FEEC UNICAMP

Presentation 4
IA376N 2s2025

Denoising Diffusion Probabilistic Models (NeurIPS 2020)
Ho, Jain & Abbeel (UC Berkeley)

Henrique Parede de Souza - 260497

Background

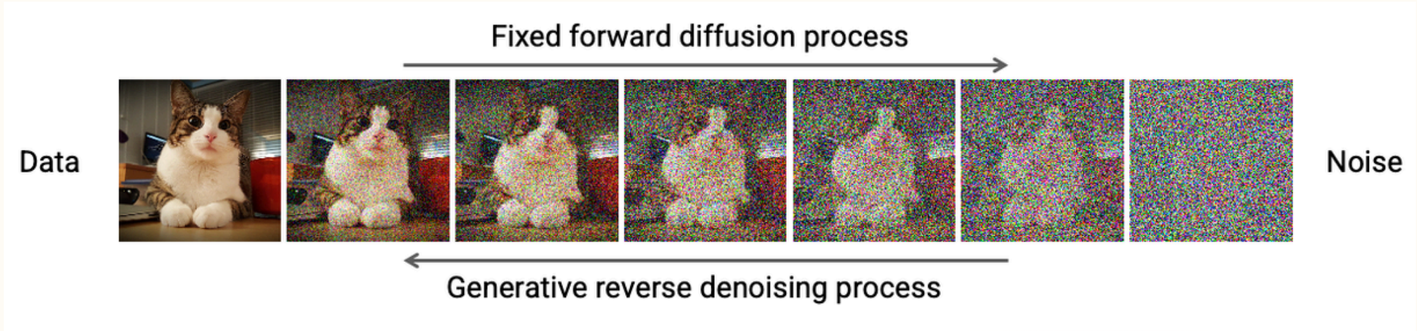
Previous architecture

- Diffusion Models originally proposed in the paper Deep Unsupervised Learning using Nonequilibrium Thermodynamics (Sohl-Dickstein et al, 2015).
- Although they are efficient to train, there has been **no demonstration** that they were capable of generating high quality samples.

Main contributions of new paper

- Capability of generating high quality samples, sometimes even better than GANs and VAEs.
- **New parameterization** for diffusion models.

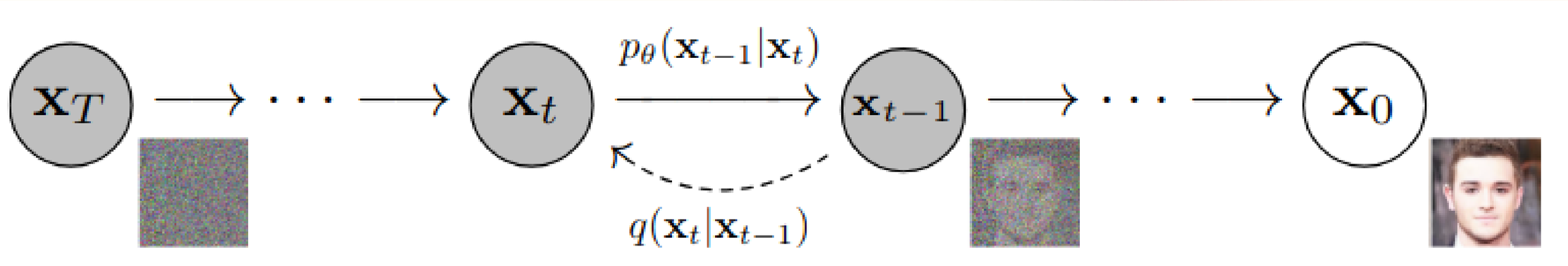
Diffusion Probabilistic Models (DDPM)



- Parameterized Markov chain trained using variational inference to produce samples matching the data.
- Gradually **adds Gaussian noise** to the data in the opposite direction of sampling until signal is destroyed.
- Neural network learns to **reverse the diffusion** process.

Markov Chain

Next stage only depends on the **previous state**.



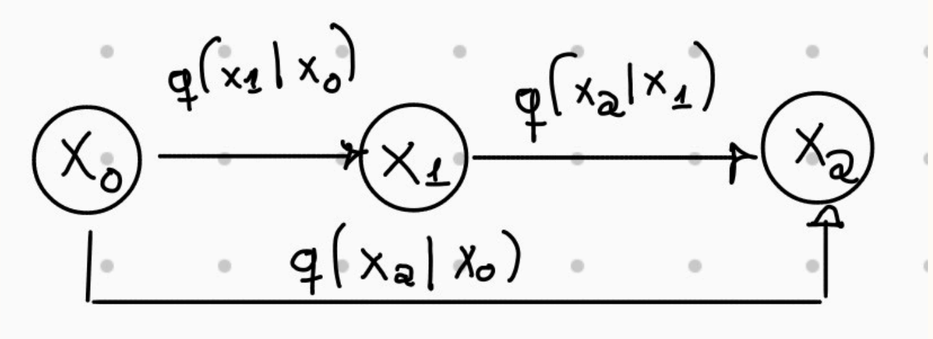
Forward process (right to left) and Reverse process (left to right).

$q(x_t|x_{t-1})$: likelihood obtaining a noisy image given previous state.

$p_\theta(x_{t-1}|x_t)$: likelihood obtaining denoised version given previous state.

Forward Process

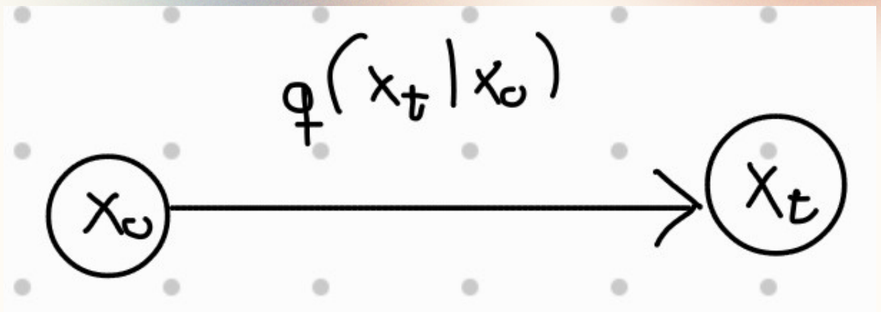
Adds Gaussian noise at each step from the beginning image x_0.



$$x_1 = x_0 + \beta e$$
$$\Rightarrow q(x_1|x_0) = \mathcal{N}(x_0, \beta)$$

$$x_2 = x_1 + \beta e$$
$$\Rightarrow q(x_2|x_1) = \mathcal{N}(x_1, \beta)$$

$$q(x_2|x_0) = \mathcal{N}(x_0, 2\beta)$$



Jump to any state by adding noises:

$$x_t = x_0 + t\beta e$$
$$\Rightarrow q(x_t|x_0) = \mathcal{N}(x_0, t\beta)$$

Forward Process

- We aim to obtain a final distribution as close as possible to a Gaussian

$$q(x_t|x_0) \to \mathcal{N}(0,1)$$

- Currently, the final distribution does not converge (**Exploding Variance**)

$$\mathcal{N}(x_0, t\beta) \nrightarrow \mathcal{N}(0,1)$$

Fix: change the iteration process

$$q(x_t \mid x_{t-1}) = \sqrt{1-\beta} \cdot x_{t-1} + \beta \cdot \epsilon$$

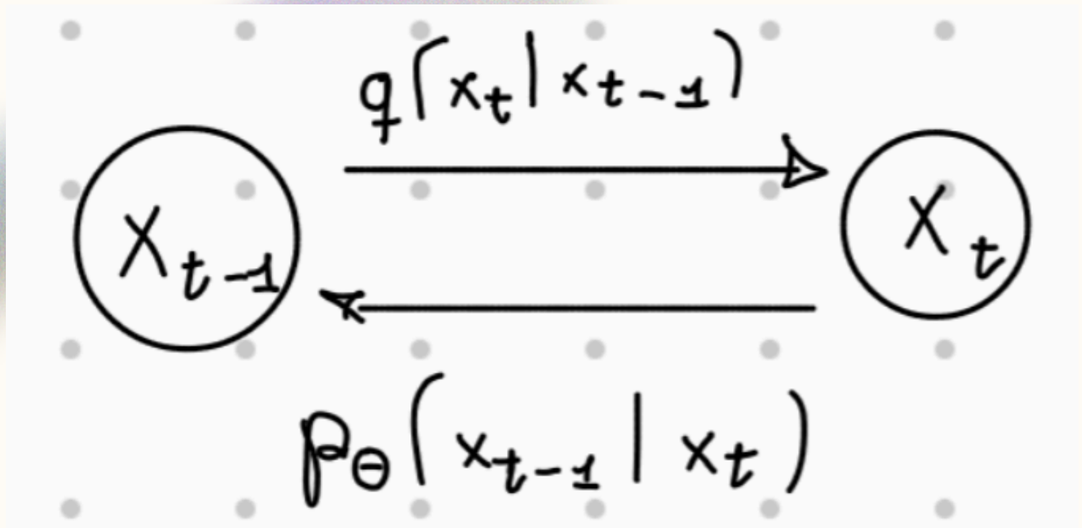
let $\bar{\alpha}_t = (1-\beta)^t$ so

$$q(x_t \mid x_0) = \sqrt{\bar{\alpha}_t}\, x_0 + (1-\bar{\alpha}_t)\epsilon$$

Reverse Process

- We aim to construct a neural network w/ parameters θ that learns to remove noise.
- Goal: maximize the likelihood of retrieving x_0 from pure noise.
- Same as minimize the likelihood of not retrieving x_0.
- Uses Negative Log-Likelihood:

$$-\log p_0(x_0)$$



Some notation

Complete forward process

$$q(x_{1:T} \mid x_0) = \prod_{i=1}^{T} q(x_i \mid x_{i-1})$$

Complete reverse process

$$p_\theta(x_{0:T}) = p_\theta(x_T) \prod_{i=1}^{T} p_\theta(x_{i-1} \mid x_i)$$

Deriving objective function

For the reverse neural network.

$$-\log p_\theta(x_0) = -\log \int p_\theta(x_{0:T})\, dx_{1:T}$$

Sum of all possible paths between x_T and x_0

$$= -\log \int q(x_{1:T} \mid x_0) \frac{p_\theta(x_{0:T})}{q(x_{1:T} \mid x_0)}\, dx_{1:T}$$

$$\mathbb{E}_x[f(x)] = \int x\, f(x)\, dx$$

$$\Rightarrow -\log p_\theta(x_0) = -\log \mathbb{E}_{q(x_{1:T}|x_0)} \left[\frac{p_\theta(x_{0:T})}{q(x_{1:T} \mid x_0)} \right]$$

$$\log \mathbb{E}(x) \leq \mathbb{E}(\log x)$$

$$\Rightarrow -\log p_\theta(x_0) \leq -\mathbb{E}_{q(x_{1:T}|x_0)} \left[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T} \mid x_0)} \right]$$

Evidence Lower Bound (**ELBO**)

Deriving objective function

$$-\log p_\theta(x_0) \leq -\mathbb{E}_{q(x_{1:T}|x_0)}\left[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T} \mid x_0)}\right]$$

Evidence Lower Bound
(**ELBO**)

Upper bound for NLL → Lower bound for maximizing the likelihood

After some derivations …

$$-\mathbb{E}_q\left[D_{\mathrm{KL}}\big(q(x_T \mid x_0) \,\|\, p(x_T)\big) + \sum_{t>1} D_{\mathrm{KL}}\big(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t)\big) - \log p_\theta(x_0 \mid x_1)\right]$$

Doesn't depend on θ

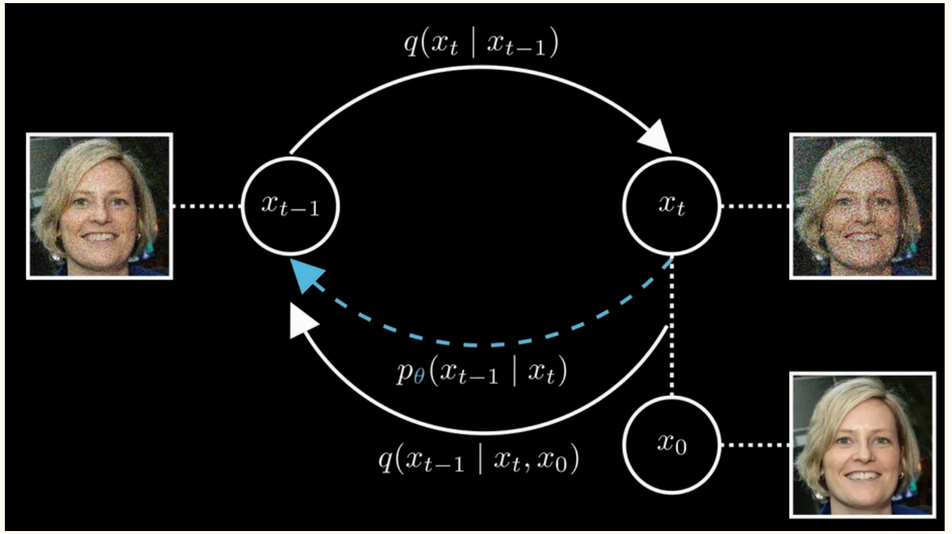
First iteration
has low influence

Deriving objective function

$$-\mathbb{E}_q \left[\sum_{t>1} D_{\mathrm{KL}} \big(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t) \big) \right]$$

$q(x_{t-1} \mid x_t, x_0)$: reconstruction based on the real image (**true posterior**).

$p_\theta(x_{t-1} \mid x_t)$: network generated reconstruction

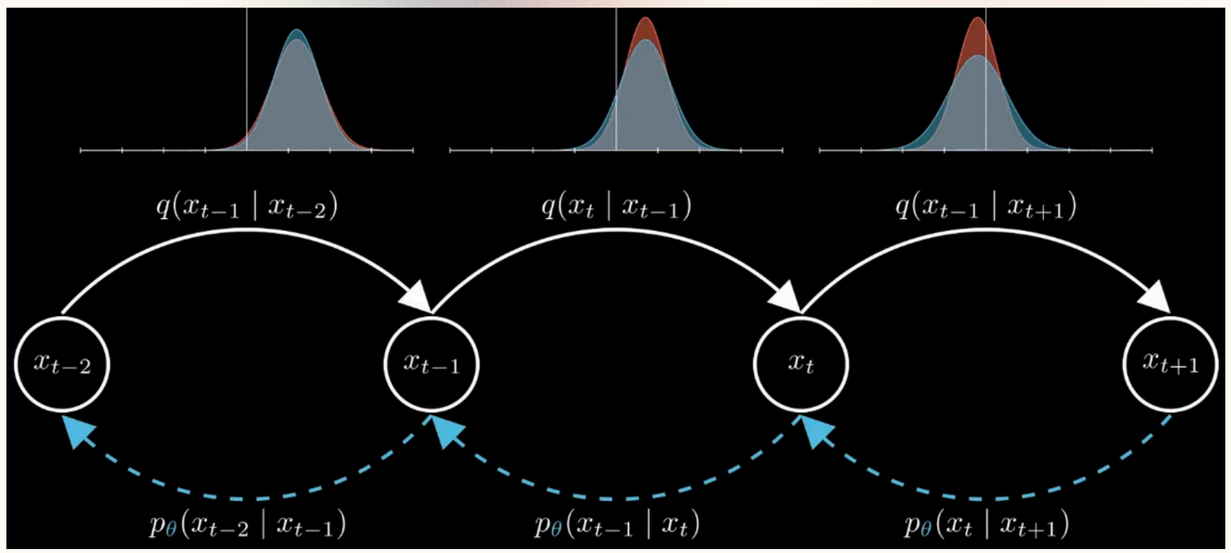


Source

Deriving objective function

We can simplify the two distributions as Gaussians:

$$q(x_{t-1} \mid x_t, x_0) = \mathcal{N}(\tilde{\mu}_t, \tilde{\beta}_t)$$
$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(\mu_\theta, \sigma_t^2)$$



Source

Therefore, the objective is to minimize the distance between the means of generated distribution and the true posterior, for each step:

$$\text{ELBO} = -\mathbb{E}_q \left[\sum_{t>1} \frac{1}{2\sigma_t^2} \|\tilde{\mu}_t - \mu_\theta(x_t, t)\|^2 \right]$$

Deriving objective function

We know that:

$$\tilde{\mu}_t = \frac{\sqrt{\bar{\alpha}_{t-1}}\,\beta_t}{1-\bar{\alpha}_t} x_0 + \frac{\sqrt{\alpha_t}\,(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} x_t$$

This expression is not convenient. Using the reparametrization trick: $x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\,\varepsilon$

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\varepsilon_\theta(x_t, t)\right) \quad \text{e} \quad \tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon\right)$$

Back to ELBO: $$\mathcal{L} = \mathbb{E}_q \left[\sum_{t>1} \frac{\beta_t^2}{2\,\sigma_t^2\,\alpha_t\,(1-\bar{\alpha}_t)} \left\| \varepsilon - \varepsilon_\theta(x_t, t) \right\|^2 \right]$$

Engineering trick!

Deriving objective function

Finally, our final loss is:

$$\mathcal{L} = \mathbb{E}_q \left[\|\varepsilon - \varepsilon_\theta(x_t, t)\|^2 \right]$$

Only parametrized in terms of the noise.

Training and Sampling

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** $\mathbf{x}_0$

Results

Sample quality

The model achieves better sample quality than most models in the literature.

Table 1: CIFAR10 results. NLL measured in bits/dim.

Model	IS	FID	NLL Test (Train)
Conditional			
EBM [11]	8.30	37.9	
JEM [17]	8.76	38.4	
BigGAN [3]	9.22	14.73	
StyleGAN2 + ADA (v1) [29]	**10.06**	**2.67**	
Unconditional			
Diffusion (original) [53]			$\leq$ 5.40
Gated PixelCNN [59]	4.60	65.93	3.03 (2.90)
Sparse Transformer [7]			**2.80**
PixelIQN [43]	5.29	49.46	
EBM [11]	6.78	38.2	
NCSNv2 [56]		31.75	
NCSN [55]	8.87$\pm$0.12	25.32	
SNGAN [39]	8.22$\pm$0.05	21.7	
SNGAN-DDLS [4]	9.09$\pm$0.10	15.42	
StyleGAN2 + ADA (v1) [29]	**9.74** $\pm$ 0.05	3.26	
Ours (L, fixed isotropic $\boldsymbol{\Sigma}$)	7.67$\pm$0.13	13.51	$\leq$ 3.70 (3.69)
Ours (L_{simple})	9.46$\pm$0.11	**3.17**	$\leq$ 3.75 (3.72)

Results

Reverse process parameterization

Predicting ϵ performs approximately as well as predicting μ when trained on the variational bound with fixed variances, but much better when trained with our simplified objective.

Table 2: Unconditional CIFAR10 reverse process parameterization and training objective ablation. Blank entries were unstable to train and generated poor samples with out-of-range scores.

Objective	IS	FID
$\tilde{\mu}$ prediction (baseline)		
L, learned diagonal Σ	7.28 ± 0.10	23.69
L, fixed isotropic Σ	8.06 ± 0.09	13.22
$\|\tilde{\mu} - \tilde{\mu}_\theta\|^2$	–	–
ϵ prediction (ours)		
L, learned diagonal Σ	–	–
L, fixed isotropic Σ	7.67 ± 0.13	13.51
$\|\tilde{\epsilon} - \epsilon_\theta\|^2$ (L_{simple})	$\mathbf{9.46 \pm 0.11}$	**3.17**

Results

Progressive generation from random bits

Figure 6: Unconditional CIFAR10 progressive generation ($\hat{\mathbf{x}}_0$ over time, from left to right). Extended samples and sample quality metrics over time in the appendix (Figs. 10 and 14).

Results

Progressive generation from random bits

Figure 7: When conditioned on the same latent, CelebA-HQ 256×256 samples share high-level attributes. Bottom-right quadrants are $\mathbf{x}_t$, and other quadrants are samples from $p_\theta(\mathbf{x}_0|\mathbf{x}_t)$.

Conclusion

High quality image samples using diffusion models

- New parameterization for diffusion models acted as the main contribution from this article.

Comparable to SOTA models

- StyleGAN2 and most VAE models.

Difficult points

Understand mathematical formulation

Explaining the each step of diffusion

Thank you!