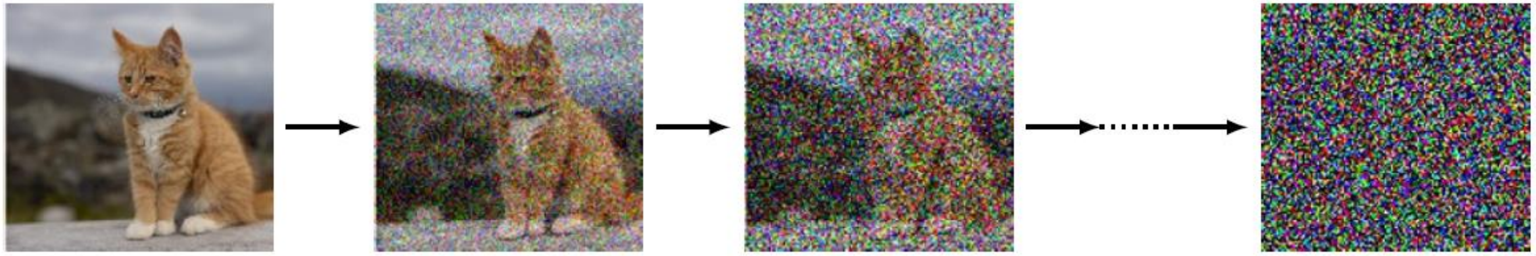
Deep Generative Models (Denoising Diffusion Models)

CS772A: Probabilistic Machine Learning

Piyush Rai

Denoising Diffusion Models

- Based on a forward process (adding noise) and a reverse process (denoising)
- The forward process as is follows $q(\mathbf{z}_1, \ldots, \mathbf{z}_t|\mathbf{x}) = q(\mathbf{z}_1|\mathbf{x}) \prod_{\tau=2}^{t} q(\mathbf{z}_\tau|\mathbf{z}_{\tau-1})$



$$\mathbf{z}_1 = \sqrt{1-\beta_1}\mathbf{x} + \sqrt{\beta_1}\boldsymbol{\epsilon}_1$$

$$q(\mathbf{z}_1|\mathbf{x}) = \mathcal{N}(\mathbf{z}_1|\sqrt{1-\beta_1}\mathbf{x}, \beta_1\mathbf{I})$$

$$\mathbf{z}_t = \sqrt{1-\beta_t}\mathbf{z}_{t-1} + \sqrt{\beta_t}\boldsymbol{\epsilon}_t$$

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

$$\beta_t \in (0,1)$$

Typically set by hand

$$\beta_1 < \beta_2 < \ldots < \beta_T.$$

Called the "diffusion kernel"

$$q(\mathbf{z}_t|\mathbf{x}) = \mathcal{N}(\mathbf{z}_t|\sqrt{\alpha_t}\mathbf{x}, (1-\alpha_t)\mathbf{I})$$

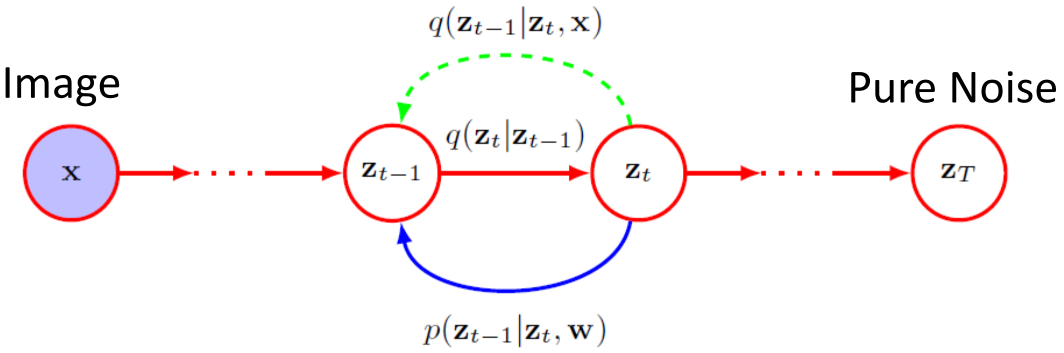
$$\alpha_t = \prod_{\tau=1}^{t}(1-\beta_\tau)$$

As $T \to \infty$

$$q(\mathbf{z}_T|\mathbf{x}) = \mathcal{N}(\mathbf{z}_T|\mathbf{0}, \mathbf{I})$$

Figure source: Bishop & Bishop (2023)

Reversing the Diffusion



- Reversing the diffusion is like denoising
- Can use it to generate data from pure noise
- Reverse process will need $q(z_{t-1}|z_t)$ but computing it is hard in general because

$$q(\mathbf{z}_{t-1}|\mathbf{z}_t) = \frac{q(\mathbf{z}_t|\mathbf{z}_{t-1})q(\mathbf{z}_{t-1})}{q(\mathbf{z}_t)} \quad \text{where} \quad q(\mathbf{z}_{t-1}) = \int q(\mathbf{z}_{t-1}|\mathbf{x})p(\mathbf{x})\,d\mathbf{x}$$

Requires integrating over the data distribution $p(x)$ which is not easy

- Conditioning also on $\boldsymbol{x}$ makes computing the reverse process distribution tractable

Equals $q(z_t|z_{t-1})$

$$q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x}) = \frac{q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x})q(\mathbf{z}_{t-1}|\mathbf{x})}{q(\mathbf{z}_t|\mathbf{x})} = \mathcal{N}\left(\mathbf{z}_{t-1}|\mathbf{m}_t(\mathbf{x}, \mathbf{z}_t), \sigma_t^2\mathbf{I}\right)$$

But this denoising only holds for training images (because it is conditioned on $\boldsymbol{x}$)

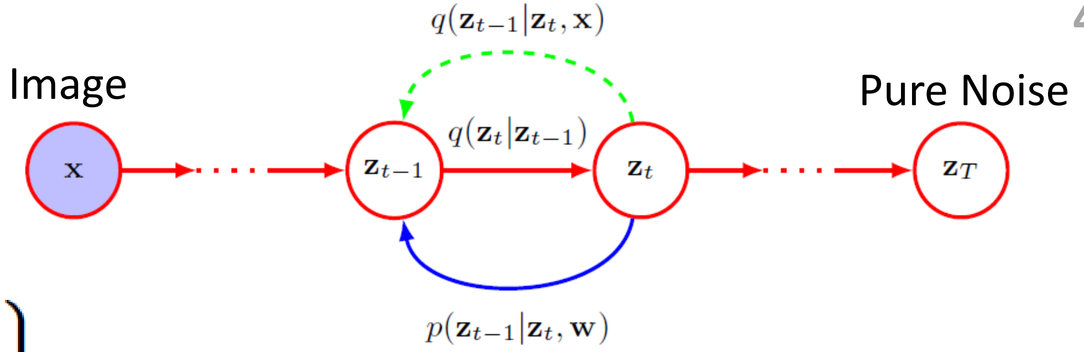
Thus we will also learn "parallel" distributions of the form $p(z_{t-1}|z_t, \boldsymbol{w})$ which approximate $q(z_{t-1}|z_t, \boldsymbol{x})$

If we can now estimate $\boldsymbol{w}$ from training data, we can use $p(z_{t-1}|z_t, \boldsymbol{w})$ to generate new synthetic data starting with pure noise

$$\mathbf{m}_t(\mathbf{x}, \mathbf{z}_t) = \frac{(1-\alpha_{t-1})\sqrt{1-\beta_t}\mathbf{z}_t + \sqrt{\alpha_{t-1}}\beta_t\mathbf{x}}{1-\alpha_t}$$

$$\sigma_t^2 = \frac{\beta_t(1-\alpha_{t-1})}{1-\alpha_t}$$

Reversing the Diffusion



- The joint distribution of data and latents

$$p(\mathbf{x},\mathbf{z}_1,\ldots,\mathbf{z}_T|\mathbf{w}) = p(\mathbf{z}_T)\left\{\prod_{t=2}^{T} p(\mathbf{z}_{t-1}|\mathbf{z}_t,\mathbf{w})\right\} p(\mathbf{x}|\mathbf{z}_1,\mathbf{w})$$

- Let's assume $p(\mathbf{z}_{t-1}|\mathbf{z}_t,\mathbf{w}) = \mathcal{N}(\mathbf{z}_{t-1}|\boldsymbol{\mu}(\mathbf{z}_t,\mathbf{w},t),\beta_t\mathbf{I})$

Note that $\boldsymbol{\mu}$ represents the denoising model (e.g., a neural net) which denoises z_t to produce z_{t-1}

- The true joint distribution of the latents given $\boldsymbol{x}$

$$q(z_1,z_2,\ldots,z_T|x) = q(z_1|x)\prod_{t=2}^{T} q(z_t|z_{t-1},x)$$

- To estimate $\boldsymbol{w}$, we can maximize the ELBO defined as

This term is just like the VAE reconstruction error term (can approximate it using samples of z_1 from $q(z_1|x)$

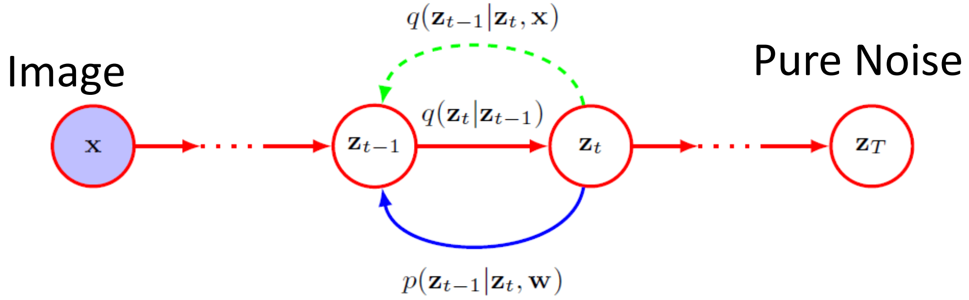
$$\mathcal{L}(\mathbf{w}) = \mathbb{E}_q\left[\ln\frac{p(\mathbf{z}_T)\left\{\prod_{t=2}^{T} p(\mathbf{z}_{t-1}|\mathbf{z}_t,\mathbf{w})\right\}p(\mathbf{x}|\mathbf{z}_1,\mathbf{w})}{q(\mathbf{z}_1|\mathbf{x})\prod_{t=2}^{T} q(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{x})}\right] = \mathbb{E}_q\left[\ln p(\mathbf{z}_T) + \sum_{t=2}^{T}\ln\frac{p(\mathbf{z}_{t-1}|\mathbf{z}_t,\mathbf{w})}{q(\mathbf{z}_t|\mathbf{z}_{t-1},\mathbf{x})} - \ln q(\mathbf{z}_1|\mathbf{x}) + \ln p(\mathbf{x}|\mathbf{z}_1,\mathbf{w})\right]$$

From ELBO definition $\mathbb{E}_q\left[\log\frac{p(X,Z)}{q(Z)}\right]$

Also note that unlike VI, here we aren't estimating the q distribution

First and third terms don't contain $\boldsymbol{w}$ so can be ignored when maximizing the ELBO

ELBO (contd)



- Recall the ELBO for the denoising diffusion model

$$\mathcal{L}(\mathbf{w}) = \mathbb{E}_q\left[\ln p(\mathbf{z}_T) + \sum_{t=2}^{T} \ln \frac{p(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{w})}{q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x})} - \ln q(\mathbf{z}_1|\mathbf{x}) + \ln p(\mathbf{x}|\mathbf{z}_1, \mathbf{w})\right]$$

- Ignoring terms that don't depend on $\boldsymbol{w}$ and using $q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x}) = \frac{q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})q(\mathbf{z}_t|\mathbf{x})}{q(\mathbf{z}_{t-1}|\mathbf{x})}$

$$\ln \frac{p(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{w})}{q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x})} = \ln \frac{p(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{w})}{q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})} + \ln \frac{q(\mathbf{z}_{t-1}|\mathbf{x})}{q(\mathbf{z}_t|\mathbf{x})} \Longrightarrow \mathcal{L}(\mathbf{w}) = \mathbb{E}_q\left[\sum_{t=2}^{T} \ln \frac{p(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{w})}{q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})} + \ln p(\mathbf{x}|\mathbf{z}_1, \mathbf{w})\right]$$

- The ELBO becomes $\mathcal{L}(\mathbf{w}) = \underbrace{\int q(\mathbf{z}_1|\mathbf{x}) \ln p(\mathbf{x}|\mathbf{z}_1, \mathbf{w})\, d\mathbf{z}_1}_{\text{reconstruction term}} - \underbrace{\sum_{t=2}^{T} \int \mathrm{KL}(q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x}) \| p(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{w}))q(\mathbf{z}_t|\mathbf{x})\, d\mathbf{z}_t}_{\text{consistency terms}}$

- Since both distributions in the KL divergence term are Gaussians, it becomes

$$\mathrm{KL}(q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x}) \| p(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{w})) = \frac{1}{2\beta_t} \|\mathbf{m}_t(\mathbf{x}, \mathbf{z}_t) - \boldsymbol{\mu}(\mathbf{z}_t, \mathbf{w}, t)\|^2 + \text{const}$$

Predicting the noise

- The KL terms in the ELBO are of the form

Network which gives the mean of the denoised z_{t-1}

$$\mathrm{KL}(q(\mathbf{z}_{t-1}|\mathbf{z}_t,\mathbf{x})||p(\mathbf{z}_{t-1}|\mathbf{z}_t,\mathbf{w})) = \frac{1}{2\beta_t}\left\|\mathbf{m}_t(\mathbf{x},\mathbf{z}_t)-\boldsymbol{\mu}(\mathbf{z}_t,\mathbf{w},t)\right\|^2+\text{const}$$

- Note that

From the definition of $m_t(x,z_t)$

$$\mathbf{x}=\frac{1}{\sqrt{\alpha_t}}\mathbf{z}_t-\frac{\sqrt{1-\alpha_t}}{\sqrt{\alpha_t}}\boldsymbol{\epsilon}_t \Rightarrow \mathbf{m}_t(\mathbf{x},\mathbf{z}_t)=\frac{1}{\sqrt{1-\beta_t}}\left\{\mathbf{z}_t-\frac{\beta_t}{\sqrt{1-\alpha_t}}\boldsymbol{\epsilon}_t\right\}$$

- Instead of learning $\mu_t(z_t,w,t)$, we will learn a noise predictor $g(z_t,w,t)$ s.t.

Using the same form as m_t with $g(z_t,w,t)$ trying to predict ϵ_t

$$\boldsymbol{\mu}(\mathbf{z}_t,\mathbf{w},t)=\frac{1}{\sqrt{1-\beta_t}}\left\{\mathbf{z}_t-\frac{\beta_t}{\sqrt{1-\alpha_t}}\mathbf{g}(\mathbf{z}_t,\mathbf{w},t)\right\}$$

- Therefore

$$\begin{aligned}&\mathrm{KL}(q(\mathbf{z}_{t-1}|\mathbf{z}_t,\mathbf{x})||p(\mathbf{z}_{t-1}|\mathbf{z}_t,\mathbf{w}))\\&=\frac{\beta_t}{2(1-\alpha_t)(1-\beta_t)}\left\|\mathbf{g}(\mathbf{z}_t,\mathbf{w},t)-\boldsymbol{\epsilon}_t\right\|^2+\text{const}\\&=\frac{\beta_t}{2(1-\alpha_t)(1-\beta_t)}\left\|\mathbf{g}(\sqrt{\alpha_t}\mathbf{x}+\sqrt{1-\alpha_t}\boldsymbol{\epsilon}_t,\mathbf{w},t)-\boldsymbol{\epsilon}_t\right\|^2+\text{const}\end{aligned}$$

Basically, we are now just predicting the noise ϵ_t using the neural network $g(z_t,w,t)$

Predicting the noise

- We basically had the following

$$\mathrm{KL}(q(\mathbf{z}_{t-1}|\mathbf{z}_t,\mathbf{x})\|p(\mathbf{z}_{t-1}|\mathbf{z}_t,\mathbf{w})) = \frac{\beta_t}{2(1-\alpha_t)(1-\beta_t)}\left\|\mathbf{g}(\sqrt{\alpha_t}\mathbf{x}+\sqrt{1-\alpha_t}\boldsymbol{\epsilon}_t,\mathbf{w},t)-\boldsymbol{\epsilon}_t\right\|^2 + \text{const}$$

- The reconstruction error part in the ELBO can also be written as noise prediction

$$\ln p(\mathbf{x}|\mathbf{z}_1,\mathbf{w}) = -\frac{1}{2\beta_1}\|\mathbf{x}-\boldsymbol{\mu}(\mathbf{z}_1,\mathbf{w},1)\|^2 + \text{const.} = -\frac{1}{2(1-\beta_t)}\|\mathbf{g}(\mathbf{z}_1,\mathbf{w},1)-\boldsymbol{\epsilon}_1\|^2 + \text{const}$$

- Ignoring the constants in front of the squared error terms above, the ELBO becomes

Empirically found to give improved performance

$$\mathcal{L}(\mathbf{w}) = \underbrace{\int q(\mathbf{z}_1|\mathbf{x})\ln p(\mathbf{x}|\mathbf{z}_1,\mathbf{w})\,\mathrm{d}\mathbf{z}_1}_{\text{reconstruction term}} - \underbrace{\sum_{t=2}^{T}\int \mathrm{KL}(q(\mathbf{z}_{t-1}|\mathbf{z}_t,\mathbf{x})\|p(\mathbf{z}_{t-1}|\mathbf{z}_t,\mathbf{w}))q(\mathbf{z}_t|\mathbf{x})\,\mathrm{d}\mathbf{z}_t}_{\text{consistency terms}}$$

$$= -\sum_{t=1}^{T}\left\|\mathbf{g}(\sqrt{\alpha_t}\mathbf{x}+\sqrt{1-\alpha_t}\boldsymbol{\epsilon}_t,\mathbf{w},t)-\boldsymbol{\epsilon}_t\right\|^2$$

Pick an example x randomly, generate a corruption z_t by sampling ϵ_t and make a gradient based update to w

Can optimize using stochastic optimization

Denoising Diffusion Model: The Training Algo

- The overall training algo is as follows

Input: Training data $\mathcal{D} = \{\mathbf{x}_n\}$
Noise schedule $\{\beta_1, \ldots, \beta_T\}$
Output: Network parameters $\mathbf{w}$

---

for $t \in \{1, \ldots, T\}$ **do**
    $\alpha_t \leftarrow \prod_{\tau=1}^{t}(1-\beta_\tau)$ `// Calculate alphas from betas`
end for
repeat
    $\mathbf{x} \sim \mathcal{D}$ `// Sample a data point`
    $t \sim \{1, \ldots, T\}$ `// Sample a point along the Markov chain`
    $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0}, \mathbf{I})$ `// Sample a noise vector`
    $\mathbf{z}_t \leftarrow \sqrt{\alpha_t}\mathbf{x} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}$ `// Evaluate noisy latent variable`
    $\mathcal{L}(\mathbf{w}) \leftarrow \|\mathbf{g}(\mathbf{z}_t, \mathbf{w}, t) - \boldsymbol{\epsilon}\|^2$ `// Compute loss term`
    Take optimizer step
until converged
return $\mathbf{w}$

Pseudo-code from Bishop & Bishop (2023)

Denoising Diffusion Model: Generation

- Using the training model, we can now generate data as follows

Input: Trained denoising network $\mathbf{g}(\mathbf{z}, \mathbf{w}, t)$
Noise schedule $\{\beta_1, \ldots, \beta_T\}$
Output: Sample vector $\mathbf{x}$ in data space

$\mathbf{z}_T \sim \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I})$ `// Sample from final latent space`
for $t \in T, \ldots, 2$ **do**
  $\alpha_t \leftarrow \prod_{\tau=1}^{t}(1-\beta_\tau)$ `// Calculate alpha`
  `// Evaluate network output`
  $\boldsymbol{\mu}(\mathbf{z}_t, \mathbf{w}, t) \leftarrow \frac{1}{\sqrt{1-\beta_t}}\left\{\mathbf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\mathbf{g}(\mathbf{z}_t, \mathbf{w}, t)\right\}$
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0}, \mathbf{I})$ `// Sample a noise vector`
  $\mathbf{z}_{t-1} \leftarrow \boldsymbol{\mu}(\mathbf{z}_t, \mathbf{w}, t) + \sqrt{\beta_t}\boldsymbol{\epsilon}$ `// Add scaled noise`
end for
$\mathbf{x} = \frac{1}{\sqrt{1-\beta_1}}\left\{\mathbf{z}_1 - \frac{\beta_1}{\sqrt{1-\alpha_1}}\mathbf{g}(\mathbf{z}_1, \mathbf{w}, t)\right\}$ `// Final denoising step`
return $\mathbf{x}$

Pseudo-code from Bishop & Bishop (2023)