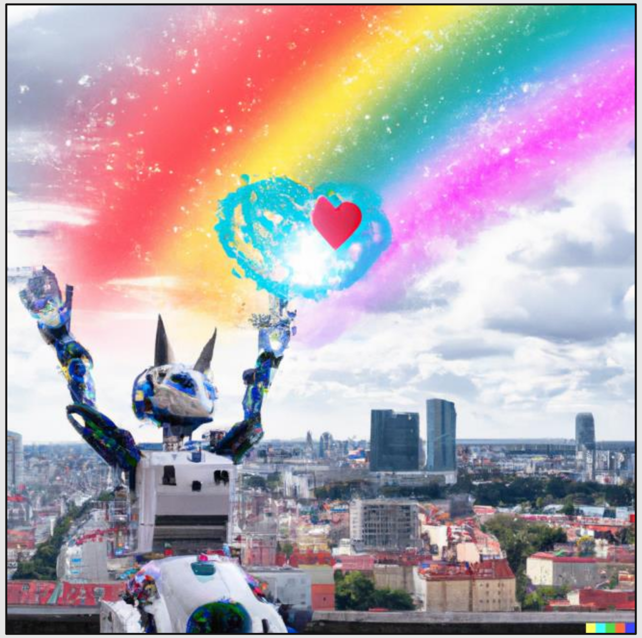
Warsaw University of Technology

VU UNIVERSITY AMSTERDAM

Introduction and Analysis of Generative and Denoising Capabilities of Diffusion-based Deep Generative Models

Kamil Deja, Anna Kuzina, Tomasz Trzciński, Jakub M. Tomczak

Teaser (DALL-E)

AI robot sends hearts and rainbows while flying over Warsaw realistic photo


Renaissance style painting of Machine Learning researchers gathering in Poland with Polish flag and emblems

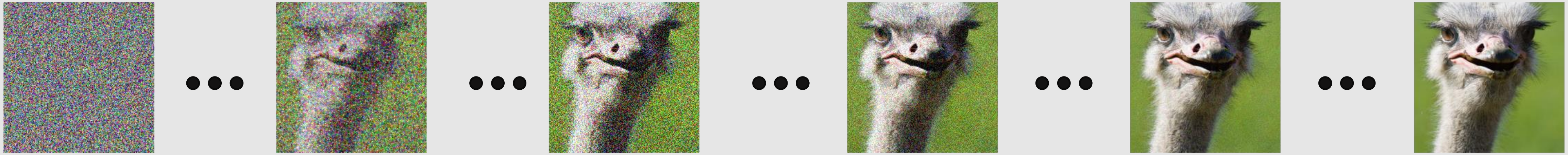
Machine Learning researchers gathering in Poland at the University

Diffusion-based generative models

Backward generative process

$p_\theta(x_{t-1}|x_t)$

x_T $\quad$ x_t $\quad$ x_{t-1} $\quad$ x_0



$N(0, \boldsymbol{I})$

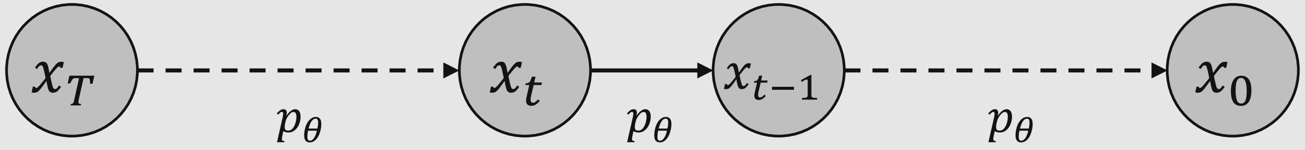
$q(x_t|x_{t-1})$

Forward diffusion process

Diffusion models training - intuitively

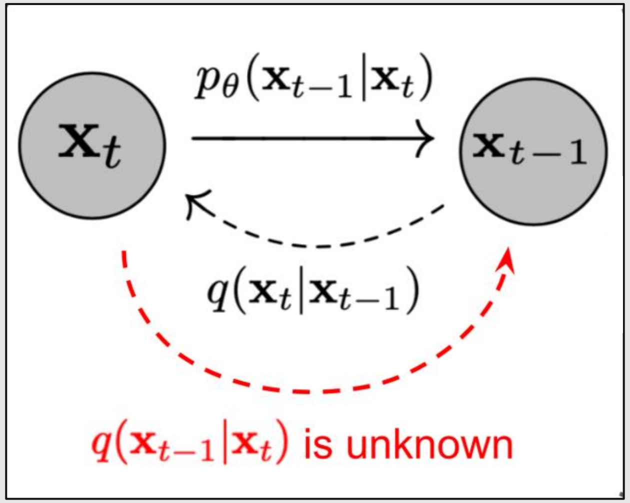
For diffusion with T steps:

- We apply the same decoder model p_θ T times to generate image from random noise
- We calculate loss on each step separately
- We train the model with the sum of individual losses



Diffusion model as Variational Autoencoder

- Forward diffusion process – Fixed encoder
- Backward generative pass – Generative decoder that predicts mean and variance for the previous diffusion step
- Loss = $D_{KL}[q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t)]$

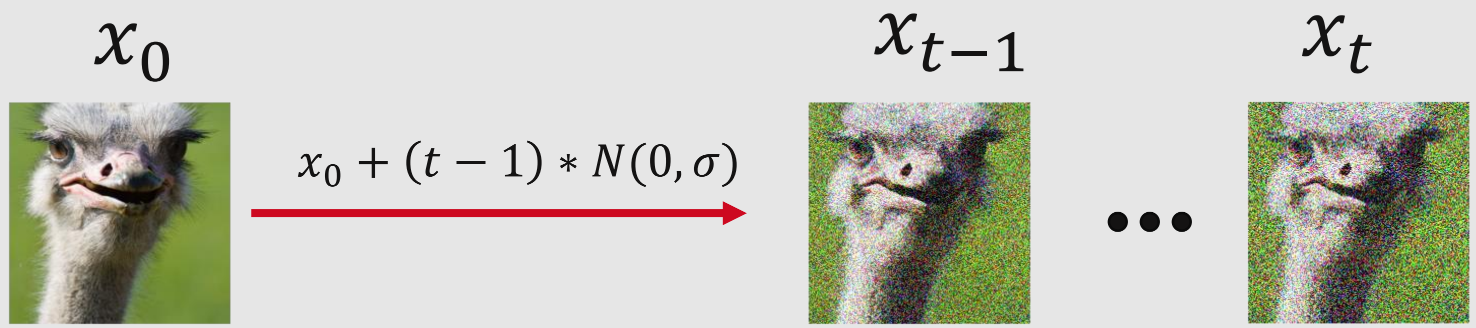


Diffusion models training

For T diffusion steps we can calculate ELBO as :

$$L = L_0 + L_1 + \cdots + L_{T-1} + L_T$$

But with a great number of diffusion steps it would be extremally slow



- We encode image with a known amount of Gaussian noise, so in fact we can always sample *any* timestep by adding all of the cumulated noise in just one step
- We can approximate sum of losses by randomly sampling different diffusion step used for training (Monte Carlo)

Model training - example

Data batch

x_0

x_0

x_0

Data samples encoded to random steps t:
$q(x_t|x_0)$
$t \sim U(0,T)$

x_{10}

x_{73}

x_{510}

Further encoding to random steps t+1:
$q(x_{t+1}|x_t)$

x_{11}

x_{74}

x_{511}

Decoding to steps t with decoder
$p_\theta(x_t|x_{t+1})$

$\hat{x}_{10}$

$\hat{x}_{73}$

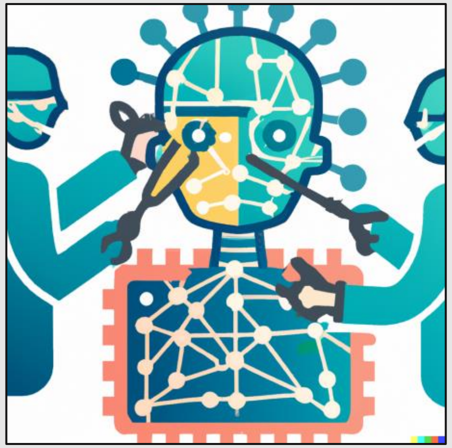
$\hat{x}_{510}$

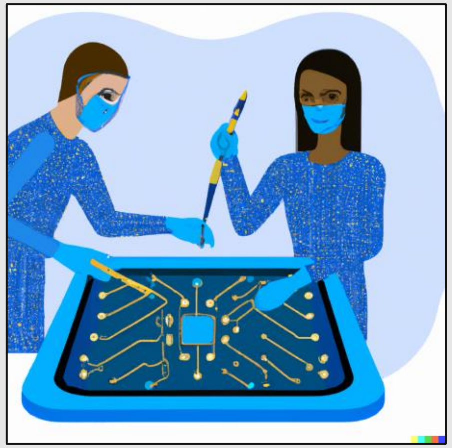
$$D_{KL}[q(x_t|x_0)||p_\theta(x_t|x_{t+1})]$$

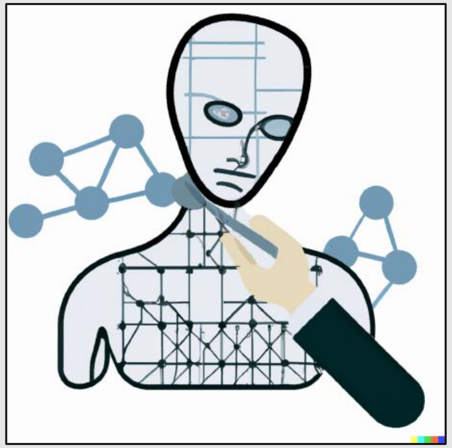
Is this all?

- DALLE-2 – conditioning diffusion models on CLIP text embeddings
- Stable diffusion – encoding original data samples to latent representations of the deterministic autoencoder before diffusion
- Classifier and classifier-free guidance – for conditional generations
- GradTTS – alternation to the prior distribution for speech synthesis
- And many more…

Analysis of DDGMs

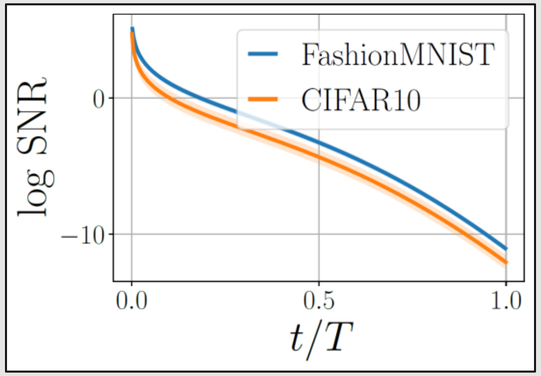




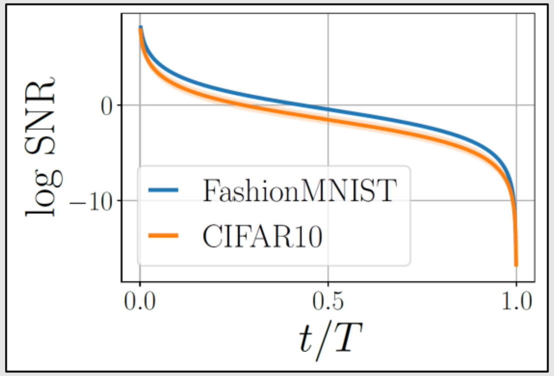


Analysis of the diffusion process

- The biggest changes in the log-SNR are noticeable within the first 10% of steps
- Data signal is the strongest within the first 10-20% of diffusion steps
- Reconstruction error grows beyond significant values after ~10% of steps

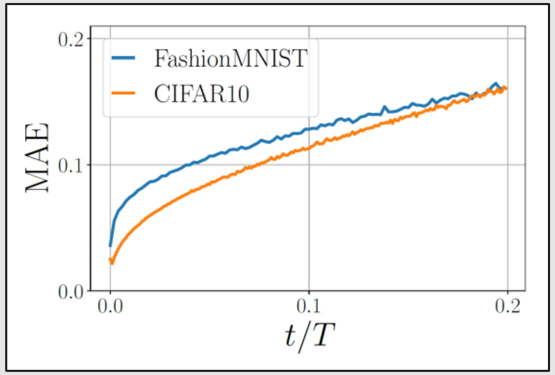


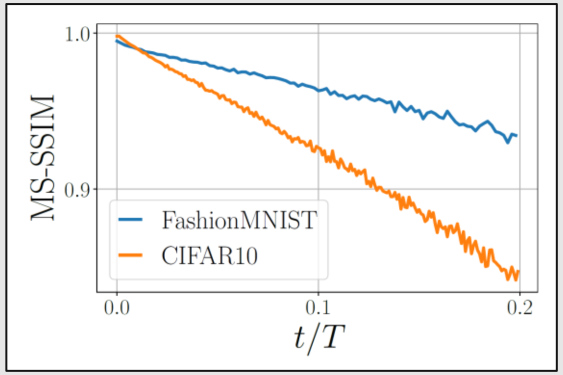
Linear schedule



Cosine schedule

Logarithm of the signal-to-noise ratio

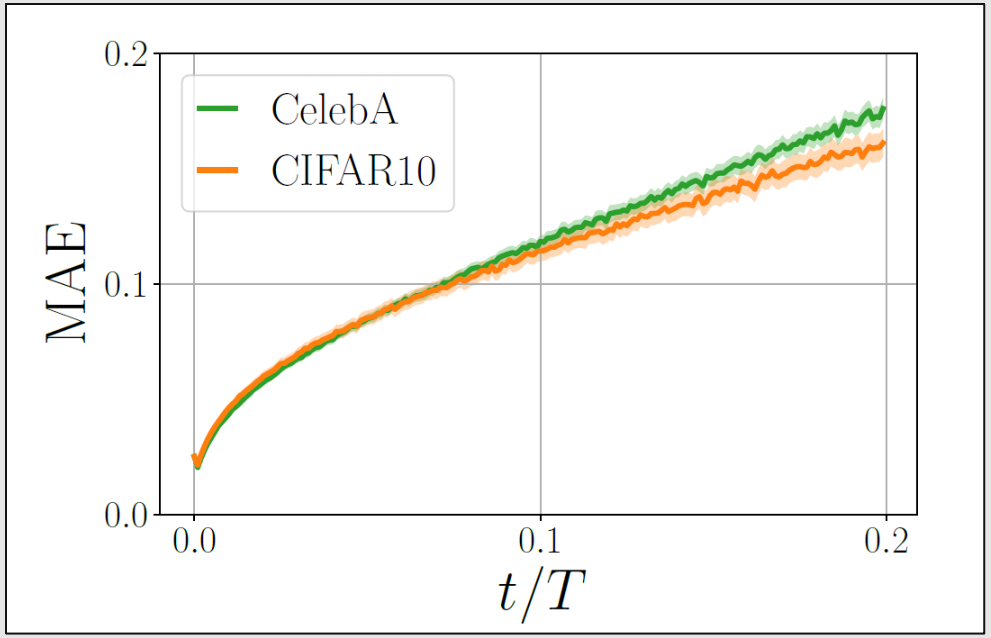




Reconstruction error from different diffusion step

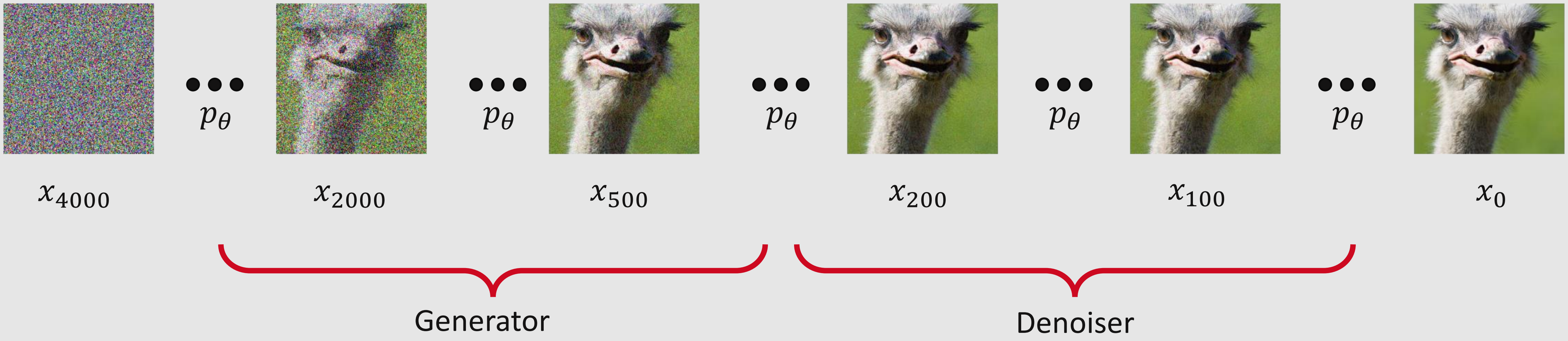
Do we need to know data distribution to remove noise?

- Reconstruction to original data sample with DDGM from a noised example does not require information about original data distribution
- Transition point between ***creation of new image features*** and ***removal of the remaining noise***

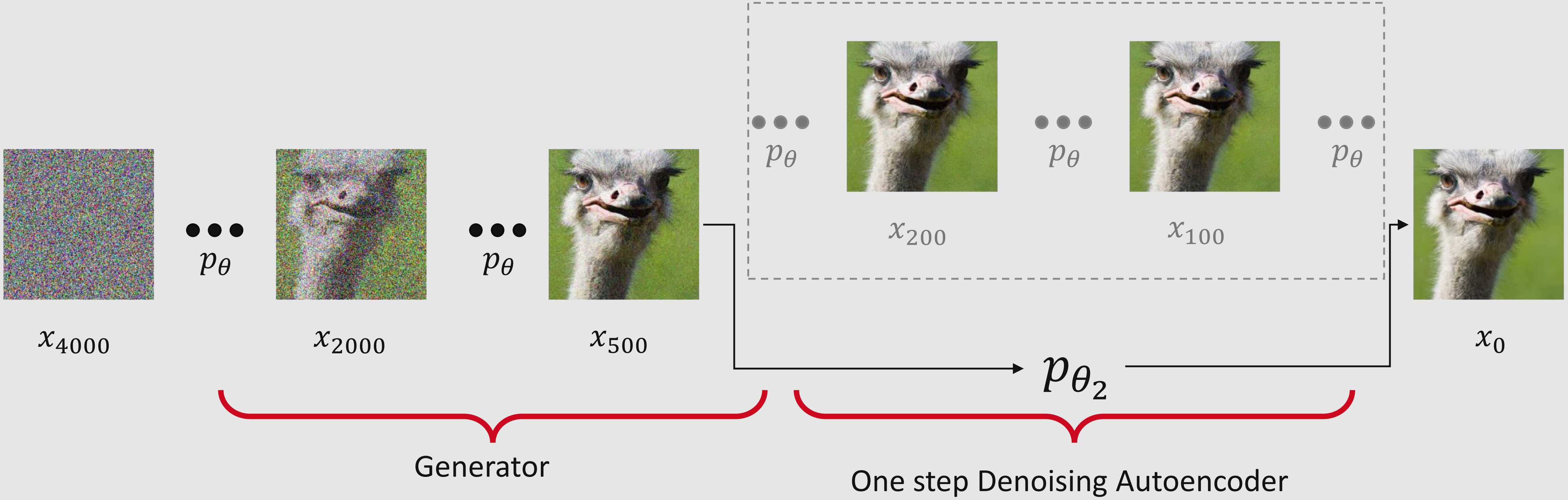


Reconstruction error for a DDGM trained on CIFAR10 and evaluated on different datasets

DDGM = Generator + Denoiser



DAED = DDGM + DAE



DAED formulation

We propose to bring a DDGM-based part into DAE for generating corrupted images.

The resulting combined loss:

$$\bar{\ell}(\mathbf{x}_0; \varphi, \theta) = \mathbb{E}_{\mathbf{x}_1 \sim q(\mathbf{x}_1 | \mathbf{x}_0)} \left[\ln p \left(\mathbf{x}_0 \mid f_\varphi(\mathbf{x}_1) \right) + \ln p_\theta(\mathbf{x}_1) \right]$$

$$\geq \underbrace{\mathbb{E}_{\mathbf{x}_1 \sim q(\mathbf{x}_1 | \mathbf{x}_0)} \left[\ln p \left(\mathbf{x}_0 \mid f_\varphi(\mathbf{x}_1) \right) \right]}_{\ell_{\mathrm{DAE}}(\mathbf{x}_0; \varphi)} + \underbrace{\mathbb{E}_{q(\mathbf{x}_2, \ldots, \mathbf{x}_T | \mathbf{x}_1)} \left[\frac{\ln p_\theta(\mathbf{x}_1, \ldots, \mathbf{x}_T)}{q(\mathbf{x}_1, \ldots, \mathbf{x}_T | \mathbf{x}_0)} \right]}_{\ell_{\mathrm{D}}(\mathbf{x}_0; \theta)},$$

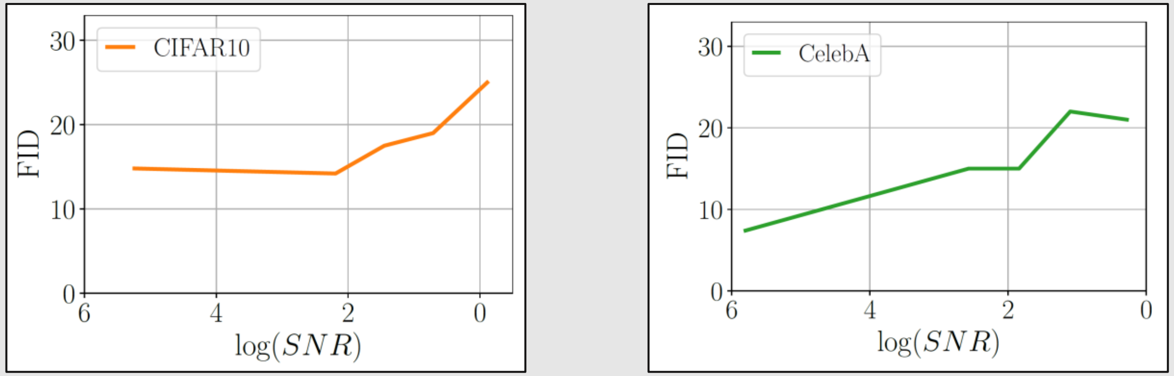
Differences between DAED and DDGM:

- We can control the amount of noise in $q(x_1|x_0)$
- We use two different parametrizations
- In the DAED, we introduce the explicit *denoiser*

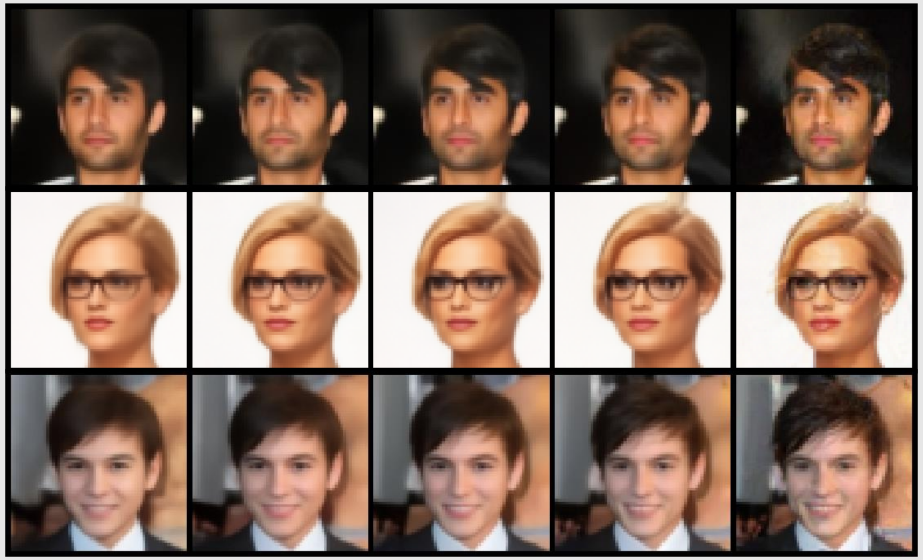
Experiments

How does the selection of splitting point affect the performance

We experimentally show that with up to 10% of diffusion steps replaced with DAE we can observe no significant drop in model's performance



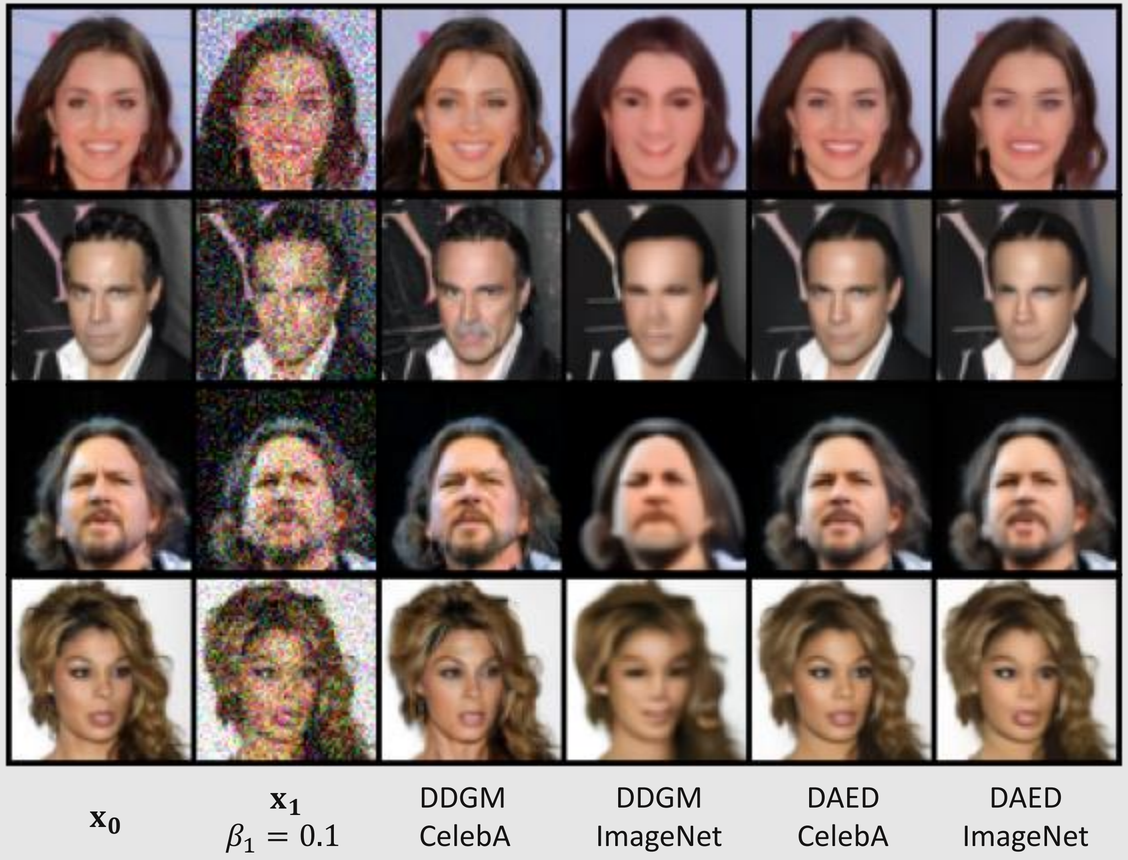
Quality of generations (FID) for DAED with different switching points selected according to log(SNR)



$\beta_1 = 0.2$ $\log(SNR)$ 0.3 | $\beta_1 = 0.1$ $\log(SNR)$ 1.1 | $\beta_1 = 0.05$ $\log(SNR)$ 1.8 | $\beta_1 = 0.025$ $\log(SNR)$ 2.6 | $\beta_1 = 0.001$ $\log(SNR)$ 5.8

Transferability of noise removal between data distributions

- DDGMs trained on one dataset can be used to remove noise on examples from an entirely different distribution
- DAED generalize can even better remove noise from unseen data distribution



Conclusion

- We observe and experimentally validate that it is reasonable to understand DDGMs as a combination of *generator* and *denoiser*
- We propose DAED – a new setup that is explicitly build as a combination of generative DDGM and DAE
- DAED performs on par with standard DDGM
- Finally, we present that DDGMs, and DAED especially, generalize well to unseen data