# Deep Generative Models

## Chapter 6: Generation by Diffusion Process

Ali Bereyhi

ali.bereyhi@utoronto.ca

Department of Electrical and Computer Engineering

University of Toronto

Summer 2025

# Diffusion Probabilistic Model

## Diffusion Probabilistic Models (DPMs)

*DPMs learn the reverse diffusion process by considering a Gaussian conditional distribution, i.e., they assume*

$$P_{\mathbf{w}}\left(x_{t-1}|x_t\right) \equiv \mathcal{N}\left(\mu_{\mathbf{w}}\left(x_t, t\right), \Sigma_{\mathbf{w}}\left(x_t, t\right)\right)$$

*for some computational models $\mu_{\mathbf{w}}$ and $\Sigma_{\mathbf{w}}$*

+ *Do DPMs really use $\Sigma_{\mathbf{w}}$?! You said we only learn the mean!*
- Not really in practice!

*Most practical DPMs only learn the mean value*

$$P_{\mathbf{w}}\left(x_{t-1}|x_t\right) \equiv \mathcal{N}\left(\mu_{\mathbf{w}}\left(x_t, t\right), \sigma_t^2\right)$$

*for some predefined $\sigma_t$*

# Visualizing a DPM: *Additive Gaussian Noise*

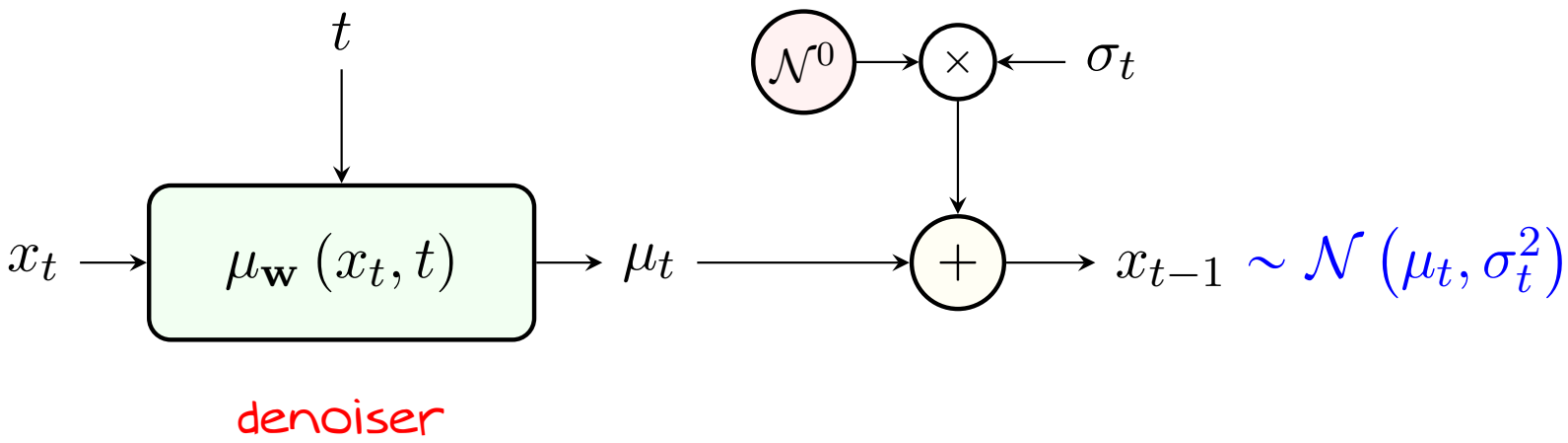
The reverse process *is dual of the forward diffusion: we can see*

$$P_{\mathbf{w}}\left(x_{t-1}|x_t\right) \equiv \mathcal{N}\left(\mu_{\mathbf{w}}\left(x_t, t\right), \sigma_t^2\right)$$

*as a standard Gaussian process with additive noise*

$$x_{t-1} = \mu_{\mathbf{w}}\left(x_t, t\right) + \sigma_t \overleftarrow{\varepsilon}_t$$

*This is very easy to implement by a denoising network!*



# Generation by DPMs

+ *How do DPMs generate a new data sample?*
– Just move on the reverse process back in time

*We can sample Gaussian noise and move reversely in time*

$$x_0 \xleftarrow{P_{\mathbf{w},1}} x_1 \xleftarrow{P_{\mathbf{w},2}} \cdots \longleftarrow x_{t-1} \xleftarrow{P_{\mathbf{w},t}} x_t \longleftarrow \cdots \xleftarrow{P_{\mathbf{w},T}} x_T$$

`Sample_DPM(`$\mu_{\mathbf{w}}$`:Denoiser)`

1: *Sample latent* $x_T \sim \mathcal{N}^0$
2: **for** $t = T, \ldots, 1$ **do**
3: *Pass* $x_t$ *and* $t$ *forward to find* $\mu_t = \mu_{\mathbf{w}}(x_t, t)$
4: *Sample* $\varepsilon \sim \mathcal{N}^0$
5: *Denoise as* $x_{t-1} = \mu_t + \sigma_t \varepsilon$
6: **end for**
7: **return** *Sample* $x_0$

# DPM Training: *Naive Training*

+ *How do we train DPMs?!*
– We should only train the denoiser

Following ELBO maximization: *we compute empirical risk as*

$$\begin{aligned}\hat{R}(\mathbf{w}) &= \hat{\mathbb{E}}_{x_0 \sim P_{\text{data}}} \{R(\mathbf{w}|x_0)\} \\ &= \hat{\mathbb{E}}_{x_0 \sim P_{\text{data}}} \left\{ \sum_{t=1}^{T} \frac{1}{\sigma_t^2} \|\mu_{\mathbf{w}}(x_t, t) - \eta_t\|^2 \right\}\end{aligned}$$

*And, just to recall:* $\eta_t$ *is the posterior mean computed as*

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

# DPM Training: *Naive Training*

*We could define* $\omega_t = 1/\sigma_t^2$ *and write the risk function compactly as*

$$\hat{R}(\mathbf{w}) = \hat{\mathbb{E}}_{x_0 \sim P_{\text{data}}} \left\{ \sum_{t=1}^{T} \omega_t \| \mu_{\mathbf{w}}(x_t, t) - \eta_t \|^2 \right\}$$

*We can then write a training loop to minimize this empirical risk*

---

- \+ *What is the order of* $T$*?!*
- – $T$ *is rather large, e.g.,* $T = 1000$*!*
- ◉ *This sounds like a huge forward and backward pass at every sample!!*
- – *Totally agree! But, there is a way around it*

# DPM Training: *Time Sampling*

*There is no harm to scale* $\hat{R}(\mathbf{w})$ *by a constant!*

$$\hat{R}(\mathbf{w}) = \frac{1}{T}\hat{\mathbb{E}}_{x_0 \sim P_{\text{data}}} \left\{ \sum_{t=1}^{T} \omega_t \left\| \mu_{\mathbf{w}}(x_t, t) - \eta_t \right\|^2 \right\}$$

$$= \hat{\mathbb{E}}_{x_0 \sim P_{\text{data}}} \left\{ \frac{1}{T} \sum_{t=1}^{T} \omega_t \left\| \mu_{\mathbf{w}}(x_t, t) - \eta_t \right\|^2 \right\}$$

*Now, assuming that* $t \sim \mathcal{U}_T$ *is uniformly distributed on* $\{1, \ldots, T\}$*, we could say*

$$\hat{R}(\mathbf{w}) = \hat{\mathbb{E}}_{x_0 \sim P_{\text{data}}} \left\{ \mathbb{E}_{t \sim \mathcal{U}_T} \left\{ \omega_t \left\| \mu_{\mathbf{w}}(x_t, t) - \eta_t \right\|^2 \right\} \right\}$$

# DPM Loss Computation by *Time Sampling*

Similar to averaging over data *we can estimate time expectation by sampling: let the loss function for a single data and time sample be*

$$R\left(\mathbf{w}|x_0, t\right) = \omega_t \|\mu_{\mathbf{w}}\left(x_t, t\right) - \eta_t\|^2$$

*We could say that*

$$\mathbb{E}_{t\sim\mathcal{U}_T}\left\{\omega_t \|\mu_{\mathbf{w}}\left(x_t, t\right) - \eta_t\|^2\right\} \approx \hat{\mathbb{E}}_{t\sim\mathcal{U}_T}\left\{\omega_t \|\mu_{\mathbf{w}}\left(x_t, t\right) - \eta_t\|^2\right\}$$

*where for computing right-hand-side, we sample $t$ uniformly from $\{1, \ldots, T\}$*

*The empirical risk is then estimated as*

$$\hat{R}\left(\mathbf{w}\right) = \hat{\mathbb{E}}_{x_0,t}\left\{\omega_t \|\mu_{\mathbf{w}}\left(x_t, t\right) - \eta_t\|^2\right\}$$

# DPM Loss Computation: *Pseudo Code*

*It might be easier to think of it algorithmically*

`Risk_Computation_DPM(`$\mu_{\mathbf{w}}$`:Denoiser,`$\mathbb{B}$`:data_batch)`

1: **for** *each sample* $j$ *in* $\mathbb{B}$ **do**
2: *Sample* $t^j$ *uniformly from* $\{1, \ldots, T\}$
3: *Sample* $\varepsilon \sim \mathcal{N}^0$ *and compute* $x^j_{tj} = \sqrt{\bar{\alpha}_{tj}}\, x^j_0 + \sqrt{1 - \bar{\alpha}_{tj}}\, \varepsilon$
4: *Compute* $\eta^j_{tj}$ *from* $x^j_0$ *and* $x^j_{tj}$
5: *Pass* $x_{tj}$ *and* $t^j$ *forward to find* $\mu_{tj} = \mu_{\mathbf{w}}\left(x_{tj}, t^j\right)$
6: *Compute sample loss as* $R^j_{tj} = \omega_{tj} \left\| \mu_{tj} - \eta_{tj} \right\|^2$
7: *Backpropagate to compute* $\nabla R^j_{tj}$
8: **end for**
9: *Estimate empirical risk as* $\hat{R} = \texttt{mean}\left(R^1_{t^1}, \ldots R^n_{t^n}\right)$
10: *Estimate batch gradient as* $\nabla \hat{R} = \texttt{Opt\_avg}\left\{R^1_{t^1}, \ldots R^n_{t^n}\right\}$
11: **return** $\hat{R}$ *and* $\nabla \hat{R}$

# Training DPMs: *Efficient Approach*

*We can use this trick to build an efficient training loop*

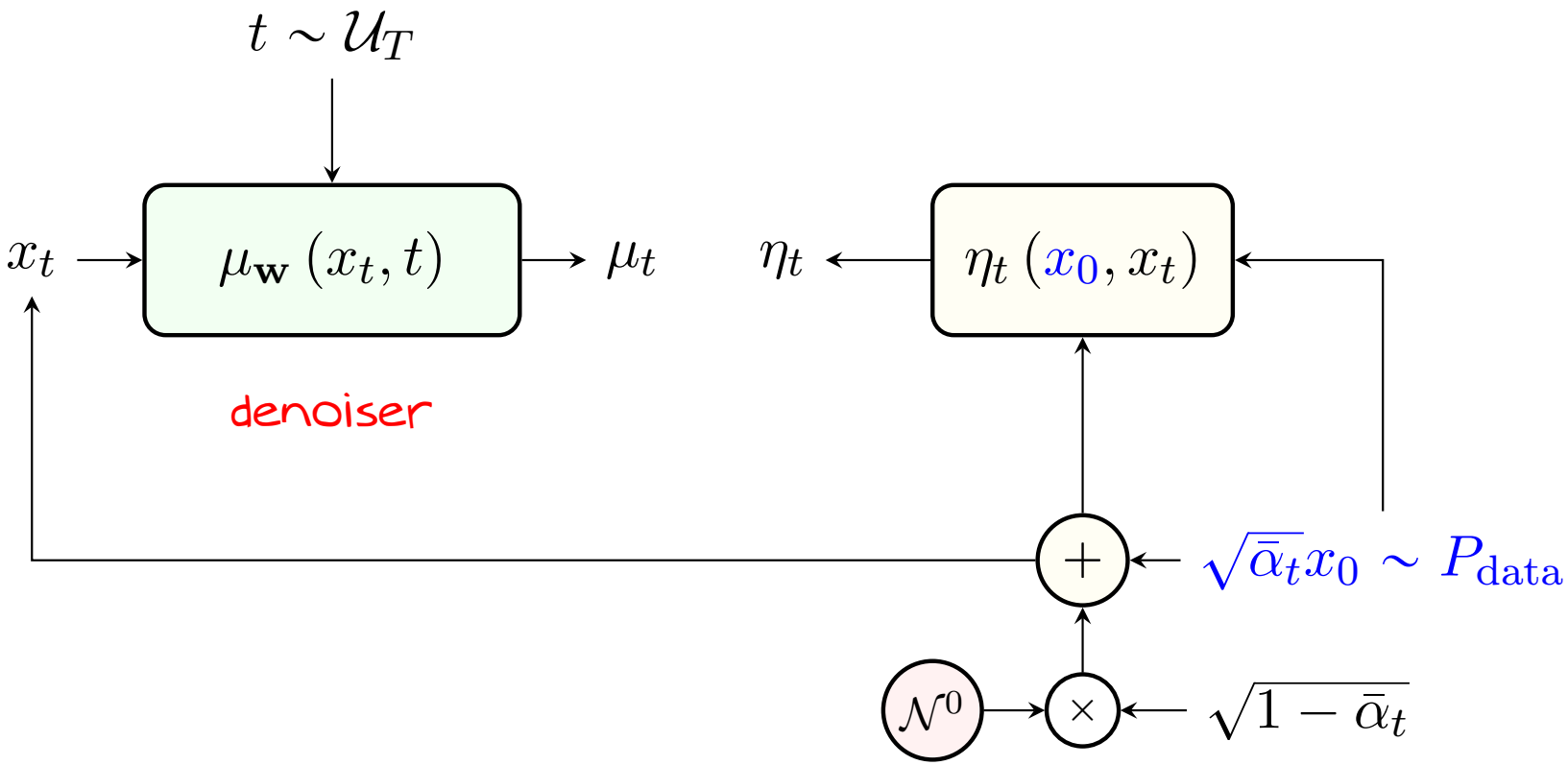
`Training_DPM(`$\mu_{\mathbf{w}}$`:Denoiser,`$\mathbb{D}$`:dataset)`

1: *Initiate* $\mu_{\mathbf{w}}$ *with some weights*
2: **for** *multiple epochs* **do**
3: *Sample a batch* $\mathbb{B}$ *from* $\mathbb{D}$
4: *Compute batch gradient as* $\hat{R}, \nabla\hat{R} \leftarrow$ `Risk_Computation_DPM(`$\mu_{\mathbf{w}}, \mathbb{B}$`)`
5: *Update the weights as* $\mathbf{w} \leftarrow \mathbf{w} - \nabla\hat{R}$
6: **end for**
7: **return** $\mu_{\mathbf{w}}$

*and we could use the trained model to generate a new data sample*

$$x_{\text{new}} \leftarrow \texttt{Sample\_DPM(}\mu_{\mathbf{w}}\texttt{)}$$

# DPM Training Loop: *Visualization*



# Posterior Mean: *Denoising Interpretation*

*Let's take a look at posterior mean that we use for training*

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

*We also recall that*

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon_t$$

*So, we can expand the term as*

$$\begin{aligned}
\eta_t &= \frac{\sqrt{\bar{\alpha}_{t-1}}\,(1-\alpha_t)}{1-\bar{\alpha}_t} x_0 + \frac{\sqrt{\alpha_t}\,(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\left(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon_t\right) \\
&= \underbrace{\frac{\sqrt{\bar{\alpha}_{t-1}}\,(1-\alpha_t) + \sqrt{\bar{\alpha}_t \alpha_t}\,(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}}_{a_t} x_0 + \underbrace{\frac{\sqrt{\alpha_t}\,(1-\bar{\alpha}_{t-1})}{\sqrt{1-\bar{\alpha}_t}}}_{b_t} \varepsilon_t
\end{aligned}$$

# Posterior Mean: *Denoising Interpretation*

*We can use the fact that*

$$\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i \rightsquigarrow \bar{\alpha}_t = \bar{\alpha}_{t-1}\alpha_t$$

*to simplify the coefficients: for* $a_t$ *we have*

$$\begin{aligned} a_t &= \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_t}\left[1-\alpha_t+\alpha_t\left(1-\bar{\alpha}_{t-1}\right)\right] \\ &= \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_t}\left(1-\bar{\alpha}_t\right) \\ &= \sqrt{\bar{\alpha}_{t-1}} = \frac{\sqrt{\bar{\alpha}_t}}{\sqrt{\alpha_t}} \end{aligned}$$

# Posterior Mean: *Denoising Interpretation*

*Similarly for $b_t$ we can write*

$$
\begin{aligned}
b_t &= \frac{\sqrt{\alpha_t}\,(1-\bar{\alpha}_{t-1})}{\sqrt{1-\bar{\alpha}_t}} \\
&= \frac{1}{\sqrt{\alpha_t}} \frac{\alpha_t\,(1-\bar{\alpha}_{t-1})}{\sqrt{1-\bar{\alpha}_t}} \\
&= \frac{1}{\sqrt{\alpha_t}} \frac{\alpha_t - \bar{\alpha}_t + 1 - 1}{\sqrt{1-\bar{\alpha}_t}} \\
&= \frac{1}{\sqrt{\alpha_t}} \frac{1-\bar{\alpha}_t - (1-\alpha_t)}{\sqrt{1-\bar{\alpha}_t}} \\
&= \frac{1}{\sqrt{\alpha_t}} \left( \sqrt{1-\bar{\alpha}_t} - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \right)
\end{aligned}
$$

# Posterior Mean: *Denoising Interpretation*

*Putting them together: we can write $\eta_t$ ias*

$$\begin{aligned}
\eta_t &= a_t x_0 + b_t \varepsilon_t \\
&= \frac{\sqrt{\bar{\alpha}_t}}{\sqrt{\alpha_t}} x_0 + \frac{1}{\sqrt{\alpha_t}} \left( \sqrt{1-\bar{\alpha}_t} - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \right) \varepsilon_t \\
&= \frac{1}{\sqrt{\alpha_t}} \left( \underbrace{\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t} \varepsilon_t}_{x_t} - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \varepsilon_t \right) \\
&= \frac{1}{\sqrt{\alpha_t}} \left( x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \varepsilon_t \right) \equiv \text{denoiser of } x_t
\end{aligned}$$

*$\eta_t$ is indeed a denoising of $x_t$ $\rightsquigarrow$ **Bingo!** This is why we called $\mu_{\mathbf{w}}$ denoiser*

# Connection to *Direct Denoiser*

*We could also denoise simply by reversing the forward diffusion*

$$x_t = \sqrt{\alpha_t} x_{t-1} + \sqrt{1-\alpha_t}\tilde{\varepsilon}_t$$

**! Attention**

$\tilde{\varepsilon}_t$ *is noise sample added in step* $t$ *of forward diffusion*

- ↳ *It is different from* $\varepsilon_t$ *which builds the direct link*
- ↳ *They are though correlated!*

*We can directly denoise by reversing this equation as*

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \sqrt{1-\alpha_t}\tilde{\varepsilon}_t\right)$$

# Connection to *Direct Denoiser*

*Indeed, we could think of* $\eta_t$ *as same reverse diffusion when*

*we use an estimator of* $\tilde{\varepsilon}_t$ *computed from* $\varepsilon_t$ *as*

$$\hat{\varepsilon}_t = \sqrt{\frac{1-\alpha_t}{1-\bar{\alpha}_t}}\varepsilon_t$$

*Using this estimator, we can estimate the reverse diffusion as*

$$\begin{aligned}\hat{x}_{t-1} &= \frac{1}{\sqrt{\alpha_t}}\left(x_t - \sqrt{1-\alpha_t}\hat{\varepsilon}_t\right) \\ &= \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_t\right) = \eta_t\end{aligned}$$

# Key Observation: *Sample Loss*

Now, let us look at the training again: *we compute sample loss as*

$$R\left(\mathbf{w}|x_0,t\right)=\omega_t\|\mu_{\mathbf{w}}\left(x_t,t\right)-\eta_t\|^2$$

*and we know that*

$$\eta_t=\hat{x}_{t-1}=\frac{1}{\sqrt{\alpha_t}}\left(x_t-\frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_t\right)$$

## Interesting Observation

*When we train DPM, we have $\varepsilon_t$, since we use it for sample $x_t$ from $x_0$*

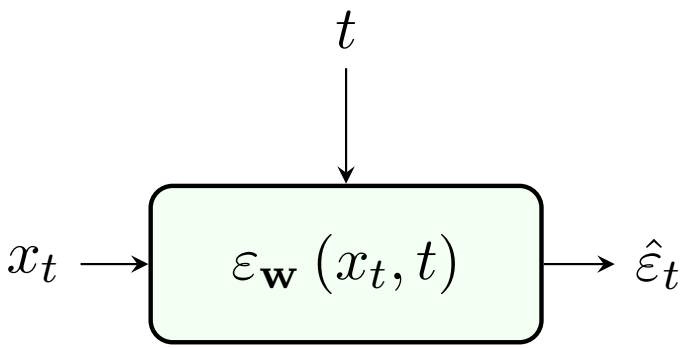
*This observation leads to introduction of denoising DPMs*

# DDPM: *Specific Choice of DPM*

## Denoising DPM (DDPM)

*In DDPMs, we set the model to estimate direct noise at time $t$ and let*

$$\mu_{\mathbf{w}}\left(x_t, t\right) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_{\mathbf{w}}\left(x_t, t\right)\right)$$



# Sample Loss of DDPM

Using DDPM: *the sample loss becomes*

$$
\begin{aligned}
R\left(\mathbf{w}|x_0,t\right) &= \omega_t \left\|\mu_{\mathbf{w}}\left(x_t,t\right)-\eta_t\right\|^2 \\
&= \underbrace{\omega_t \frac{\left(1-\alpha_t\right)^2}{\alpha_t\left(1-\bar{\alpha}_t\right)}}_{\bar{\omega}_t} \left\|\varepsilon_{\mathbf{w}}\left(x_t,t\right)-\varepsilon_t\right\|^2 \\
&= \bar{\omega}_t \left\|\varepsilon_{\mathbf{w}}\left(x_t,t\right)-\varepsilon_t\right\|^2
\end{aligned}
$$

*which we can easily compute!*

## Intuitive Interpretation

*We train a model to estimate direct noise sample from noisy data*

- ↳ *We use the model to estimate direct noise in each time step*
- ↳ *We denoise to reverse the diffusion time step*

# DDPM: *Risk Computation*

*We can build a similar loop to compute empirical risk*

`Risk_Computation_DDPM(`$\varepsilon_{\mathbf{w}}$`:Denoiser,`$\mathbb{B}$`:data_batch)`

1: **for** *each sample* $x_0^j$ *in* $\mathbb{B}$ **do**
2: *Sample* $t^j$ *uniformly from* $\{1, \ldots, T\}$
3: *Sample* $\varepsilon \sim \mathcal{N}^0$ *and compute* $x_{tj}^j = \sqrt{\bar{\alpha}_{tj}} x_0^j + \sqrt{1 - \bar{\alpha}_{tj}}\varepsilon$
4: *Pass* $x_{tj}$ *and* $t^j$ *forward to find* $\hat{\varepsilon}_{tj} = \varepsilon_{\mathbf{w}}\left(x_{tj}, t^j\right)$
5: *Compute sample loss as* $R_{tj}^j = \bar{\omega}_{tj} \left\|\hat{\varepsilon}_{tj} - \varepsilon\right\|^2$
6: *Backpropagate to compute* $\nabla R_{tj}^j$
7: **end for**
8: *Estimate empirical risk as* $\hat{R} = \texttt{mean}\left(R_{t1}^1, \ldots R_{tn}^n\right)$
9: *Estimate gradient as* $\nabla\hat{R} = \texttt{Opt\_avg}\left\{\nabla R_{t1}^1, \ldots \nabla R_{tn}^n\right\}$
10: **return** $\hat{R}$ *and* $\nabla\hat{R}$

# DDPM: *Training Loop*

`Training_DDPM(`$\varepsilon_{\mathbf{w}}$`:Denoiser,`$\mathbb{D}$`:dataset)`

1: *Initiate* $\varepsilon_{\mathbf{w}}$ *with some weights*
2: **for** *multiple epochs* **do**
3: *Sample a batch* $\mathbb{B}$ *from* $\mathbb{D}$
4: *Compute batch gradient as* $\hat{R}, \nabla\hat{R} \leftarrow$ `Risk_Computation_DDPM(`$\varepsilon_{\mathbf{w}}, \mathbb{B}$`)`
5: *Update the weights as* $\mathbf{w} \leftarrow \mathbf{w} - \nabla\hat{R}$
6: **end for**
7: **return** $\varepsilon_{\mathbf{w}}$

# DDPM: *Generation*

`Sample_DDPM(`$\varepsilon_{\mathbf{w}}$`:Denoiser)`

1: *Sample latent* $x_T \sim \mathcal{N}^0$
2: **for** $t = T, \ldots, 1$ **do**
3: *Pass* $x_t$ *and* $t$ *forward to find* $\hat{\varepsilon}_t = \varepsilon_{\mathbf{w}}(x_t, t)$
4: *Denoise the sample* $x_t$ *as*

$$\hat{x}_{t-1} \leftarrow \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\hat{\varepsilon}_t\right)$$

5: *Sample* $\overleftarrow{\varepsilon} \sim \mathcal{N}^0$ *and move reversely as* $x_{t-1} = \hat{x}_{t-1} + \sigma_t \overleftarrow{\varepsilon}$
6: **end for**
7: **return** *Sample* $x_0$

# Few Implementational Notes on DDPM

There are a few notes: *in practice with DDPMs*

- *We typically set $\bar{\omega}_t = 1$ for all $t$*
  - ↳ *This factor does not really make a significant impact*
- *In basic proposal, it is suggested to set $\sigma_t = \sqrt{1 - \alpha_t}$*
  - ↳ *This means that the variance in both directions are the same*
  - ↳ *At time $t = 1$, we set $\sigma_t = 0$, i.e., no noise added to last denoising*
- *The coefficients $\alpha_t$ in general should be scheduled*
  - ↳ *Linear scheduling is one classical option where $\alpha_t = \xi t + \kappa$*
  - ↳ *This might impact considerably on performance*

# Sampling Time of DDPM

The training of DDPM seems solid; *however*,

*the sampling can take long*

*This is because the reverse diffusion is stochastic*

- ↳ *It needs time to converge*

*There are two simple remedies*

- ↳ *Learn more parameters for reverse trajectory, e.g.,* $\sigma_t$
  - ↳ *Simple extension of DDPM with learnable variances*
- ↳ *Make a deterministic sampling mechanism for generation*

# DDIM: *Sampling with Zero Variance*

We can use DDPM *to estimate noise sample*

$$\hat{\varepsilon}_t = \varepsilon_{\mathbf{w}}\left(x_t, t\right)$$

*If this was a very accurate estimate, we could have said*

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon_t \rightsquigarrow \hat{x}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \sqrt{1-\bar{\alpha}_t}\hat{\varepsilon}_t\right)$$

*This is though not accurate!*

## Denoising Diffusion with Implicit Modeling $\equiv$ DDIM

*We may use this estimate to directly estimate previous time sample* $x_{t-1}$

# DDIM: Sampling with Zero Variance

*At time $t$, we get*

$$\hat{x}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \sqrt{1-\bar{\alpha}_t}\varepsilon_{\mathbf{w}}\left(x_t, t\right)\right)$$

*We could then estimate $x_{t-1}$ as*

$$\begin{aligned}\hat{x}_{t-1} &= \sqrt{\bar{\alpha}_{t-1}}\hat{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}}\hat{\varepsilon}_{t-1} \\ &= \sqrt{\bar{\alpha}_{t-1}}\hat{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}}\varepsilon_{\mathbf{w}}\left(x_{t-1}, t-1\right)\end{aligned}$$

*We cannot compute $\varepsilon_{\mathbf{w}}\left(x_{t-1}, t-1\right)$ since we don't have $x_{t-1}$, but*

- *This describes approximate ordinary differential equation for continuous $t$*
- *With no randomness we can say $\varepsilon_{\mathbf{w}}\left(x_{t-1}, t-1\right) \approx \varepsilon_{\mathbf{w}}\left(x_t, t\right)$*

# DDIM: Sampling with Zero Variance

*At time $t$, we get*

$$\hat{x}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}} \left( x_t - \sqrt{1 - \bar{\alpha}_t} \varepsilon_{\mathbf{w}} \left( x_t, t \right) \right)$$

*We could then estimate $x_{t-1}$ as*

$$\hat{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}} \hat{x}_0 + \sqrt{1 - \bar{\alpha}_{t-1}} \varepsilon_{\mathbf{w}} \left( x_t, t \right)$$

**! Attention**

*DDIM only modifies generation! Training is carried out identical to DDPM!*

# DDIM Sampling

`Sample_DDIM(`$\varepsilon_{\mathbf{w}}$`:Denoiser)`

1: *Sample latent* $x_T \sim \mathcal{N}^0$
2: **for** $t = T, \ldots, 1$ **do**
3: *Pass* $x_t$ *and* $t$ *forward to find* $\hat{\varepsilon}_t = \varepsilon_{\mathbf{w}}(x_t, t)$
4: *Find an estimate of data sample as*

$$\hat{x}_0 \leftarrow \frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \sqrt{1-\bar{\alpha}_t}\hat{\varepsilon}_t\right)$$

5: *Denoise as* $x_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}}\hat{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}}\hat{\varepsilon}_t$
6: **end for**
7: **return** *Sample* $x_0$