# Diffusion Models

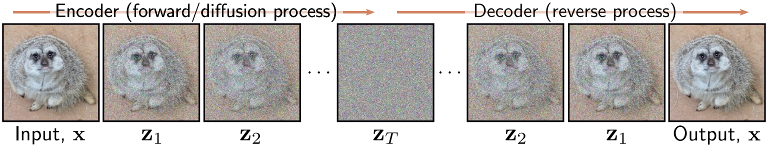
Naresh Manwani

Machine Learning Lab, IIIT Hyderabad

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

HYDERABAD

1. A diffusion model consists of an encoder and a decoder.
2. The encoder takes a data sample $\mathbf{x}$ and maps it through a series of intermediate latent variables $\mathbf{z}_1, \ldots, \mathbf{z}_T$.
3. The decoder reverses this process; it starts with $\mathbf{z}_T$ and maps back through $\mathbf{z}_{T-1}, \ldots, \mathbf{z}_1$ until it recreates a data point $\mathbf{x}$.
4. In both encoder and decoder, the mappings are stochastic rather than deterministic.



# Encoder (Forward Process)

The diffusion or forward process maps a data example $\mathbf{x}$ through a series of intermediate variables $\mathbf{z}_1, \ldots, \mathbf{z}_T$ according to:

$$
\begin{aligned}
\mathbf{z}_1 &= \sqrt{1-\beta_1}\,\mathbf{x} + \sqrt{\beta_1}\,\epsilon_1 \\
\mathbf{z}_t &= \sqrt{1-\beta_t}\,\mathbf{z}_{t-1} + \sqrt{\beta_t}\,\epsilon_t, \quad t \in \{1, \ldots, T\}
\end{aligned}
$$

- $\epsilon_t$ is noise drawn from a standard normal distribution ($\mathcal{N}(\mathbf{0}, I)$).
- The hyperparameters, $\beta_t \in [0,1],\ t = 1 \ldots T$, determine how quickly the noise is blended.
- $\beta_t,\ t = 1 \ldots T$ are collectively known as the noise schedule.

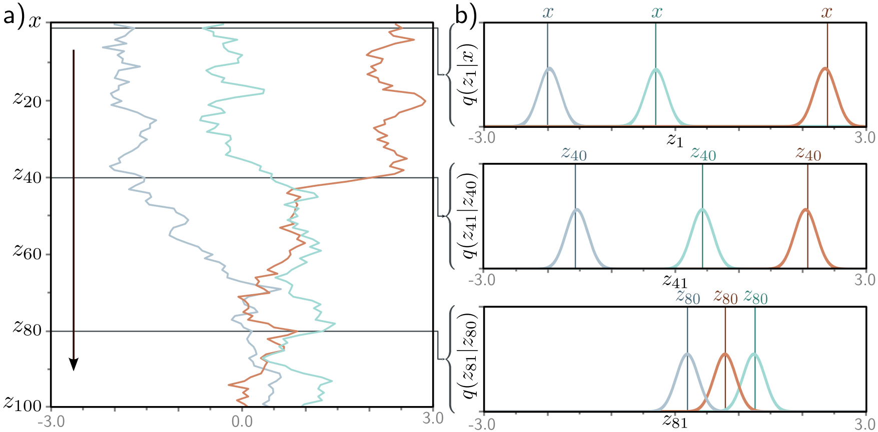



Figure: $\beta_t = 0.03$, $\forall t \in \{1, \ldots, 100\}$. The conditional probabilities $q(\mathbf{z}_1|\mathbf{x})$ and $q(\mathbf{z}_t|\mathbf{z}_{t-1})$ are normal distributions with a mean that is slightly closer to zero than the current point and a fixed variance $\beta_t$.

# Equivalent Forward Process

The forward process can equivalently be written as:

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

1. This is a Markov chain because the probability $\mathbf{z}_t$ depends only on the value of the immediately preceding variable $\mathbf{z}_{t-1}$.
2. With sufficient steps $T$, $q(\mathbf{z}_T|\mathbf{x}) = q(\mathbf{z}_T)$ becomes a standard normal distribution.
3. The joint distribution of $\mathbf{z}_1, \ldots, \mathbf{z}_T$ given $\mathbf{x}$ is as follows.

$$q(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_T|\mathbf{x}) = q(\mathbf{z}_1|\mathbf{x}) \prod_{t=2}^{T} q(\mathbf{z}_t|\mathbf{z}_{t-1})$$

# Diffusion Kernel $q(\mathbf{z}_t|\mathbf{x})$

1. We can see that

$$\mathbf{z}_t = \sqrt{\alpha_t}\,\mathbf{x} + \sqrt{1-\alpha_t}\,\epsilon$$

   where $\alpha_t = \prod_{s=1}^{t}(1-\beta_s)$ and $\epsilon \sim \mathcal{N}(\mathbf{0}, I)$.

2. Thus,

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

3. **Advantage of Diffusion Kernel:** Having a closed form expression for $q(\mathbf{z}_t|\mathbf{x})$ allows us to directly draw samples $\mathbf{z}_t$ given initial data point $\mathbf{x}$ without computing the intermediate variables $\mathbf{z}_1, \ldots, \mathbf{z}_{t-1}$.

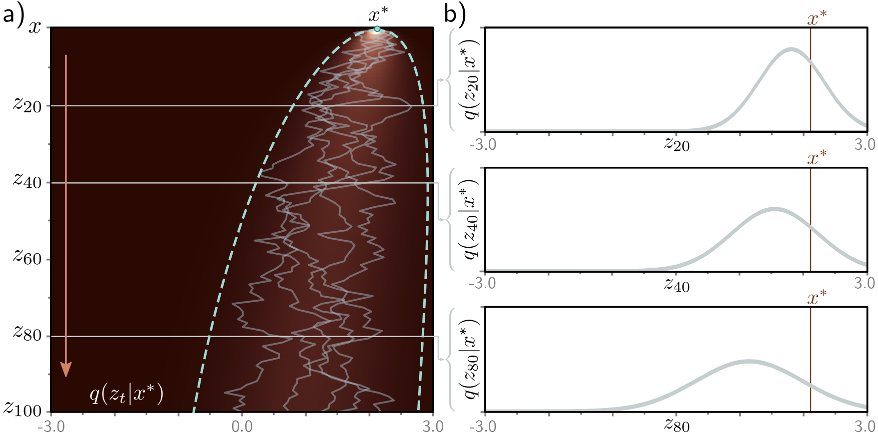
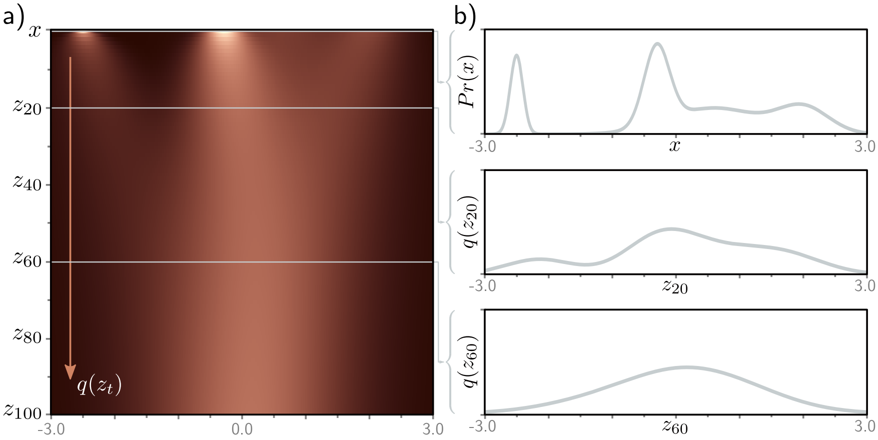



Figure: Started with $x^* = 2$. Cyan lines show $\pm 2$ standard deviations from the mean.

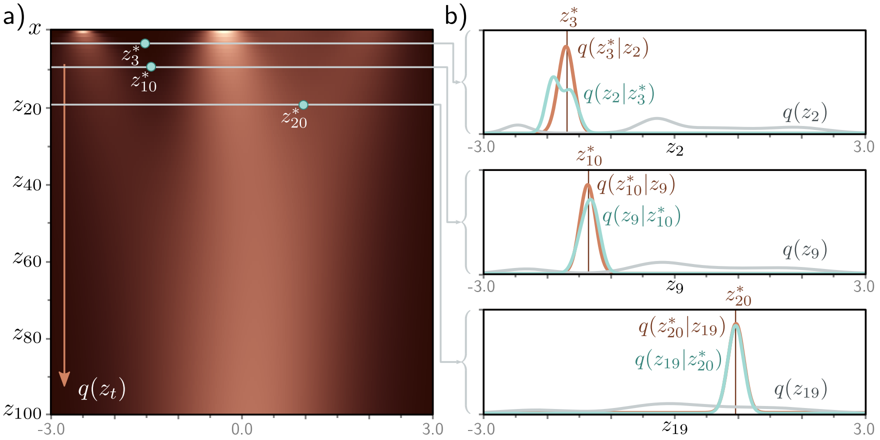
# Marginal Distribution $q(\mathbf{z}_t)$ and Conditional Distrib… $q(\mathbf{z}_{t-1}|\mathbf{z}_t)$

1. **Marginal Distribution:** $q(\mathbf{z}_t) = \int q(\mathbf{z}_t|\mathbf{x})q(\mathbf{x})d\mathbf{x}$
   - Cannot write closed form for $q(\mathbf{z}_t)$ because we do not $q(\mathbf{x})$.
2. **Conditional Distribution** $q(\mathbf{z}_{t-1}|\mathbf{z}_t)$**:** $q(\mathbf{z}_{t-1}|\mathbf{z}_t) = \frac{q(\mathbf{z}_t|\mathbf{z}_{t-1})q(\mathbf{z}_{t-1})}{q(\mathbf{z}_t)}$.
   - This is intractable since we can not compute the marginal distribution $q(\mathbf{z}_{t-1})$.

# Example: Marginal Diffusion Density $q(\mathbf{z}_t)$



# Example: Conditional Distribution $q(\mathbf{z}_{t-1}|\mathbf{z}_t)$



# Conditional Diffusion Distribution $q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})$

1. Conditional distribution $q(\mathbf{z}_{t-1}|\mathbf{z}_t)$ is intractable since we can not compute the marginal distribution $q(\mathbf{z}_{t-1})$.
2. However, if we know the starting variable $\mathbf{x}$, then we do know the distribution $q(\mathbf{z}_{t-1}|\mathbf{x})$ (this is diffusion kernel and has Gaussian density).
3. Thus, it is possible to find closed form of distribution $q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})$.
4. This is used in decoder.

# Computation of Conditional Diffusion Distribution $q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})$

$$
\begin{aligned}
q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x}) &= \frac{q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{x})q(\mathbf{z}_{t-1}|\mathbf{x})}{q(\mathbf{z}_t|\mathbf{x})} \\
&\propto q(\mathbf{z}_t|\mathbf{z}_{t-1})q(\mathbf{z}_{t-1}|\mathbf{x}) \\
&= \mathcal{N}_{\mathbf{z}_t}(\sqrt{1-\beta_t}\ \mathbf{z}_{t-1}, \beta_t I)\ \ \mathcal{N}_{\mathbf{z}_{t-1}}(\sqrt{\alpha_{t-1}}\ \mathbf{x}, (1-\alpha_{t-1})I) \\
&\propto \mathcal{N}_{\mathbf{z}_{t-1}}(\frac{1}{\sqrt{1-\beta_t}}\ \mathbf{z}_t, \frac{\beta_t}{1-\beta_t} I)\ \ \mathcal{N}_{\mathbf{z}_{t-1}}(\sqrt{\alpha_{t-1}}\ \mathbf{x}, (1-\alpha_{t-1})I)
\end{aligned}
$$

where we have used the following change of variable result for Gaussian distribution.

$$
\mathcal{N}_{\mathbf{x}}(A\mathbf{y} + \mathbf{b}, \Sigma) \propto \mathcal{N}_{\mathbf{y}}((A^T\Sigma A)^{-1}A^T\Sigma^{-1}(\mathbf{x} - \mathbf{b}), (A^T\Sigma A)^{-1})
$$

# Computation of Conditional Diffusion Distribution $q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})$–Continue

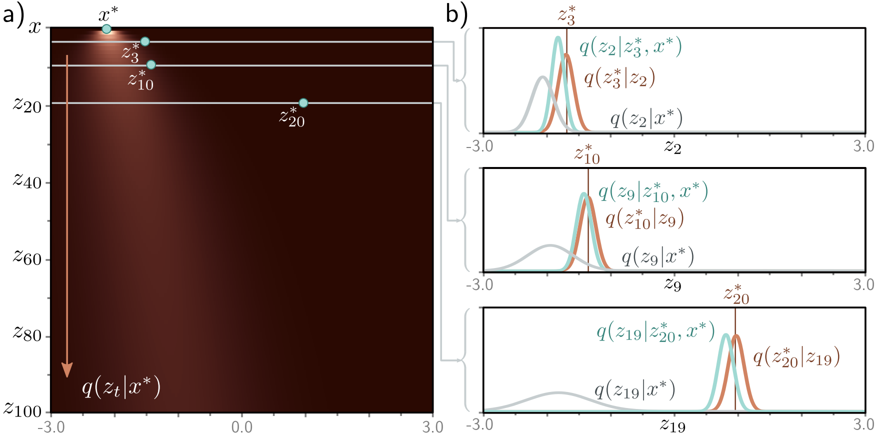
1. We now use the following reults

$$\mathcal{N}_{\mathbf{w}}(\mathbf{a}, A)\mathcal{N}_{\mathbf{w}}(\mathbf{b}, B) \propto \mathcal{N}_{\mathbf{w}}((A^{-1} + B^{-1})^{-1})(A^{-1}\mathbf{a} + B^{-1}\mathbf{b}, (A^{-1} + B^{-1})^{-1})$$

2. We get the following form for $p(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})$

$$q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x}) = \mathcal{N}_{\mathbf{z}_{t-1}}\left(\frac{1-\alpha_{t-1}}{1-\alpha_t}\sqrt{1-\beta_t}\,\mathbf{z}_t + \frac{\sqrt{\alpha_{t-1}}}{1-\alpha_t}\beta_t\mathbf{x}, \frac{\beta_t(1-\alpha_{t-1})}{1-\alpha_t}I\right)$$

# Example: Conditional Distribution $q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})$



# Decoder Model: Reverse Process

1. In decoder, we learn a series of probabilistic mappings back from latent variable $\mathbf{z}_T$ to $\mathbf{z}_{T-1}$, from $\mathbf{z}_{T-1}$ to $\mathbf{z}_{T-2}$, and so on, until we reach the data $\mathbf{x}$.
2. The true reverse distributions $p(\mathbf{z}_{t-1}|\mathbf{z}_t)$ of the diffusion process are complex multi-modal distributions that depend on the data distribution $p(\mathbf{x})$.
3. We approximate these as normal distributions.

$$
\begin{aligned}
p(\mathbf{z}_T) &= \mathcal{N}(\mathbf{0}, I) \\
p(\mathbf{z}_{t-1}|\mathbf{z}_t, \phi_t) &= \mathcal{N}(\mathbf{f}_t(\mathbf{z}_t, \phi_t), \sigma_t^2 I) \\
p(\mathbf{x}|\mathbf{z}_1, \phi_1) &= \mathcal{N}(\mathbf{f}_1(\mathbf{z}_1, \phi_1), \sigma_1^2 I)
\end{aligned}
$$

   where $\mathbf{f}_t(\mathbf{z}_t, \phi_t)$ is a neural network that computes the mean of the normal distribution in the estimated mapping from $\mathbf{z}_t$ to $\mathbf{z}_{t-1}$.
4. The terms $\sigma_t^2,\ t = 1 \ldots T$ are predetermined.

# Training

1. The joint distribution of the observed variable $\mathbf{x}$ and the latent variables $\mathbf{z}_1, \ldots, \mathbf{z}_T$ is
$$p(\mathbf{x}, \mathbf{z}_{1\ldots T} | \phi_{1\ldots T}) = p(\mathbf{x}|\mathbf{z}_1, \phi_1) \prod_{t=2}^{T} p(\mathbf{z}_{t-1}|\mathbf{z}_t, \phi_t)\; p(\mathbf{z}_T)$$
2. The likelihood of the observed data is found by marginalizing over the latent variables
$$p(\mathbf{x}|\phi_{1\ldots T}) = \int p(\mathbf{x}, \mathbf{z}_{1\ldots T}|\phi_{1\ldots T}) d\mathbf{z}_{1\ldots T} \tag{1}$$
3. To train the model, we maximize the log-likelihood of the training data $\{\mathbf{x}_i\}$ with respect to $\phi_{1\ldots T}$
$$\hat{\phi}_{1\ldots T} = \arg\max_{\phi_{1\ldots T}} \sum_{i=1}^{N} \log p(\mathbf{x}_i|\phi_{1\ldots T})$$
4. We can't maximize this directly because the marginalization in equation (1) is intractable.
5. We use Jensen's inequality to define a lower bound on the likelihood and optimize the parameters $\phi_{1\ldots T}$ with respect to this bound.

# Evidence Lower Bound (ELBO)

$$\begin{aligned}\log p(\mathbf{x}|\phi_{1\ldots T}) &= \log\left[\int p(\mathbf{x},\mathbf{z}_{1\ldots T}|\phi_{1\ldots T})d\mathbf{z}_{1\ldots T}\right] \\ &= \log\left[\int q(\mathbf{z}_{1\ldots T}|\mathbf{x})\frac{p(\mathbf{x},\mathbf{z}_{1\ldots T}|\phi_{1\ldots T})}{q(\mathbf{z}_{1\ldots T}|\mathbf{x})}\right]d\mathbf{z}_{1\ldots T} \\ &\geq \int q(\mathbf{z}_{1\ldots T}|\mathbf{x})\log\left[\frac{p(\mathbf{x},\mathbf{z}_{1\ldots T}|\phi_{1\ldots T})}{q(\mathbf{z}_{1\ldots T}|\mathbf{x})}\right]d\mathbf{z}_{1\ldots T}\end{aligned}$$

This gives the evidence lower bound (ELBO):

$$ELBO[\phi_{1\ldots T}] = \int q(\mathbf{z}_{1\ldots T}|\mathbf{x})\log\left[\frac{p(\mathbf{x},\mathbf{z}_{1\ldots T}|\phi_{1\ldots T})}{q(\mathbf{z}_{1\ldots T}|\mathbf{x})}\right]d\mathbf{z}_{1\ldots T}$$

1. In diffusion models, the decoder is trained to make the bound tighter by changing its parameters

# Simplified ELBO

$$
\begin{aligned}
ELBO[\phi_{1\dots T}] &= \mathbb{E}_{q(\mathbf{z}_1|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{z}_1, \phi_1)\right] \\
&\quad - \sum_{t=2}^{T} \mathbb{E}_{q(\mathbf{z}_1|\mathbf{x})}\left[KL[q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})||p(\mathbf{z}_{t-1}|\mathbf{z}_t, \phi_t)]\right]
\end{aligned}
$$

- The first probability term in the ELBO is

$$p(\mathbf{x}|\mathbf{z}_1, \phi_1) = \mathcal{N}_{\mathbf{z}_{t-1}}(\mathbf{f}_t[\mathbf{z}_t, \phi_t], \sigma_t^2 I)$$

  The ELBO will be larger if the model prediction matches the observed data.

- $KL[q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})||p(\mathbf{z}_{t-1}|\mathbf{z}_t, \phi_t)]$ has closed form expression. Many of the terms are independent of $\phi_t$.

$$KL[q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x})||p(\mathbf{z}_{t-1}|\mathbf{z}_t, \phi_t)]$$
$$= \left\| \frac{1-\alpha_{t-1}}{1-\alpha_t}\sqrt{1-\beta_t}\,\mathbf{z}_t + \frac{\sqrt{\alpha_{t-1}}\beta_t}{1-\alpha_t}\mathbf{x} - \mathbf{f}_t[\mathbf{z}_t, \phi_t] \right\|^2 + C$$

# Diffusion Loss Function

- To fit the model, we maximize the ELBO with respect to the parameters $\phi_{1\ldots T}$.
- We recast this as a minimization by multiplying with minus one and approximating the expectations with samples to give the loss function:

$$L[\phi_{1\ldots T}] = \sum_{i=1}^{I} -\log\left(\mathcal{N}_{\mathbf{x}_i}(\mathbf{f}_1[\mathbf{z}_{i1}, \phi_1], \sigma_1^2 I)\right) + \sum_{t=2}^{T} \left\| \frac{1-\alpha_{t-1}}{1-\alpha_t}\sqrt{1-\beta_t}\,\mathbf{z}_{it} + \frac{\sqrt{\alpha_{t-1}}\beta_t}{1-\alpha_t}\mathbf{x}_i - \mathbf{f}_t[\mathbf{z}_{it}, \phi_t] \right\|^2$$

where $\mathbf{x}_i$ is the $i^{th}$ data point and $\mathbf{z}_{it}$ is the associated latent variable at $t^{th}$ diffusion step.
- $\mathbf{f}_t[\mathbf{z}_t, \phi_t]$ predicts the value of $\mathbf{z}_{t-1}$.

# Reparametrization of the loss function

- Diffusion models have been found to work better with a different parameterization.
- The loss function is modified so that the model aims to predict the noise that was mixed with the original data example to create the current variable.
- To achieve that, we reparameterize both the target and the network.

# Reparameterization of the Target

- The diffusion update is
$$\mathbf{z}_t = \sqrt{\alpha_t}\mathbf{x} + \sqrt{1-\alpha_t}\,\boldsymbol{\epsilon}$$
- It follows that the data term $\mathbf{x}$ can be expressed as the diffused image $\mathbf{z}_t$ minus the noise that was added to it.
$$\mathbf{x} = \frac{1}{\sqrt{\alpha_t}}\mathbf{z}_t - \frac{\sqrt{1-\alpha_t}}{\sqrt{\alpha_t}}\boldsymbol{\epsilon}$$
- Using this, the target is modified as
$$\frac{1-\alpha_{t-1}}{1-\alpha_t}\sqrt{1-\beta_t}\,\mathbf{z}_t + \frac{\sqrt{\alpha_{t-1}}\beta_t}{1-\alpha_t}\mathbf{x} = \frac{1}{\sqrt{1-\beta_t}}\mathbf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}}\boldsymbol{\epsilon}$$
- We replace the model $\hat{\mathbf{z}}_{t-1} = \mathbf{f}_t[\mathbf{z}_t, \boldsymbol{\phi}_t]$ as
$$\mathbf{f}_t[\mathbf{z}_t, \boldsymbol{\phi}_t] = \frac{1}{\sqrt{1-\beta_t}}\mathbf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}}\mathbf{g}_t[\mathbf{z}_t, \boldsymbol{\phi}_t]$$

  where $\mathbf{g}_t[\mathbf{z}_t, \boldsymbol{\phi}_t]$ tries to approximate the noise $\epsilon$ that was added to $\mathbf{x}$ to create $\mathbf{z}_t$.

# Reparamterized Loss

$$
\begin{aligned}
L[\phi_{1\ldots T}] &= \sum_{i=1}^{I}\sum_{t=1}^{T}\left\|\mathbf{g}_t[\mathbf{z}_{it}, \phi_t] - \epsilon_{it}\right\|^2 \\
&= \sum_{i=1}^{I}\sum_{t=1}^{T}\left\|\mathbf{g}_t[\sqrt{\alpha_t}\mathbf{x}_i + \sqrt{1-\alpha_t}\,\epsilon_{it}, \phi_t] - \epsilon_{it}\right\|^2
\end{aligned}
$$

# Diffusion Model Training

**Algorithm 18.1:** Diffusion model training

**Input:** Training data $\mathbf{x}$
**Output:** Model parameters $\boldsymbol{\phi}_t$
**repeat**
  **for** $i \in \mathcal{B}$ **do** `// For every training example index in batch`
    $t \sim \text{Uniform}[1, \ldots T]$ `// Sample random timestep`
    $\boldsymbol{\epsilon} \sim \text{Norm}[\mathbf{0}, \mathbf{I}]$ `// Sample noise`
    $\ell_i = \left\| \mathbf{g}_t\left[\sqrt{\alpha_t}\mathbf{x}_i + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}, \boldsymbol{\phi}_t\right] - \boldsymbol{\epsilon} \right\|^2$ `// Compute individual loss`
  Accumulate losses for batch and take gradient step
**until** converged

- The training algorithm is simple to implement.
- It naturally augments the dataset. We can reuse every original data point $\mathbf{x}_i$ as many times as we want at each time step with different noise instantiations $\epsilon$.

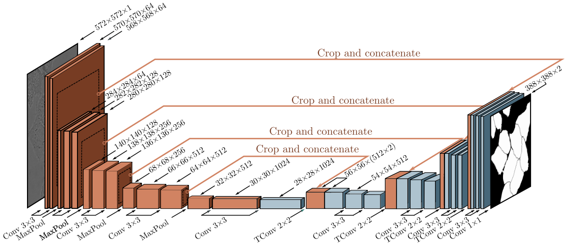
# Sampling from trained model

**Algorithm 18.2:** Sampling

**Input:** Model, $\mathbf{g}_t[\bullet, \phi_t]$
**Output:** Sample, $\mathbf{x}$
$\mathbf{z}_T \sim \text{Norm}_{\mathbf{z}}[\mathbf{0}, \mathbf{I}]$ `// Sample last latent variable`
**for** $t = T \ldots 2$ **do**
  $\hat{\mathbf{z}}_{t-1} = \frac{1}{\sqrt{1-\beta_t}}\mathbf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}}\mathbf{g}_t[\mathbf{z}_t, \phi_t]$ `// Predict previous latent variable`
  $\boldsymbol{\epsilon} \sim \text{Norm}_{\boldsymbol{\epsilon}}[\mathbf{0}, \mathbf{I}]$ `// Draw new noise vector`
  $\mathbf{z}_{t-1} = \hat{\mathbf{z}}_{t-1} + \sigma_t \boldsymbol{\epsilon}$ `// Add noise to previous latent variable`
$\mathbf{x} = \frac{1}{\sqrt{1-\beta_1}}\mathbf{z}_1 - \frac{\beta_1}{\sqrt{1-\alpha_1}\sqrt{1-\beta_1}}\mathbf{g}_1[\mathbf{z}_1, \phi_1]$ // Generate sample from $\mathbf{z}_1$ without noise

- The sampling algorithm requires serial processing of many neural networks $\mathbf{g}_t[\mathbf{z}_t, \phi_t]$ and is hence time-consuming.

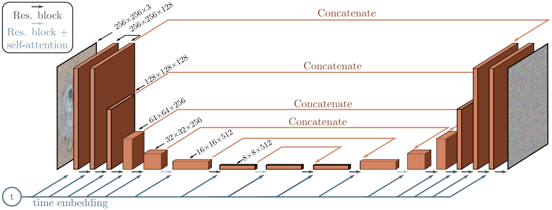
- Here, we need to construct models that can take a noisy image and predict the noise that was added at each step.
- The architectural choice for this image-to-image mapping is the U-Net.




- UNet is a semantic segmentation network that had an encoder-decoder.
- The encoder repeatedly downsamples the image until the receptive fields are large and information is integrated from across the image.
- Then the decoder upsamples it back to the size of the original image.
- The final output is a probability over possible object classes at each pixel.

# Diffusion UNet

- There may be a very large number of diffusion steps, and training and storing multiple U-Nets is inefficient.
- The solution is to train a single U-Net that also takes a predetermined vector representing the time step as input.



# Training Diffusion UNet

- A large number of time steps are needed as the conditional probabilities $q(\mathbf{z}_{t-1}|\mathbf{z}_t)$ become closer to normal when the hyperparameters $\beta_t$ are close to zero, matching the form of the decoder distributions $p(\mathbf{z}_{t-1}|\mathbf{z}_t, \phi_t)$.
- However, this makes sampling slow.
- We might have to run the U-Net model through $T = 1000$ steps to generate good images.