Variational Autoencoders

Arthur Leclaire

With illustrations and slides from several colleagues:
Bruno Galerne, Stéphane Lathuillière, Loïc Le Folgoc, Alasdair Newson

4AI07

Plan

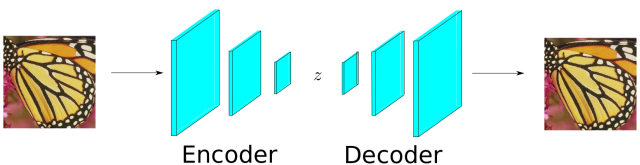
Autoencoders: The Gist

[Goodfellow et al., 2016]

- An autoencoder is a latent variable model
- The encoder will map images into a lower-dimensional latent space (feature space)
- The decoder will map latent codes back to images

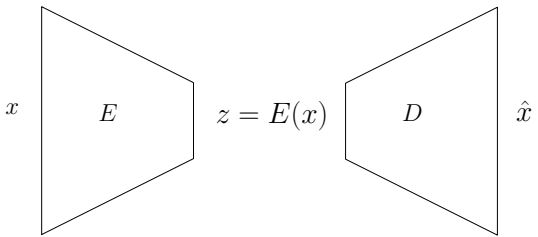
An autoencoder can be useful for

- Data compression
- Classification
- Outlier Detection
- Interpolation



Autoencoder Definition

- An autoencoder is a couple of neural networks (E, D).
- $E : \mathbf{R}^d \to \mathbf{R}^k$ is the encoder and decreases dimension (with $k < d$)
- $D : \mathbf{R}^k \to \mathbf{R}^d$ is the decoder



Autoencoder Learning

- The autoencoder is trained to have $\hat{x}$ look like x.
- In other words, we would like that $D(E(x)) \approx x$ (at least on the data points).

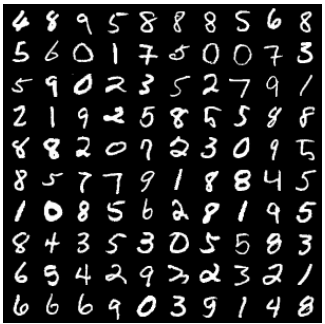
AUTOENCODING LOSS: Given N data points $x_1, \ldots, x_N \in \mathbf{R}^d$, minimize

$$\mathcal{L}_{AE}(D, E) = \sum_{n=1}^{N} \|D(E(x_n)) - x_n\|_2^2.$$

- We thus expect to have $D(E(x_n)) \approx x_n$ for any data point x_n.
- However, for other test points, $D(E(x))$ may drift away from x.

Example

Here is an example with an autoencoder trained with 1000 MNIST datapoints.
The dimension of feature space is set to $k = 2$.

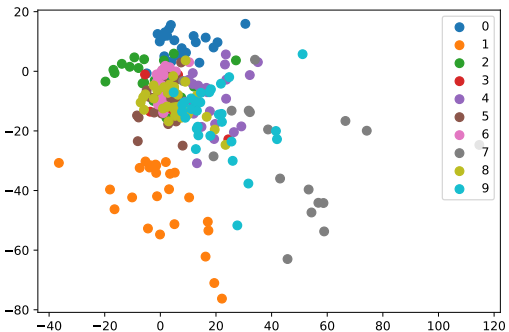


Input

Reconstruction

Example

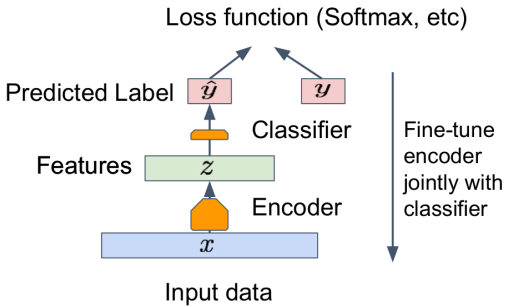
Here is an example with an autoencoder trained with 1000 MNIST datapoints.
The dimension of feature space is set to $k = 2$.



2D latent code of 256 test images

Classification based on Features

- Once trained, the autoencoder can be used for classification.
- For that, we can throw away the decoder, and train a classifier on feature representation.
- We may even fine-tune the decoder if needed.



Variants of Autoencoders

✓ Autoencoders are quite easy to train.

✗ Autoencoders may overfit, but not generalize properly.

✗ Autoencoders may have a non-structured latent space, that is,

$\rightarrow$ the directions of the latent space do not correspond to relevant features,

$\rightarrow$ variations in the latent space may not correspond to fluid variations in the image space

- In order to cope with that, many variants exist.
  Most of them consists in choosing another training loss function.

Sparse Autoencoders

- Many signal/image decomposition (e.g. wavelets) are based on sparse decompositions.
- Minimizing the ℓ^1 norm of the features $E(x)$ helps to sparsify the latent representation.
- We thus obtain "Sparse Autoencoders", associated with the following training loss:

$$\mathcal{L}_{AE}(D, E) = \sum_{n=1}^{N} \|D(E(x_n)) - x_n\|_2^2 + \lambda \|E(x_n)\|_1,$$

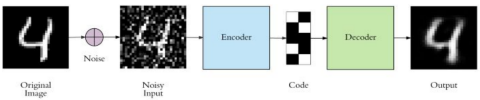
where $\lambda > 0$ is a parameter.

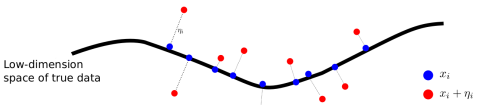
Denoising Autoencoders

- Denoising Autoencoders are AE trained to be less sensitive to noise in the data.
- For that, we change the training loss

$$\mathcal{L}_{DAE}(D, E) = \sum_{n=1}^{N} \|D(E(x_n + \varepsilon_n)) - x_n\|_2^2,$$

  where ε_n is a noise added to the data point x_n.
- This helps to learn a good low-dimensional model, and also better features.





Plan

Variational Autoencoders

[Kingma and Welling, 2019]

- One drawback of autoencoder is that they don't allow for sampling.
- We will cope with that by forcing an explicit distribution in the latent space.

VARIATIONAL AUTOENCODERS (VAE) are **latent variable models**.

When training a VAE, we will learn a distribution $p_\theta(x)$ that decomposes as

$$p_\theta(x) = \int_{\mathbf{R}^k} p_\theta(x|z)p_\theta(z)dz.$$

- The *prior* $p_\theta(z)$ may also depend on parameters, but is usually fixed (e.g. $\mathcal{N}(0, \mathrm{Id})$).
- $p_\theta(z|x)$ is called the posterior distribution of z knowing x

Deep latent variable

- A VAE is an example of deep latent variable model,
  that is, $p_\theta(x|z)$ is encoded by a neural network, for example:

$$p_\theta(x|z) \quad \text{is} \quad \mathcal{N}(\mu(z), \sigma^2(z)\text{Id})$$

  where $\mu(z)$ and $\sigma^2(z)$ are output of a neural network.

Deep latent variable

- A VAE is an example of deep latent variable model,
  that is, $p_\theta(x|z)$ is encoded by a neural network, for example:

$$p_\theta(x|z) \quad \text{is} \quad \mathcal{N}(\mu(z), \sigma^2(z)\text{Id})$$

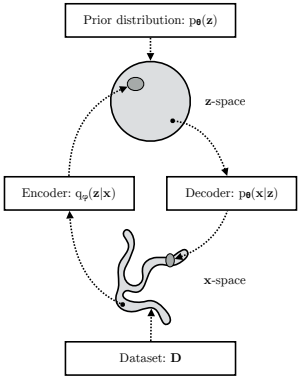
  where $\mu(z)$ and $\sigma^2(z)$ are output of a neural network.
- In such setting, $p_\theta(x) = \int p_\theta(x|z)p_\theta(z)dz$ is intractable.
- Therefore, we cannot perform directly maximum likelihood.

Deep latent variable

- A VAE is an example of deep latent variable model, that is, $p_\theta(x|z)$ is encoded by a neural network, for example:

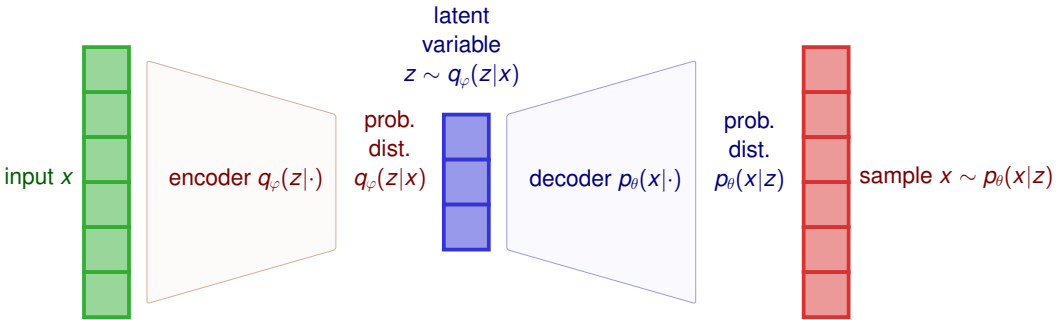
$$p_\theta(x|z) \quad \text{is} \quad \mathcal{N}(\mu(z), \sigma^2(z)\text{Id})$$

  where $\mu(z)$ and $\sigma^2(z)$ are output of a neural network.
- In such setting, $p_\theta(x) = \int p_\theta(x|z)p_\theta(z)dz$ is intractable.
- Therefore, we cannot perform directly maximum likelihood.
- **Main idea of VAE:** train jointly
  · a stochastic decoder $p_\theta(x|z)$
  · a stochastic encoder $q_\phi(z|x)$
  so that $q_\phi(z|x)$ approximates the (intractable) posterior $p_\theta(z|x)$.



(source: Kingma and Welling, 2019)

VAE diagram


latent
variable
$z \sim q_\varphi(z|x)$
input x
encoder $q_\varphi(z|\cdot)$
prob.
dist.
$q_\varphi(z|x)$
decoder $p_\theta(x|\cdot)$
prob.
dist.
$p_\theta(x|z)$
sample $x \sim p_\theta(x|z)$

Evidence lower bound (ELBO)

- We would like to train VAE by maximizing the log-likelihood $\sum_{n=1}^{N} \log p_\theta(x_n)$.
- However, $p_\theta(x)$ is not accessible and thus we will only maximize a lower bound obtained by:

$$
\begin{aligned}
\forall x \in \mathbf{R}^d, \quad \log p_\theta(x) &= \mathbb{E}_{z\sim q_\varphi(z|x)}\left[\log p_\theta(x)\right] \\
&= \mathbb{E}_{z\sim q_\varphi(z|x)}\left[\log\left[\frac{p_\theta(x,z)}{p_\theta(z|x)}\right]\right] \\
&= \mathbb{E}_{z\sim q_\varphi(z|x)}\left[\log\left[\frac{p_\theta(x,z)}{p_\theta(z|x)}\,\frac{q_\varphi(z|x)}{q_\varphi(z|x)}\right]\right] \\
&= \mathbb{E}_{z\sim q_\varphi(z|x)}\left[\log\left[\frac{p_\theta(x,z)}{q_\varphi(z|x)}\right]\right] + \underbrace{\mathbb{E}_{z\sim q_\varphi(z|x)}\left[\log\left[\frac{q_\varphi(z|x)}{p_\theta(z|x)}\right]\right]}_{\mathrm{KL}(q_\varphi(z|x)\|p_\theta(z|x))\geq 0} \\
&\geq \mathbb{E}_{z\sim q_\varphi(z|x)}\left[\log\left[\frac{p_\theta(x,z)}{q_\varphi(z|x)}\right]\right] := L(\theta,\varphi,x) \quad \text{(ELBO)}
\end{aligned}
$$

Kullback–Leibler divergence

If p, q are two probability density functions on $\mathbf{R}^d$, we define

$$\mathrm{KL}(p \parallel q) = \int_{\mathbf{R}^d} \log\left(\frac{p(x)}{q(x)}\right) p(x)dx = \mathbb{E}_{x\sim p(x)}\left[\log\left(\frac{p(x)}{q(x)}\right)\right].$$

Kullback–Leibler divergence

If p, q are two probability density functions on $\mathbf{R}^d$, we define

$$\mathrm{KL}(p \parallel q) = \int_{\mathbf{R}^d} \log\left(\frac{p(x)}{q(x)}\right) p(x)dx = \mathbb{E}_{x\sim p(x)}\left[\log\left(\frac{p(x)}{q(x)}\right)\right].$$

Main properties:

- $\mathrm{KL}(p \parallel q) \geq 0$ and $\mathrm{KL}(p \parallel q) = 0 \iff p = q$
- $\mathrm{KL}(p \parallel q) \neq \mathrm{KL}(q \parallel p)$
- $\lim_{n\to+\infty} \mathrm{KL}(p_n \parallel q) = 0$ implies convergence in distribution (and even in total variation).

Evidence lower bound (ELBO)

The **Evidence lower bound (ELBO)** is defined by

$$L(\theta, \varphi, x) = \mathbb{E}_{z \sim q_\varphi(z|x)} \left[\log \frac{p_\theta(x, z)}{q_\varphi(z|x)} \right] = \log p_\theta(x) - \mathrm{KL}(q_\varphi(z|x) \parallel p_\theta(z|x)) \leq \log p_\theta(x).$$

- The KL-divergence $\mathrm{KL}(q_\varphi(z|x) \parallel p_\theta(z|x))$ gives the tightness of the lower bound: the better is the approximation of the true posterior, the tighter is the lower bound.
- Main contribution of VAE [Kingma and Welling, 2014]: **Use the ELBO $L(\theta, \varphi, x)$ as a training loss** for improving the log-likelihood.
- To use $L(\theta, \varphi, x)$ as a training loss using SGD we need to compute unbiased estimators of both

$$\nabla_\theta L(\theta, \varphi, x) \quad \text{and} \quad \nabla_\varphi L(\theta, \varphi, x).$$

Evidence lower bound (ELBO)

$$
\begin{aligned}
L(\theta, \varphi, x) &= \mathbb{E}_{z \sim q_\varphi(z|x)} \left[\log \left[\frac{p_\theta(x, z)}{q_\varphi(z|x)} \right] \right] \\
&= \mathbb{E}_{z \sim q_\varphi(z|x)} \left[\log p_\theta(x, z) \right] - \mathbb{E}_{z \sim q_\varphi(z|x)} \left[\log q_\varphi(z|x) \right]
\end{aligned}
$$

Evidence lower bound (ELBO)

$$
\begin{aligned}
L(\theta, \varphi, x) &= \mathbb{E}_{z \sim q_\varphi(z|x)} \left[\log \left[\frac{p_\theta(x, z)}{q_\varphi(z|x)} \right] \right] \\
&= \mathbb{E}_{z \sim q_\varphi(z|x)} \left[\log p_\theta(x, z) \right] - \mathbb{E}_{z \sim q_\varphi(z|x)} \left[\log q_\varphi(z|x) \right]
\end{aligned}
$$

Unbiased estimator for $\nabla_\theta L(\theta, \varphi, x)$:

$$
\nabla_\theta L(\theta, \varphi, x) = \mathbb{E}_{z \sim q_\varphi(z|x)} \left[\nabla_\theta \log p_\theta(x, z) \right] \simeq \nabla_\theta \log p_\theta(x, z^{(1)}) \quad \text{where} \quad z^{(1)} \sim q_\varphi(z|x).
$$

Recall that $p_\theta(x, z) = p_\theta(z) p_\theta(x|z)$ is a known parametric function (involving the stochastic decoder) that can be (automatically) differentiated wrt θ.

Evidence lower bound (ELBO)

$$L(\theta, \varphi, x) = \mathbb{E}_{z \sim q_\varphi(z|x)} \left[\log \left[\frac{p_\theta(x, z)}{q_\varphi(z|x)} \right] \right]$$
$$= \mathbb{E}_{z \sim q_\varphi(z|x)} [\log p_\theta(x, z)] - \mathbb{E}_{z \sim q_\varphi(z|x)} [\log q_\varphi(z|x)]$$

Unbiased estimator for $\nabla_\theta L(\theta, \varphi, x)$:

$$\nabla_\theta L(\theta, \varphi, x) = \mathbb{E}_{z \sim q_\varphi(z|x)} [\nabla_\theta \log p_\theta(x, z)] \simeq \nabla_\theta \log p_\theta(x, z^{(1)}) \quad \text{where} \quad z^{(1)} \sim q_\varphi(z|x).$$

Recall that $p_\theta(x, z) = p_\theta(z) p_\theta(x|z)$ is a known parametric function (involving the stochastic decoder) that can be (automatically) differentiated wrt θ.

Unbiased estimator for $\nabla_\varphi L(\theta, \varphi, x)$:

- Not as straightforward since the ELBO expectation is taken with respect to $q_\varphi(z|x)$ that depends on φ!

Evidence lower bound (ELBO)

Reparameterization trick:

- Hypothesis: There exist a random variable ϵ and a deterministic function g such that

$$\forall x, \varphi, \quad z = g(\epsilon, \varphi, x) \sim q_\varphi(z|x).$$

- The function g decouples the randomness source and the parameters for simulating the approximate posterior $q_\varphi(z|x)$.

Example of Gaussian stochastic encoder:

- $q_\varphi(z|x)$ is $\mathcal{N}(\mu_\varphi(x), \mathrm{diag}(\sigma^2_\varphi(x)))$ with $(\mu_\varphi(x), \log \sigma_\varphi(x)) = \mathrm{NN}_\varphi(x)$.
- With $\epsilon \sim \mathcal{N}(0, \mathrm{Id})$ the standard Gaussian distribution:

$$z = \mu_\varphi(x) + \sigma_\varphi(x) \odot \epsilon \quad \sim \quad \mathcal{N}(\mu_\varphi(x), \mathrm{diag}(\sigma^2_\varphi(x))).$$

Evidence lower bound (ELBO)

Reparameterization trick:

- Hypothesis: There exist a random variable ϵ and a deterministic function g such that

$$\forall x, \varphi, \quad z = g(\epsilon, \varphi, x) \sim q_\varphi(z|x).$$

Evidence lower bound (ELBO)

Reparameterization trick:

- Hypothesis: There exist a random variable ϵ and a deterministic function g such that

$$\forall x, \varphi, \quad z = g(\epsilon, \varphi, x) \sim q_\varphi(z|x).$$

Change of variable in the ELBO:

$$\begin{aligned} L(\theta, \varphi, x) &= \mathbb{E}_{z \sim q_\varphi(z|x)} \left[\log p_\theta(x, z)\right] - \mathbb{E}_{z \sim q_\varphi(z|x)} \left[\log q_\varphi(z|x)\right] \\ &= \mathbb{E}_{\epsilon \sim p(\epsilon)} \left[\log p_\theta(x, g(\epsilon, \varphi, x))\right] - \mathbb{E}_{\epsilon \sim p(\epsilon)} \left[\log q_\varphi(g(\epsilon, \varphi, x)|x)\right] \end{aligned}$$

Evidence lower bound (ELBO)

Reparameterization trick:

- Hypothesis: There exist a random variable ϵ and a deterministic function g such that

$$\forall x, \varphi, \quad z = g(\epsilon, \varphi, x) \sim q_\varphi(z|x).$$

Change of variable in the ELBO:

$$\begin{aligned} L(\theta, \varphi, x) &= \mathbb{E}_{z \sim q_\varphi(z|x)} \left[\log p_\theta(x, z)\right] - \mathbb{E}_{z \sim q_\varphi(z|x)} \left[\log q_\varphi(z|x)\right] \\ &= \mathbb{E}_{\epsilon \sim p(\epsilon)} \left[\log p_\theta(x, g(\epsilon, \varphi, x))\right] - \mathbb{E}_{\epsilon \sim p(\epsilon)} \left[\log q_\varphi(g(\epsilon, \varphi, x)|x)\right] \end{aligned}$$

Unbiased estimator for $\nabla_\varphi L(\theta, \varphi, x)$:

- Draw $\epsilon^{(1)} \sim p(\epsilon)$ and (automatically) differentiate wrt φ the expression

$$\log p_\theta(x, g(\epsilon^{(1)}, \varphi, x)) - \log q_\varphi(g(\epsilon^{(1)}, \varphi, x)|x)$$

- Same for differentiating wrt θ.

VAE Training Algorithm

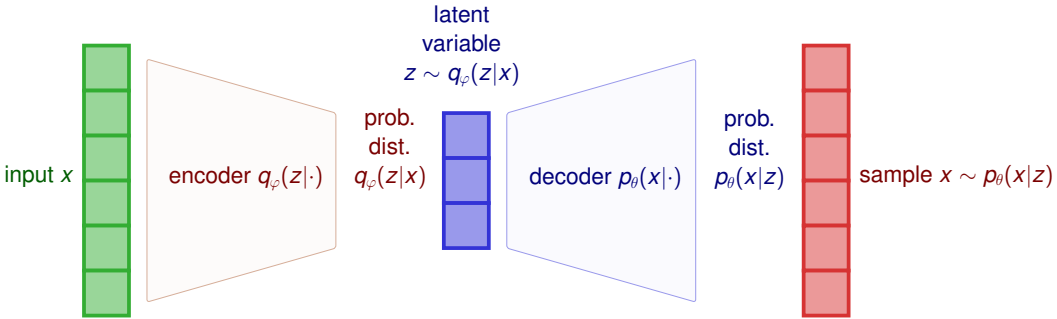
For K iterations, repeat the following steps

1. Draw a minibatch $\mathcal{M} = \{x_{i_1}, \ldots, x_{i_M}\}$ of M samples from $\mathcal{D} = \{x_n,\ n = 1, \ldots, N\}$
2. Draw M random $\epsilon_m \sim p(\epsilon)$, $m = 1, \ldots, M$.
3. Compute $z_m = g(\epsilon_m, \varphi, x_{i_m}) \sim q_\varphi(z|x_{i_m})$ using the encoder network parameters.
4. Apply the decoder network to each latent variable z^m and return

$$\tilde{L}(\theta, \varphi, \mathcal{M}) = \frac{1}{M} \sum_{m=1}^{M} \log p_\theta(x_{i_m}, g(\epsilon_m, \varphi, x_{i_m})) - \log q_\varphi(g(\epsilon_m, \varphi, x_{i_m})|x_{i_m})$$

5. Compute $\nabla_\theta \tilde{L}(\theta, \varphi, \mathcal{M})$ and $\nabla_\varphi \tilde{L}(\theta, \varphi, \mathcal{M})$ by automatic differentiation
6. Update the parameters θ and φ with one SGD step.

Gaussian VAE



Example of Gaussian stochastic encoder and decoder:

- $p_\theta(z)$ is $\mathcal{N}(0, \mathrm{Id})$ (fixed Gaussian, no parameter to learn).
- $q_\varphi(z|x)$ is $\mathcal{N}(\mu_\varphi(x), \mathrm{diag}(\sigma^2_\varphi(x)))$ with $(\mu_\varphi(x), \log \sigma_\varphi(x)) = \mathrm{NN}_\varphi(x)$.
- $p_\theta(x|z)$ is $\mathcal{N}(\mu_\theta(z), s^2\mathrm{Id})$ with $\mu_\theta(z) = \mathrm{NN}_\theta(z)$.
- The architectures for NN_φ and NN_θ are generally chosen symmetric.

VAE: ELBO and Kullback–Leibler divergence

The “latent code regularization” is better seen by **refactorizing the ELBO**:

$$
\begin{aligned}
L(\theta, \varphi, x) &= \mathbb{E}_{z\sim q_\varphi(z|x)}\left[\log\left[\frac{p_\theta(x,z)}{q_\varphi(z|x)}\right]\right] \\
&= \mathbb{E}_{z\sim q_\varphi(z|x)}\left[\log\left[\frac{p_\theta(z)p_\theta(x|z)}{q_\varphi(z|x)}\right]\right] \\
&= \mathbb{E}_{z\sim q_\varphi(z|x)}\left[\log p_\theta(x|z)\right] + \mathbb{E}_{z\sim q_\varphi(z|x)}\left[\log\left[\frac{p_\theta(z)}{q_\varphi(z|x)}\right]\right] \\
&= \underbrace{\mathbb{E}_{z\sim q_\varphi(z|x)}\left[\log p_\theta(x|z)\right]}_{\text{reconstruction error}} - \underbrace{\mathrm{KL}(q_\varphi(z|x)\parallel p_\theta(z))}_{\text{latent code regularization}}
\end{aligned}
$$

- The latent code regularization enforces all the approximate posterior to be close to the prior.
- And the supports of all distributions $q_\varphi(z|x)$ should be well adjusted.
- This results in an encoder-decoder with well-spread latent code distribution.

ELBO with Gaussian VAE

Density of a Gaussian distribution: Recall that the log-density of $\mathcal{N}(\mu, \Sigma)$ is

$$\log g_{\mu,\Sigma}(x) = -\frac{1}{2}\log(|2\pi\Sigma|) - \frac{1}{2}(x-\mu)^{\mathrm{T}}\Sigma^{-1}(x-\mu)$$

KL divergence between two Gaussian distributions on $\mathbf{R}^k$: is

$$\mathrm{KL}\left(\mathcal{N}(\mu_1, \Sigma_1) \parallel \mathcal{N}(\mu_2, \Sigma_2)\right) = \frac{1}{2}\left[\log\left(\frac{|\Sigma_2|}{|\Sigma_1|}\right) - k + \mathrm{Tr}(\Sigma_2^{-1}\Sigma_1) + (\mu_2-\mu_1)^T\Sigma_2^{-1}(\mu_2-\mu_1)\right].$$

Expression of the ELBO loss: When $q_\varphi(z|x)$ is $\mathcal{N}(\mu, \mathrm{diag}(\sigma^2))$ and $p_\theta(z)$ is $\mathcal{N}(0, \mathrm{Id})$, we get

$$\mathrm{KL}\left(q_\varphi(z|x) \parallel p_\theta(z)\right) = \frac{1}{2}\left[-\sum_i(\log(\sigma_i^2)+1) + \sum_i \sigma_i^2 + \sum_i \mu_i^2\right].$$

ELBO with Gaussian VAE

On the other hand, when $p_\theta(x|z)$ is $\mathcal{N}(\mu_\theta(z), s^2\mathrm{Id})$,

$$\log p_\theta(x|z) = -\frac{1}{2}\log(|2\pi s^2 \mathrm{Id}|) - \frac{1}{2s^2}\|x - \mu_\theta(z)\|^2.$$

Therefore,

$$-L(\theta, \varphi, x) = \mathbb{E}_{z\sim q_\varphi(z|x)}\left[-\log p_\theta(x|z)\right] + \mathrm{KL}(q_\varphi(z|x) \parallel p_\theta(z))$$

can be estimated by

$$-\tilde{L}(\theta, \varphi, x, z) = C + \frac{1}{2s^2}\|x - \mu_\theta(z)\|^2 + \frac{1}{2}\left[-\sum_i(\log(\sigma_{\varphi,i}(x)^2) + 1) + \sum_i \sigma_{\varphi,i}(x)^2 + \sum_i \mu_{\varphi,i}(x)^2\right],$$

where z is a sample of $q_\varphi(z|x)$.

ELBO with Gaussian VAE

On the other hand, when $p_\theta(x|z)$ is $\mathcal{N}(\mu_\theta(z), s^2\mathrm{Id})$,

$$\log p_\theta(x|z) = -\frac{1}{2}\log(|2\pi s^2 \mathrm{Id}|) - \frac{1}{2s^2}\|x - \mu_\theta(z)\|^2.$$

Therefore,

$$-L(\theta, \varphi, x) = \mathbb{E}_{z\sim q_\varphi(z|x)}\left[-\log p_\theta(x|z)\right] + \mathrm{KL}(q_\varphi(z|x) \parallel p_\theta(z))$$

can be estimated by

$$-\tilde{L}(\theta, \varphi, x, z) = C + \frac{1}{2s^2}\|x - \mu_\theta(z)\|^2 + \frac{1}{2}\left[-\sum_i(\log(\sigma_{\varphi,i}(x)^2) + 1) + \sum_i \sigma_{\varphi,i}(x)^2 + \sum_i \mu_{\varphi,i}(x)^2\right],$$

where z is a sample of $q_\varphi(z|x)$.

✓ You can use this expression for automatic differentiation with Pytorch!

AE v.s. VAE

Here is an example with AE and VAE trained with 1000 MNIST datapoints.
The dimension of feature space is set to $k = 2$.

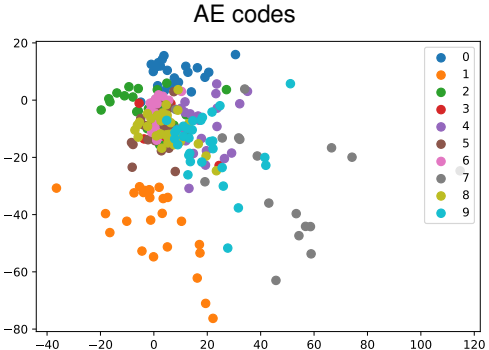
Input

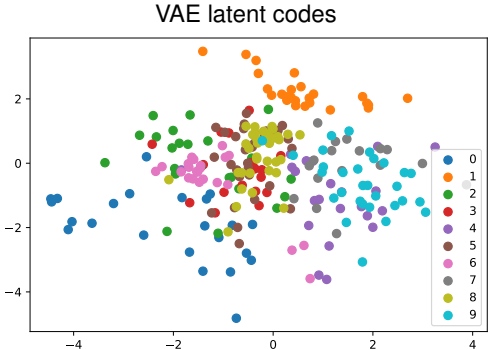
AE Output

VAE Output

AE v.s. VAE

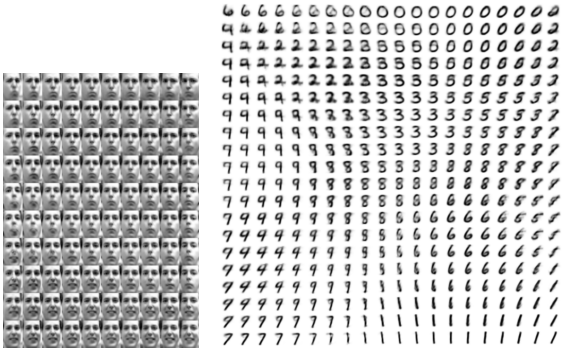
Here is an example with AE and VAE trained with 1000 MNIST datapoints.
The dimension of feature space is set to $k = 2$.





VAE: Example of Results

From the original paper: [Kingma and Welling, 2014]: “Auto-Encoding Variational Bayes”



(a) Learned Frey Face manifold

(b) Learned MNIST manifold

Figure 4: Visualisations of learned data manifold for generative models with two-dimensional latent space, learned with AEVB. Since the prior of the latent space is Gaussian, linearly spaced coordinates on the unit square were transformed through the inverse CDF of the Gaussian to produce values of the latent variables $\mathbf{z}$. For each of these values $\mathbf{z}$, we plotted the corresponding generative $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ with the learned parameters $\boldsymbol{\theta}$.

Comments on VAE

The training of VAEs is more stable than the one of WGANs.

However, there are still issues regarding VAE [Kingma and Welling, 2019, Tomczak, 2022]:

- *Posterior collapse*: All approximate posteriors $q_\varphi(z|x)$ are stucked to the prior.
- *Hole problem*: Some subset of the latent space is not populated by encoded data.
- *Blurriness of generative model*: produced images tend to be blurry.

Comments on VAE

The training of VAEs is more stable than the one of WGANs.

However, there are still issues regarding VAE [Kingma and Welling, 2019, Tomczak, 2022]:

- *Posterior collapse*: All approximate posteriors $q_\varphi(z|x)$ are stucked to the prior.
- *Hole problem*: Some subset of the latent space is not populated by encoded data.
- *Blurriness of generative model*: produced images tend to be blurry.

As for GANs, VAEs can also be conditioned on labels ℓ (e.g. digits, image sketch, attributes, etc). For that

- Replace $p_\theta(x|z)$ by $p_\theta(x|z,\ell)$
- Replace $q_\varphi(z|x)$ by $q_\varphi(z|x,\ell)$
- Train by adapting the loss function...

About Conditional VAEs, see for example [Sohn et al., 2015].

Hierarchical VAE

- VAE can be made competitive using well-designed architectures (e.g. hierarchical).

Example from [Vahdat and Kautz, 2020]: “NVAE: A Deep Hierarchical Variational Autoencoder”

- See also the architecture called Very Deep VAE (VDVAE) [Child, 2021].
- Hierarchical VAEs can also be used for diverse image restoration [Prost et al., 2023].

Summary and comments

- VAE is a stochatic variant of AE that allows for image sampling.
- VAEs are simple to train
- But simple VAEs tend to produce blurry images (compared to GANs)
- Hierarchical VAEs help to produce images to very high resolution.
- State-of-the-art image generation is based on (latent) diffusion models, that is, diffusion is the latent space of a VAE. Example: the Stable Diffusion model [Rombach et al., 2022].

THANK YOU FOR YOUR ATTENTION!

References I

Child, R. (2021).
Very deep {vae}s generalize autoregressive models and can outperform them on images.
In *International Conference on Learning Representations*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016).
Deep Learning.
MIT Press.
http://www.deeplearningbook.org.

Ho, J., Jain, A., and Abbeel, P. (2020).
Denoising diffusion probabilistic models.
In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851.

Kim, G., Kwon, T., and Ye, J. C. (2022).
Diffusionclip: Text-guided diffusion models for robust image manipulation.
In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2426–2435.

References II


Kingma, D. P. and Welling, M. (2014).
Auto-encoding variational Bayes.
In Bengio, Y. and LeCun, Y., editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.


Kingma, D. P. and Welling, M. (2019).
An introduction to variational autoencoders.
Foundations and Trends in Machine Learning, 12(4):307–392.


Prost, J., Houdard, A., Almansa, A., and Papadakis, N. (2023).
Inverse problem regularization with hierarchical variational autoencoders.
In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22894–22905.


Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022).
High-resolution image synthesis with latent diffusion models.
In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695.

References III

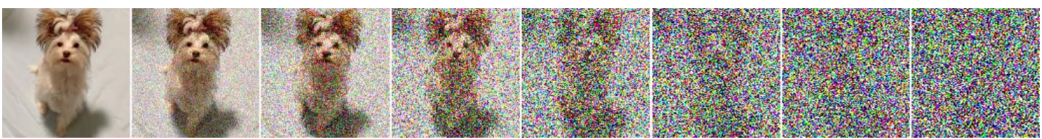

Sohn, K., Lee, H., and Yan, X. (2015).
Learning structured output representation using deep conditional generative models.
Advances in neural information processing systems, 28.


Tomczak, J. M. (2022).
Deep Generative Modeling.
Springer International Publishing, Cham.


Vahdat, A. and Kautz, J. (2020).
Nvae: A deep hierarchical variational autoencoder.
In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 19667–19679. Curran Associates, Inc.

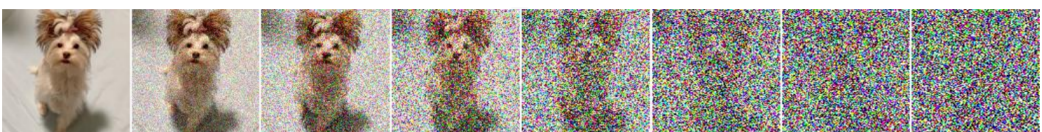
Generative Modeling with Diffusion

- Let q_0 be the empirical distribution associated to N data points in $\mathbf{R}^d$.
- Goal: design a generative model μ_θ that fits q_0.
- Starting remark: if $x_0 \sim q_0$ and $z \sim \mathcal{N}(0, \mathrm{Id})$ are independent, then $x_t = x_0 + \sqrt{t}z$ looks noisier and noisier as $t \to +\infty$.
  Also, the distribution of x_t "resembles more and more" to $\mathcal{N}(0, t\mathrm{Id})$.



Generative Modeling with Diffusion

- Let q_0 be the empirical distribution associated to N data points in $\mathbf{R}^d$.
- Goal: design a generative model μ_θ that fits q_0.
- Starting remark: if $x_0 \sim q_0$ and $z \sim \mathcal{N}(0, \mathrm{Id})$ are independent, then $x_t = x_0 + \sqrt{t}z$ looks noisier and noisier as $t \to +\infty$.
  Also, the distribution of x_t "resembles more and more" to $\mathcal{N}(0, t\mathrm{Id})$.



- Thus we can try to sample $\tilde{x}_T \sim \mathcal{N}(0, T\mathrm{Id})$ for a large $T > 0$, and inverse the noising process to get back $\tilde{x}_0$ with law $\approx q_0$.
- How to reverse in practice? Which denoiser to use?...

Denoising Diffusion Probabilistic Models (DDPM)

[Ho et al., 2020]

- DDPM is a discrete-time model defined on $[0:T] := \{0, 1, \dots, T\}$ with $T \in \mathbb{N}$ (large).
- It starts from a forward Markov chain $(x_t)_{t\in[0:T]}$ with distribution

$$q(x_0, \dots, x_T) = q(x_0)q(x_1|x_0)q(x_2|x_1)\dots q(x_T|x_{T-1})$$

where the $q(x_t|x_{t-1})$ corresponds to normalized noising.

Denoising Diffusion Probabilistic Models (DDPM)

[Ho et al., 2020]

- DDPM is a discrete-time model defined on $[0:T] := \{0, 1, \dots, T\}$ with $T \in \mathbb{N}$ (large).
- It starts from a forward Markov chain $(x_t)_{t\in[0:T]}$ with distribution

$$q(x_0, \dots, x_T) = q(x_0)q(x_1|x_0)q(x_2|x_1)\dots q(x_T|x_{T-1})$$

  where the $q(x_t|x_{t-1})$ corresponds to normalized noising.
- The final generative model will be a process $(\tilde{x}_t)_{t\in[0:T]}$ with distribution

$$p_\theta(\tilde{x}_0, \dots, \tilde{x}_T) = p_\theta(\tilde{x}_T)p_\theta(\tilde{x}_{T-1}|\tilde{x}_T)\dots p_\theta(\tilde{x}_0|\tilde{x}_1)$$

  where $p_\theta(\tilde{x}_T)$ is a Gaussian, and where $p_\theta(\tilde{x}_{t-1}|\tilde{x}_t)$ corresponds to normalized denoising.

  This backward transition relies on a learned neural network with parameters θ.

DDPM: forward process

Let $\alpha_t > 0$ and $\beta_t > 0$ be two sequences (expectation/variance schedules).

- $x_0 \sim q_0$ (data distribution)
- (x_t) is defined recursively:

$$x_t = \sqrt{\alpha_t} x_{t-1} + \sqrt{\beta_t} z_t, \tag{1}$$

  where $(z_t)_{t \geq 1}$ is a sequence of i.i.d. random variables of law $\mathcal{N}(0, \mathrm{Id})$.
- Therefore, $q(x_t | x_{t-1})$ is $\mathcal{N}(\sqrt{\alpha_t} x_{t-1}, \beta_t \mathrm{Id})$.

DDPM: forward process

Let $\alpha_t > 0$ and $\beta_t > 0$ be two sequences (expectation/variance schedules).

- $x_0 \sim q_0$ (data distribution)
- (x_t) is defined recursively:

$$x_t = \sqrt{\alpha_t} x_{t-1} + \sqrt{\beta_t} z_t, \tag{1}$$

  where $(z_t)_{t \geq 1}$ is a sequence of i.i.d. random variables of law $\mathcal{N}(0, \mathrm{Id})$.
- Therefore, $q(x_t | x_{t-1})$ is $\mathcal{N}(\sqrt{\alpha_t} x_{t-1}, \beta_t \mathrm{Id})$.

We can obtain the distribution $q(x_t | x_0)$:

- For each t, there exists $\epsilon_t \sim \mathcal{N}(0, \mathrm{Id})$ such that

$$x_t = \sqrt{\overline{\alpha}_t} x_0 + \sqrt{\overline{\beta}_t} \epsilon_t, \tag{2}$$

  where $(\overline{\alpha}_t), (\overline{\beta}_t)$ are defined recursively: $\overline{\alpha}_t = \overline{\alpha_{t-1}} \alpha_t$, and $\overline{\beta}_t = \alpha_t \overline{\beta_{t-1}} + \beta_t$.
- Therefore, $q(x_t | x_0) = \mathcal{N}(\sqrt{\overline{\alpha}_t} x_0, \overline{\beta}_t \mathrm{Id})$.

The Variance Schedule

Since

$$x_t = \sqrt{\overline{\alpha}_t} x_0 + \sqrt{\overline{\beta}_t} \epsilon_t$$

the covariance matrix of x_t is

$$\mathbb{V}\text{ar}(x_t) = \overline{\alpha}_t \mathbb{V}\text{ar}(x_0) + \overline{\beta}_t \text{Id}.$$

The Variance Schedule

Since

$$x_t = \sqrt{\overline{\alpha}_t} x_0 + \sqrt{\overline{\beta}_t} \epsilon_t$$

the covariance matrix of x_t is

$$\mathbb{V}\mathrm{ar}(x_t) = \overline{\alpha}_t \mathbb{V}\mathrm{ar}(x_0) + \overline{\beta}_t \mathrm{Id}.$$

- **Variance exploding (VE) choice:** $\alpha_t = 1$ and $\beta_t = \sqrt{\sigma_t^2 - \sigma_{t-1}^2}$.

  Therefore, if $\mathbb{V}\mathrm{ar}(x_0) = \sigma_0^2 \mathrm{Id}$, then $\quad \forall t \geq 0, \quad \mathbb{V}\mathrm{ar}(x_t) = \sigma_t^2 \mathrm{Id}$.

The Variance Schedule

Since

$$x_t = \sqrt{\overline{\alpha}_t} x_0 + \sqrt{\overline{\beta}_t} \epsilon_t$$

the covariance matrix of x_t is

$$\mathbb{V}\mathrm{ar}(x_t) = \overline{\alpha}_t \mathbb{V}\mathrm{ar}(x_0) + \overline{\beta}_t \mathrm{Id}.$$

- **Variance exploding (VE) choice:** $\alpha_t = 1$ and $\beta_t = \sqrt{\sigma_t^2 - \sigma_{t-1}^2}$.

  Therefore, if $\mathbb{V}\mathrm{ar}(x_0) = \sigma_0^2 \mathrm{Id}$, then $\quad \forall t \geq 0, \quad \mathbb{V}\mathrm{ar}(x_t) = \sigma_t^2 \mathrm{Id}$.

- **Variance preserving (VP) choice:** $\alpha_t + \beta_t = 1$.

  By induction, we can show that this implies $\overline{\alpha}_t + \overline{\beta}_t = 1$.

  Therefore, if $\mathbb{V}\mathrm{ar}(x_0) = \mathrm{Id}$, then $\quad \forall t \geq 0, \quad \mathbb{V}\mathrm{ar}(x_t) = \mathrm{Id}$.

The Variance Schedule

Since

$$x_t = \sqrt{\overline{\alpha}_t} x_0 + \sqrt{\overline{\beta}_t} \epsilon_t$$

the covariance matrix of x_t is

$$\mathbb{V}\mathrm{ar}(x_t) = \overline{\alpha}_t \mathbb{V}\mathrm{ar}(x_0) + \overline{\beta}_t \mathrm{Id}.$$

- **Variance exploding (VE) choice:** $\alpha_t = 1$ and $\beta_t = \sqrt{\sigma_t^2 - \sigma_{t-1}^2}$.

  Therefore, if $\mathbb{V}\mathrm{ar}(x_0) = \sigma_0^2 \mathrm{Id}$, then $\quad \forall t \geq 0, \quad \mathbb{V}\mathrm{ar}(x_t) = \sigma_t^2 \mathrm{Id}$.

- **Variance preserving (VP) choice:** $\alpha_t + \beta_t = 1$.

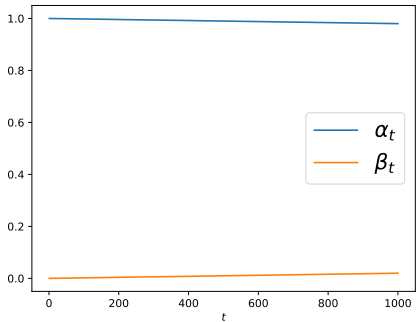
  By induction, we can show that this implies $\overline{\alpha}_t + \overline{\beta}_t = 1$.

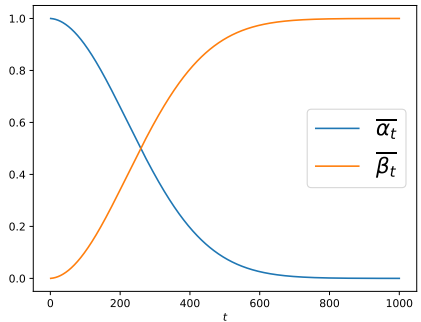
  Therefore, if $\mathbb{V}\mathrm{ar}(x_0) = \mathrm{Id}$, then $\quad \forall t \geq 0, \quad \mathbb{V}\mathrm{ar}(x_t) = \mathrm{Id}$.

Practical Choice of (β_t)

In the implementation of DDPM, (β_t) is chosen small, with linear increase

$$\beta_t = (1 - \tfrac{t-1}{T-1}) \times 10^{-4} + \tfrac{t-1}{T-1} \times 0.02.$$





Learning DDPM with an ELBO

The training of the DDPM process $(\tilde{x}_t)_{t\in[0:T]}$ is inspired by maximum likelihood estimation:

Proposition

We have the following evidence lower bound (ELBO):

$$\mathbb{E}[\log p_\theta(x_0)] \geq -\mathbb{E}[\mathrm{KL}(q(x_T|x_0), p_\theta(x_T))] + \mathbb{E}[\log p_\theta(x_0|x_1)] - \mathbb{E}\left[\sum_{t=2}^{T} \mathrm{KL}(q(x_{t-1}|x_0, x_t), p_\theta(x_{t-1}|x_t))\right],$$

where the expectation is taken w.r.t. $q(x_0, \ldots, x_T)$ *(true forward process).*

Learning DDPM with an ELBO

The training of the DDPM process $(\tilde{x}_t)_{t\in[0:T]}$ is inspired by maximum likelihood estimation:

Proposition

We have the following evidence lower bound (ELBO):

$$\mathbb{E}[\log p_\theta(x_0)] \geq -\mathbb{E}[\mathrm{KL}(q(x_T|x_0), p_\theta(x_T))] + \mathbb{E}[\log p_\theta(x_0|x_1)] - \mathbb{E}\left[\sum_{t=2}^{T} \mathrm{KL}(q(x_{t-1}|x_0, x_t), p_\theta(x_{t-1}|x_t))\right],$$

where the expectation is taken w.r.t. $q(x_0, \dots, x_T)$ (true forward process).

Since $q(x_{t-1}|x_0, x_t), p_\theta(x_{t-1}|x_t)$ are Gaussian distributions, the last term can be computed with

$$L_t(x_0, x_t) = \mathrm{KL}(q(x_{t-1}|x_0, x_t), p_\theta(x_{t-1}|x_t)) = \frac{1}{\beta_t}\|\mu_\theta(x_t, t) - \tilde{\mu}_t(x_t, x_0)\|^2 + cst,$$

where

$$\tilde{\mu}_t(x_t, x_0) = \frac{\sqrt{\overline{\alpha_{t-1}}}\beta_t x_0 + \sqrt{\alpha_t}\overline{\beta_{t-1}} x_t}{\overline{\beta}_t}.$$

Parameterization with Noise Prediction

Consider the following parameterization of the backward transition:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{\bar{\beta}_t}} \varepsilon_\theta(x_t, t) \right).$$

Then the L_t part of the ELBO can be written as $\mathbb{E}[L_t(x_0, x_t)] = \frac{\beta_t}{\alpha_t \bar{\beta}_t} \mathbb{E}[\|\varepsilon_\theta(x_t, t) - \epsilon_t\|^2] + cst.$

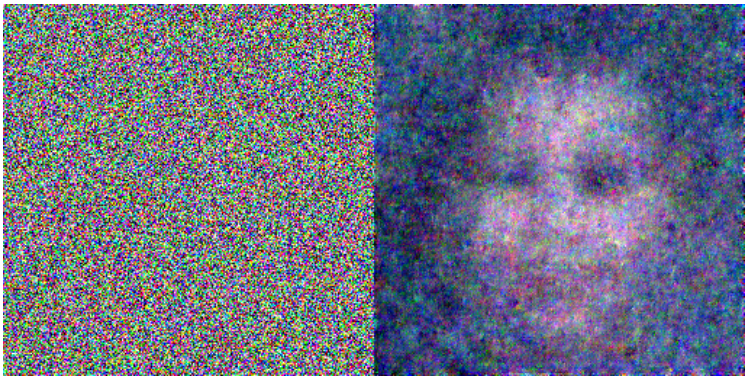
In other words, $\varepsilon_\theta(x_t, t)$ is trained to predict the noise added from x_0 to x_t.

With this parameterization, backward sampling rewrites as $\tilde{x}_T \sim \mathcal{N}(0, \mathrm{Id})$

$$\tilde{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\tilde{x}_t - \frac{\beta_t}{\sqrt{\bar{\beta}_t}} \varepsilon_\theta(\tilde{x}_t, t) \right) + \sqrt{\tilde{\beta}_t} \tilde{z}_t \tag{3}$$

where $\tilde{z}_1, \ldots, \tilde{z}_T$ are independent $\mathcal{N}(0, \mathrm{Id})$ (actually $\tilde{z}_1 = 0$ in practice).

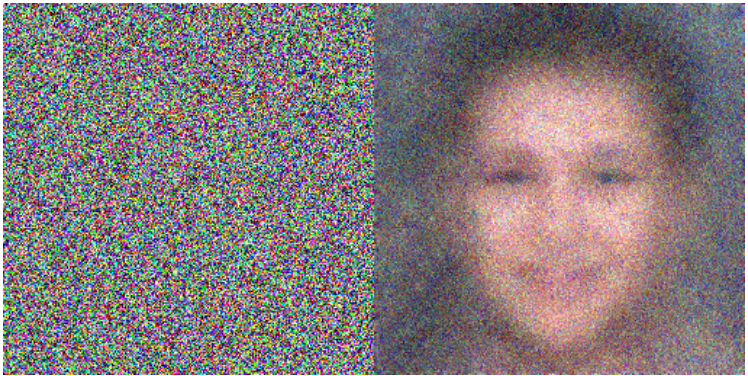
Example of DDPM Sampling



$\tilde{x}_t$ Prediction of x_0

$t = 999$

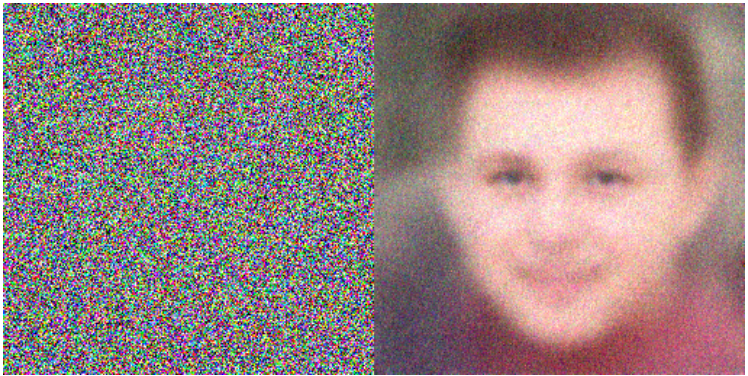
Example of DDPM Sampling



$\tilde{x}_t$ Prediction of x_0

$t = 900$

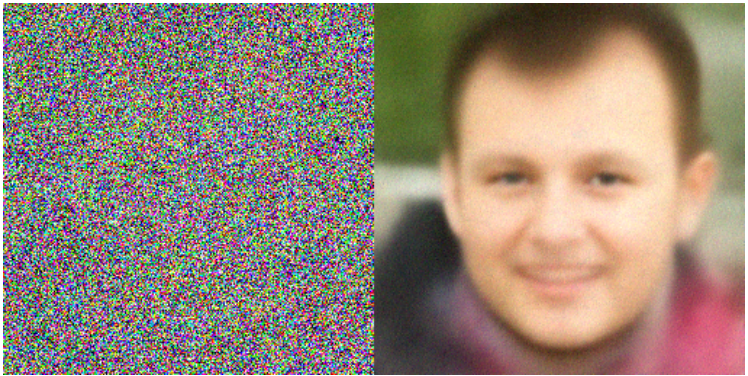
Example of DDPM Sampling



$\tilde{x}_t$ Prediction of x_0

$t = 800$

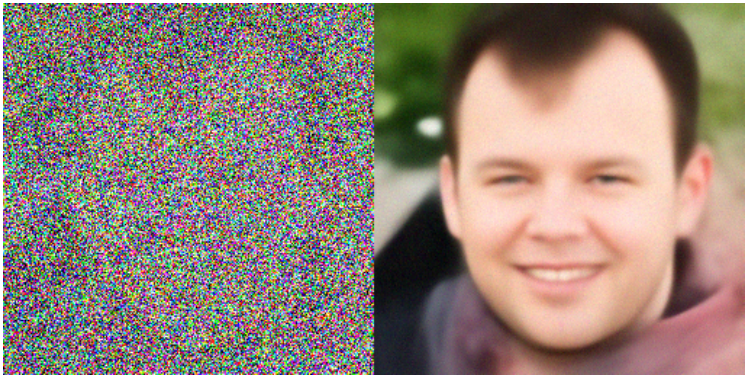
Example of DDPM Sampling



$\tilde{x}_t$ Prediction of x_0

$t = 700$

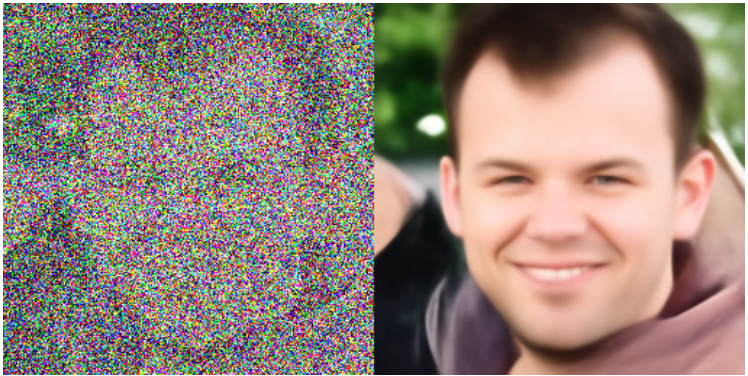
Example of DDPM Sampling



$\tilde{x}_t$ Prediction of x_0

$t = 600$

Example of DDPM Sampling



$\tilde{x}_t$ Prediction of x_0

$t = 500$

Example of DDPM Sampling

$\tilde{x}_t$ Prediction of x_0

$t = 400$

Example of DDPM Sampling

$\tilde{x}_t$ Prediction of x_0

$t = 300$

Example of DDPM Sampling

$\tilde{x}_t$ Prediction of x_0

$t = 200$

Example of DDPM Sampling

$\tilde{x}_t$ Prediction of x_0

$t = 100$

Example of DDPM Sampling

$\tilde{x}_t$ Prediction of x_0

$t = 0$