Denoising Diffusion Probabilistic Models (Ho et al., 2020)

Presented by: Yue Yu[†]

†: Statistics Club, Indiana University

Tentative Schedule of Diffusion Model Series

- **10/10**: Overview of generative AI models in Computer Vision, including GANs (Goodfellow et al., 2014), VAEs (Kingma, 2013) and diffusion models (Sohl-Dickstein et al., 2015), and detailed introduction to GANs;
- **10/17**: Detailed introduction to Autoencoders;
- **10/24: DDPM (Ho et al., 2020), the cornerstone paper about diffusion models, which enables diffusion models to produce high-quality, realistic samples and to be competitive with generative models like GANs and VAEs;**

Tentative Schedule of Diffusion Model Series Cont'

- **11/7**: DDIM (Song et al., 2020), which presents a method to speed up the sampling process in diffusion models without sacrificing much in terms of sample quality, and guided diffusion models (Dhariwal and Nichol, 2021), (Ho and Salimans, 2022), which enhances the flexibility and controllability of generated samples, and are often used in tasks requiring conditional generation, such as text-to-image synthesis;

- **11/21**: Recent years' improvements and applications of diffusion models from different aspects, e.g., Latent Diffusion Models (Stable Diffusion, Rombach et al. (2022)) and its subsequent works like conditional control to text-to-image diffusion (Zhang et al., 2023) and Stable Diffusion XL (Podell et al., 2023).

Today's Presentation Outline

- Overview of Diffusion Models;
- Forward Diffusion Process;
- Reverse Diffusion Process;
- Model Architecture: U-Nets.

Diffusion Models

Diffusion models are inspired by non-equilibrium thermodynamics. They define a Markov chain of diffusion steps to slowly add random noise to data and then learn to reverse the diffusion process to construct desired data samples from the noise. Unlike VAE, diffusion models are learned with a fixed procedure and the latent variable has high dimensionality (same as the original data). It consists of two processes:

- *Forward* process $q(\mathbf{x}_{0:T})$: The model gradually add random Gaussian noise to the original data input $\mathbf{x}_0$, until the input turns into a Gaussian noise $\mathbf{x}_T$;
- *Reverse* process $p(\mathbf{x}_{0:T})$: The model gradually denoise the Gaussian noise $\mathbf{x}_T$, until the output becomes a sample $\mathbf{x}_0^{'}$ that is similar to our input $\mathbf{x}_0$.

Diffusion Model for 2D Swiss Roll Data

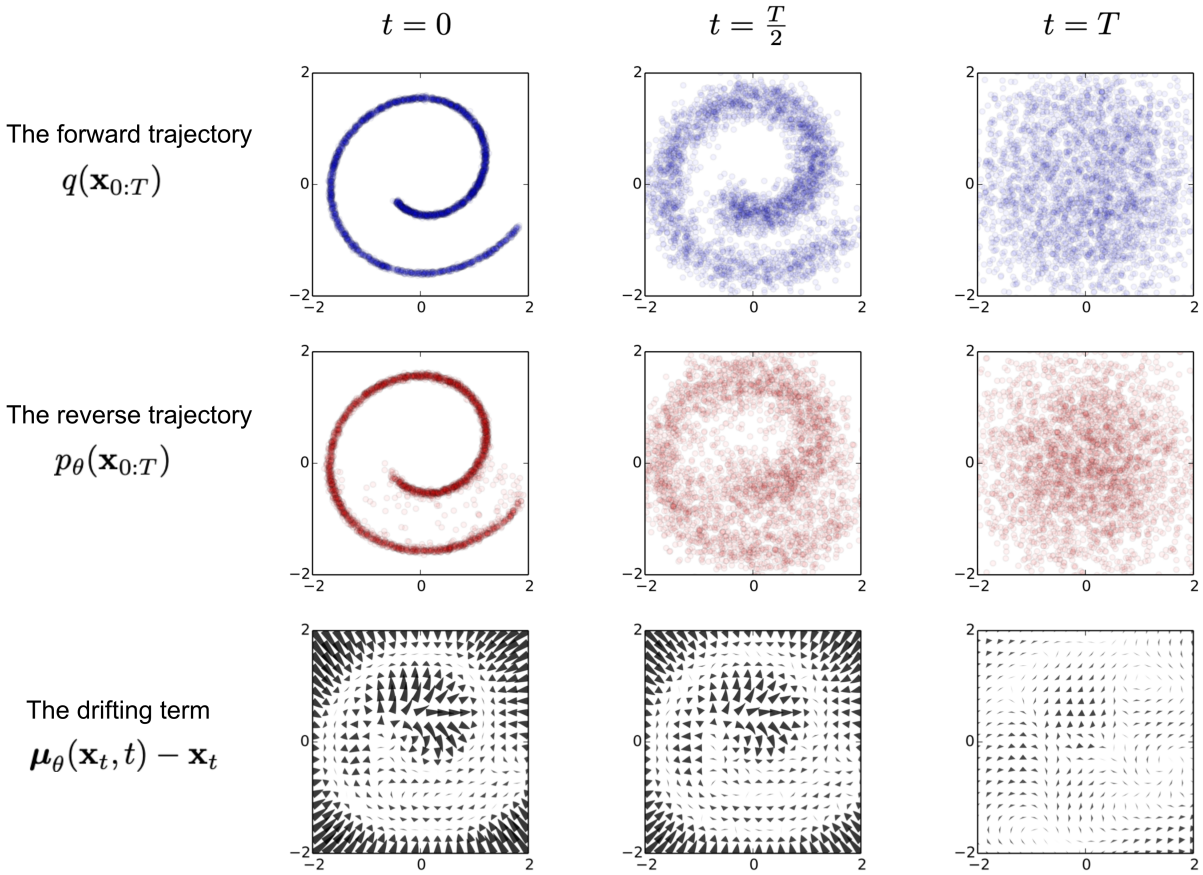

Figure: An example of training a diffusion model for modeling a 2D swiss roll data (Sohl-Dickstein et al., 2015).

Notation (1/2)

- $\mathbf{x}_0$: Original data sample (e.g., clean image).
- $\mathbf{x}_t$: Noisy data sample at diffusion step t, where $t \in \{0, 1, \ldots, T\}$.
- T: Total number of diffusion steps.
- ϵ: Gaussian noise added to the data at each diffusion step, $\epsilon \sim \mathcal{N}(0, \mathbf{I})$.
- α_t: Scaling factor for the forward diffusion process, where $\alpha_t = 1 - \beta_t$.
- $\bar{\alpha}_t$: Cumulative product of α_i from 1 to t, $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$.
- β_t: Variance parameter for the forward diffusion process, controlling the amount of noise added at each step.

Notation (2/2)

- $q(\mathbf{x}_t|\mathbf{x}_{t-1})$: Forward process distribution of adding the noise.
- $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$: Learned reverse process distribution parameterized by θ.
- $\Sigma_\theta(\mathbf{x}_t, t)$: Predicted variance for the reverse process at step t.
- $\mu_\theta(\mathbf{x}_t, t)$: Predicted mean for the reverse process at step t.
- $\tilde{\beta}_t$: Adjusted variance parameter for the reverse process, $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t$.
- L_{VLB}: Variational lower bound loss used for training the diffusion model.
- L_t^{simple}: Simplified loss term for the training objective at step t.
- $\epsilon_\theta(\mathbf{x}_t, t)$: Neural network prediction of the noise added at step t.

Forward Diffusion Process

Given a data point sampled from a real data distribution $\mathbf{x}_0 \sim q(\mathbf{x})$, let us define a *forward diffusion process* in which we add a small amount of Gaussian noise to the sample in T steps, producing a sequence of noisy samples $\mathbf{x}_1, \ldots, \mathbf{x}_T$. The step sizes are controlled by a variance schedule $\{\beta_t \in (0,1)\}_{t=1}^{T}$.

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

The data sample $\mathbf{x}_0$ gradually loses its distinguishable features as the step t becomes larger. Eventually, when $T \to \infty$, $\mathbf{x}_T$ is equivalent to an isotropic Gaussian distribution.

Sampling with Reparameterization Trick

A nice property of the above process is that we can sample $\mathbf{x}_t$ at any arbitrary time step t in a closed form using the *reparameterization trick*. Let $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$:

$$
\begin{aligned}
\mathbf{x}_t &= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\epsilon_{t-1} \quad ; \text{where } \epsilon_{t-1}, \epsilon_{t-2}, \cdots \sim \mathcal{N}(0, \mathbf{I}) \\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\bar{\epsilon}_{t-2} \quad ; \text{where } \bar{\epsilon}_{t-2} \text{ merges two Gaussians} \\
&= \dots \\
&= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon
\end{aligned}
$$

Therefore,

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

Merged Gaussians

- Recall that when we merge two Gaussians with different variances, $\mathcal{N}(0, \sigma_1^2\mathbf{I})$ and $\mathcal{N}(0, \sigma_2^2\mathbf{I})$, the new distribution is $\mathcal{N}(0, (\sigma_1^2 + \sigma_2^2)\mathbf{I})$.
- Here the merged standard deviation is:

$$\sqrt{(1-\alpha_t) + \alpha_t(1-\alpha_{t-1})} = \sqrt{1-\alpha_t\alpha_{t-1}}$$

- Usually, we can afford a larger update step when the sample gets noisier, so $\beta_1 < \beta_2 < \cdots < \beta_T$ and therefore $\bar{\alpha}_1 > \cdots > \bar{\alpha}_T$.

Reverse Diffusion Process

If we can reverse the above process and sample from $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$, we will be able to recreate the true sample from a Gaussian noise input, $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$. Note that if β_t is small enough, $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ will also be Gaussian. Unfortunately, we cannot easily estimate $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ because it needs to use the entire dataset, and therefore we need to learn a model p_θ to approximate these conditional probabilities in order to run the *reverse diffusion process*.

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$$

Tractability of the Reverse Conditional Probability

It is noteworthy that the reverse conditional probability is tractable when conditioned on $\mathbf{x}_0$:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I})$$

Using Bayes' rule, we have:

$$\begin{aligned} q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) &= \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0) q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} \\ &\propto \dots \\ &= \exp\left(-\frac{1}{2}\left(\left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right)\mathbf{x}_{t-1}^2 \right.\right. \\ &\quad \left.\left. - \left(\frac{2\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t + \frac{2\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}\mathbf{x}_0\right)\mathbf{x}_{t-1} + C(\mathbf{x}_t, \mathbf{x}_0)\right)\right), \end{aligned}$$

where $C(\mathbf{x}_t, \mathbf{x}_0)$ is some function not involving $\mathbf{x}_{t-1}$ and details are omitted.

Parameterizing the Mean

Following the standard Gaussian density function, the mean of $\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0)$ can be parameterized as follows (recall that $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{T} \alpha_i$):

$$
\begin{aligned}
\tilde{\beta}_t &= 1 \Big/ \left(\frac{\alpha_t}{\beta_t} + \frac{1}{1 - \bar{\alpha}_{t-1}} \right) \\
&= 1 \Big/ \left(\frac{\alpha_t - \bar{\alpha}_{t-1} + \beta_t}{\beta_t (1 - \bar{\alpha}_{t-1})} \right) \\
&= \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \cdot \beta_t
\end{aligned}
$$

Parameterizing the Mean (Cont')

$$
\begin{aligned}
\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) &= \left(\frac{\sqrt{\alpha_t}}{\beta_t} \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1 - \bar{\alpha}_{t-1}} \mathbf{x}_0 \right) \Bigg/ \left(\frac{\alpha_t}{\beta_t} + \frac{1}{1 - \bar{\alpha}_{t-1}} \right) \\
&= \left(\frac{\sqrt{\alpha_t}}{\beta_t} \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1 - \bar{\alpha}_{t-1}} \mathbf{x}_0 \right) \Bigg/ \left(\frac{\alpha_t}{\beta_t} + \frac{1}{1 - \bar{\alpha}_{t-1}} \right) \\
&= \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} \mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}} \beta_t}{1 - \bar{\alpha}_t} \mathbf{x}_0
\end{aligned}
$$

Simplifying the Expression for $\tilde{\mu}_t$

Notice that we can represent $\mathbf{x}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\epsilon_t\right)$ and plug it into the above equation to obtain:

$$\begin{aligned}\tilde{\mu}_t &= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\frac{1}{\sqrt{\bar{\alpha}_t}}\left(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\epsilon_t\right) \\ &= \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_t\right)\end{aligned}$$

Markov Chain in Diffusion Models

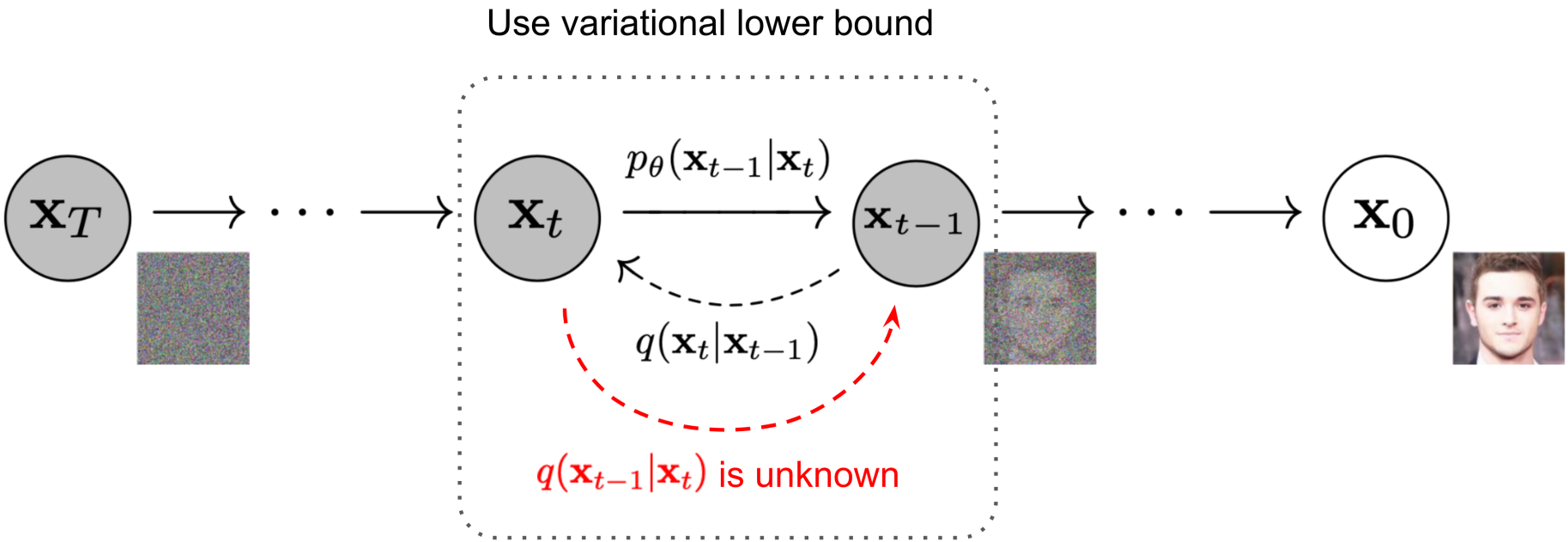

Figure: The Markov chain of forward (reverse) diffusion process of generating a sample by slowly adding (removing) noise (Ho et al., 2020).

Optimizing the Negative Log-Likelihood

We can see that such a setup is very similar to **VAE** and thus we can use the **variational lower bound/ELBO** to optimize the negative log-likelihood:

$$
\begin{aligned}
-\log p_\theta(\mathbf{x}_0) &\leq -\log p_\theta(\mathbf{x}_0) + D_{\mathsf{KL}}(q(\mathbf{x}_{1:T}|\mathbf{x}_0)||p_\theta(\mathbf{x}_{1:T}|\mathbf{x}_0)) \\
&= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_{\mathbf{x}_{1:T}\sim q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})/p_\theta(\mathbf{x}_0)}\right] \\
&= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_q\left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} + \log p_\theta(\mathbf{x}_0)\right] \\
&= \mathbb{E}_q\left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\right] \\
&:= L_{\mathsf{VLB}}.
\end{aligned}
$$

Rewriting the Variational Lower Bound

To convert each term in the equation to be analytically computable, the objective can be further rewritten as a combination of several KL-divergence and entropy terms (Sohl-Dickstein et al., 2015):

$$
\begin{aligned}
L_{\text{VLB}} &= \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\right] \\
&= \dots \\
&= \mathbb{E}_q\left[D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0)\,||\,p_\theta(\mathbf{x}_T))\right]_{L_T} \\
&\quad + \sum_{t=2}^{T} \mathbb{E}_q\left[D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)\,||\,p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))\right]_{L_{t-1}} \\
&\quad - \mathbb{E}_q\left[\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\right]_{L_0}.
\end{aligned}
$$

Labeling the Components of the Variational Lower Bound

Let's label each component in the variational lower bound loss separately:

$$
\begin{aligned}
L_{\text{VLB}} &= L_T + L_{T-1} + \cdots + L_0 \\
\text{where} \quad L_T &= D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \,||\, p_\theta(\mathbf{x}_T)) \\
L_t &= D_{\text{KL}}(q(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{x}_0) \,||\, p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})) \quad \text{for } 1 \leq t \leq T-1 \\
L_0 &= -\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)
\end{aligned}
$$

Every KL term in L_{VLB} (except for L_0) compares two Gaussian distributions and therefore they can be computed in closed form. L_T is constant and can be ignored during training because q has no learnable parameters and $\mathbf{x}_T$ is a Gaussian noise. Ho et al. (2020) models L_0 using a separate discrete decoder derived from $\mathcal{N}(\mathbf{x}_0; \mu_\theta(\mathbf{x}_1, 1), \Sigma_\theta(\mathbf{x}_1, 1))$.

Parameterization of L_t for Training Loss

Recall that we need to learn a neural network to approximate the conditioned probability distributions in the reverse diffusion process, $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$. We would like to train μ_θ to predict $\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_t\right)$. Because $\mathbf{x}_t$ is available as input at training time, we can reparameterize the Gaussian noise term instead to make it predict ϵ_t from the input $\mathbf{x}_t$ at time step t:

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right)$$

$$\text{Thus, } \mathbf{x}_{t-1} \sim \mathcal{N}\left(\mathbf{x}_{t-1}; \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right), \Sigma_\theta(\mathbf{x}_t, t)\right).$$

Parameterizing L_t to Minimize the Difference from $\tilde{\mu}$

The loss term L_t is parameterized to minimize the difference from $\tilde{\mu}$:

$$
\begin{aligned}
L_t &= \mathbb{E}_{\mathbf{x}_0,\epsilon}\left[\frac{1}{2\|\Sigma_\theta(\mathbf{x}_t,t)\|_2^2}\left\|\tilde{\mu}_t(\mathbf{x}_t,\mathbf{x}_0)-\mu_\theta(\mathbf{x}_t,t)\right\|^2\right] \\
&= \dots \\
&= \mathbb{E}_{\mathbf{x}_0,\epsilon}\left[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t)\|\Sigma_\theta\|_2^2}\left\|\epsilon_t-\epsilon_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0+\sqrt{1-\bar{\alpha}_t}\epsilon_t,t\right)\right\|^2\right].
\end{aligned}
$$

Simplification of Training Objective

Empirically, Ho et al. (2020) found that training the diffusion model works better with a simplified objective that ignores the weighting term:

$$
\begin{aligned}
L_t^{\text{simple}} &= \mathbb{E}_{t\sim[1,T],\mathbf{x}_0,\epsilon_t}\left[\|\epsilon_t - \epsilon_\theta(\mathbf{x}_t, t)\|^2\right] \\
&= \mathbb{E}_{t\sim[1,T],\mathbf{x}_0,\epsilon_t}\left[\left\|\epsilon_t - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_t, t\right)\right\|^2\right]
\end{aligned}
$$

The final simplified objective is:

$$
L_{\text{simple}} = L_t^{\text{simple}} + C
$$

where C is a constant not depending on θ.

Training and Sampling Algorithms in DDPM

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** $\mathbf{x}_0$

Figure: The training and sampling algorithms in DDPM (Ho et al., 2020).

Parameterization of β_t

- The forward variances are set to be a sequence of linearly increasing constants in Ho et al. (2020), from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$. They are relatively small compared to the normalized image pixel values between $[-1, 1]$.
- Nichol and Dhariwal (2021) proposed several improvement techniques to help diffusion models to obtain lower NLL. One of the improvements is to use a cosine-based variance schedule. The choice of the scheduling function can be arbitrary, as long as it provides a near-linear drop in the middle of the training process and subtle changes around $t = 0$ and $t = T$.

Comparison of Linear and Cosine-Based Scheduling of β_t

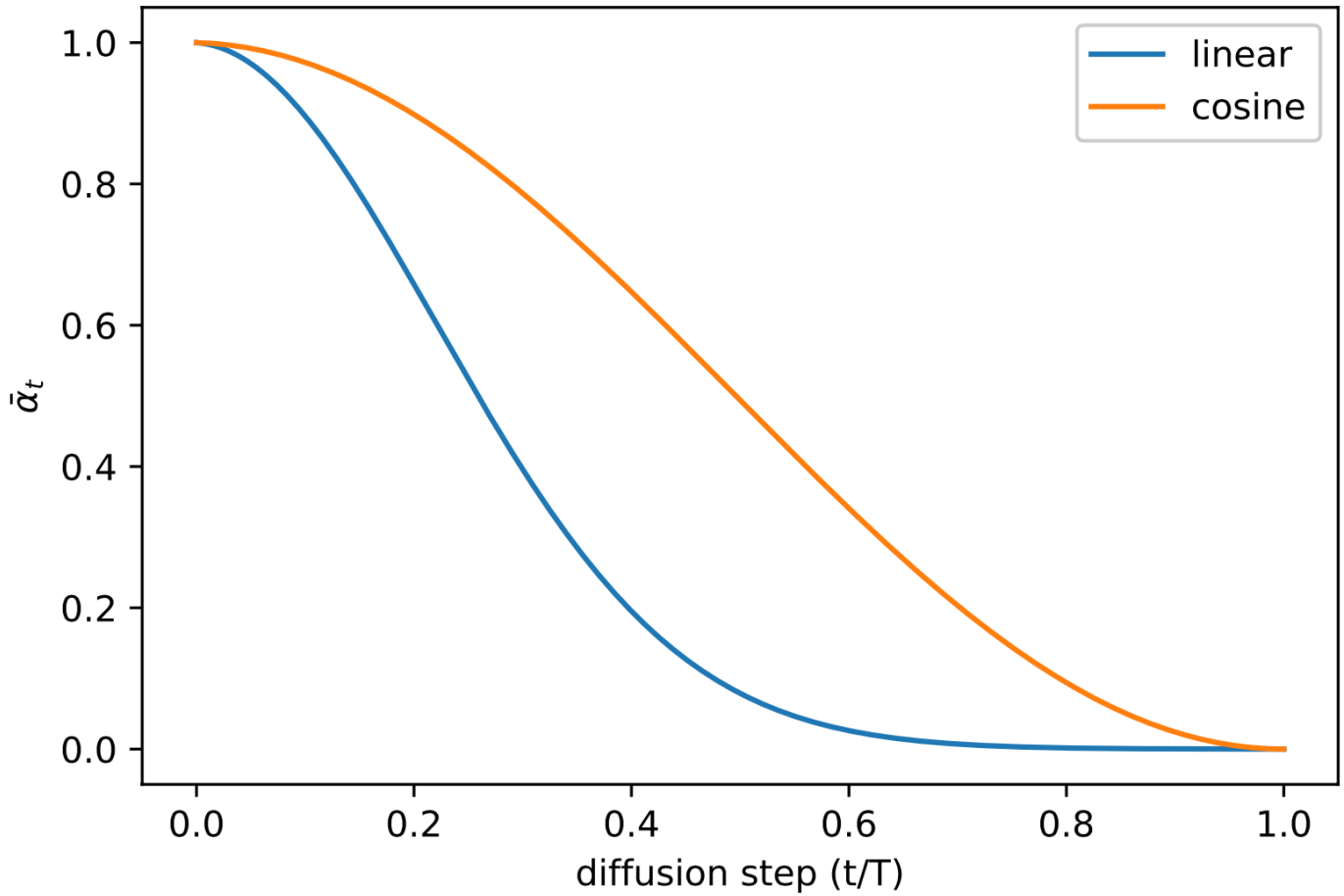

Figure: Comparison of linear and cosine-based scheduling of β_t during training (Nichol and Dhariwal, 2021).

Parameterization of Reverse Process Variance Σ_θ-DDPM

- Ho et al. (2020) chose to fix β_t as constants instead of making them learnable and set $\Sigma_\theta(\mathbf{x}_t, t) = \sigma_t^2 \mathbf{I}$, where σ_t is not learned but set to β_t or $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t$. They found that learning a diagonal variance Σ_θ leads to unstable training and poorer sample quality.

Parameterization of Reverse Process Variance Σ_θ-Improved DDPM

- Nichol and Dhariwal (2021) proposed to learn $\Sigma_\theta(\mathbf{x}_t, t)$ as an interpolation between β_t and $\tilde{\beta}_t$ by predicting a mixing vector $\mathbf{v}$:

$$\Sigma_\theta(\mathbf{x}_t, t) = \exp\left(\mathbf{v} \log \beta_t + (1 - \mathbf{v}) \log \tilde{\beta}_t\right)$$

- However, the simple objective L_{simple} does not depend on Σ_θ. To add the dependency, they constructed a hybrid objective $L_{\text{hybrid}} = L_{\text{simple}} + \lambda L_{\text{VLB}}$, where $\lambda = 0.001$ is small and stops the gradient on μ_θ in the L_{VLB} term such that L_{VLB} only guides the learning of Σ_θ. Empirically, they observed that L_{VLB} is challenging to optimize likely due to noisy gradients, so they proposed using a time-averaging smoothed version of L_{VLB} with importance sampling.

Neural Network Architecture: U-Net

U-Net (Ronneberger et al., 2015) consists of a downsampling stack and an upsampling stack, and is chosen to be the neural network architecture in Ho et al. (2020).

- **Downsampling:** Each step consists of the repeated application of two 3×3 convolutions (unpadded convolutions), each followed by a ReLU and a 2×2 max pooling with stride 2. At each downsampling step, the number of feature channels is doubled.
- **Upsampling:** Each step consists of an upsampling of the feature map followed by a 2×2 convolution and each halves the number of feature channels.
- **Shortcuts:** Shortcut connections result in a concatenation with the corresponding layers of the downsampling stack and provide the essential high-resolution features to the upsampling process.

U-Net Architecture

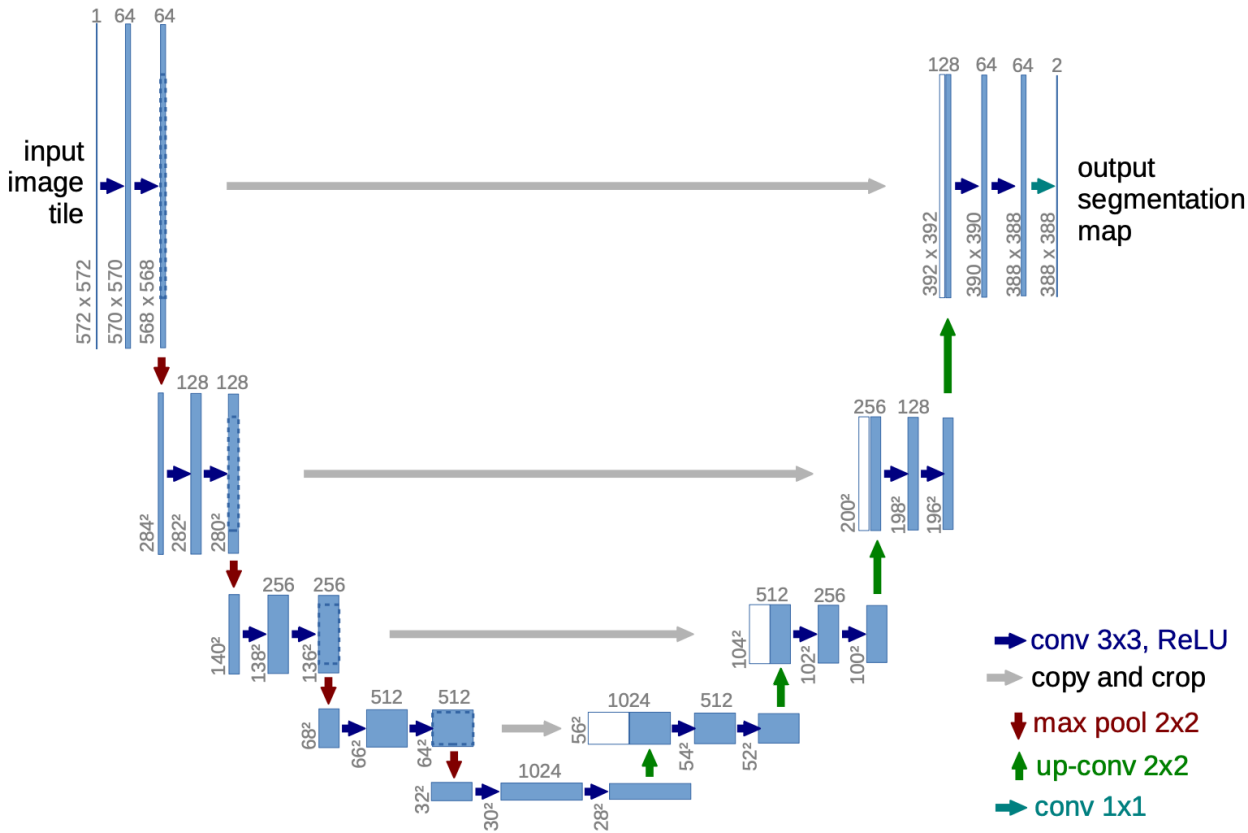

Figure: The U-Net architecture. Each blue square is a feature map with the number of channels labeled on top and the height $\times$ width dimension labeled on the left bottom side. The gray arrows mark the shortcut connections (Ronneberger et al., 2015).

Why Using U-Net in Diffusion Models?

- Diffusion models are highly related to the idea of **stacked denoising autoencoders** (Kumar et al., 2014).
- U-Net-like architectures are commonly used for autoencoders on images.
- The bottleneck and skip-connections in U-Nets help with denoising by providing representations at different granularities.
- U-Nets are well-suited for image segmentation tasks, where the segmentation output needs consistent areas representing objects in the image.
- The process effectively turns a noisy segmentation mask (original image input) into a mask with reduced noise (segmented output).

Next Talk: Improvement on Sampling Speed: DDIM

- **Denoising Diffusion Implicit Models (DDIM)** (Song et al., 2020) aim to accelerate the sampling process in diffusion models.
- Instead of using a fixed number of diffusion steps, DDIM formulates the process as a **non-Markovian transformation**, allowing flexible intermediate steps.
- Sampling can be accelerated by skipping diffusion steps, enabling high-quality samples with significantly fewer iterations.
- Empirically, DDIM achieves comparable sample quality to DDPM but with **30-50 times fewer sampling steps**.

Next Talk: Guidance in Diffusion Models

- Guided diffusion introduces conditioning mechanisms to control the output of diffusion models.
- **Classifier Guidance** (Dhariwal and Nichol, 2021): Uses a pre-trained classifier's gradient to guide the denoising process, encouraging samples toward desired classes.
- **Classifier-Free Guidance** (Ho and Salimans, 2022): Eliminates the need for a classifier by directly training a model with and without conditioning signals.
- **Strength**: Allows fine control over the generation quality and attributes, balancing between diversity and fidelity.

Questions?

Thank you!

References I

P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

J. Ho and T. Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

D. P. Kingma. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

References II

V. Kumar, G. C. Nandi, and R. Kala. Static hand gesture recognition using stacked denoising sparse autoencoders. In *2014 seventh international conference on contemporary computing (IC3)*, pages 99–104. IEEE, 2014.

A. Q. Nichol and P. Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pages 8162–8171. PMLR, 2021.

D. Podell, Z. English, K. Lacey, A. Blattmann, T. Dockhorn, J. Müller, J. Penna, and R. Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023.

R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

References III

O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

L. Zhang, A. Rao, and M. Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023.