MiRL Medical Imaging & Reconstruction Lab

Diffusion Models

Ganapathy Krishnamurthi

Overview

- Consiss of an encoder and a decoder
- Encoder takes data sample x and maps it through a series of intermediate latent variables, z1.. zT.
- The decoder reverses this process.
  - It starts with zT and maps it back through zT-1 to z1 until it recreates x
- In both encoder and decoder, the mappings are stochastics rather than deterministic

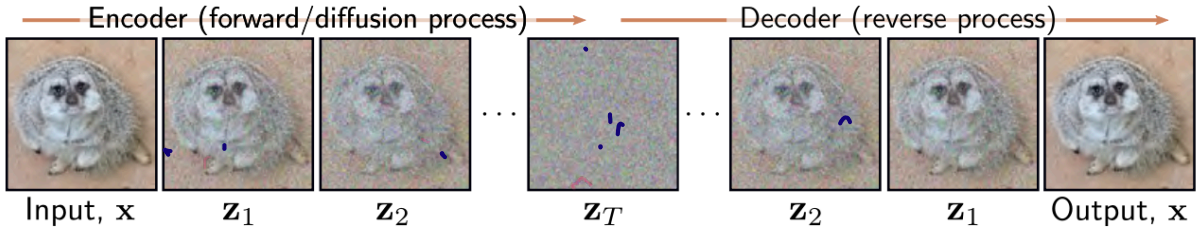
Overview- Encoder

- Encoder is pre-specified- it adds noise to the input gradually
- With enough noise addition steps the information in the image is wiped out and we are left only with noise
- All learned parameters are in the decoder

Overview-Decoder

- A series of networks are trained to map backward betwen each of the adjacent pair of variabels zt and zt-1
- The loss function encourages the network to invert the corresponding encoder step
- Noise is gradually removed until a realistic looking data is obtained
- To generate a new data example x, we draw from the noise distribution and pass it through the decoder

Encoder-Decoder



- Diffusion models. The encoder (forward, or diffusion process) maps the input x through a series of latent variables z1 . . . zT .
- This process is pre- specified and gradually mixes the data with noise until only noise remains.
- The decoder (reverse process) is learned and passes the data back through the latent variables, removing noise at each stage.
- After training, new examples are generated by sampling noise vectors zT and passing them through the decoder.

Encoder

Noise Schedule

$\beta \rightarrow$ Noise schedule $\quad 0 \cdot 1$

$\beta \rightarrow$ can be time dependent

$\epsilon_t \rightarrow N(0, \sigma)$

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

$$\mathbf{z}_t = \sqrt{1-\beta_t} \cdot \mathbf{z}_{t-1} + \sqrt{\beta_t} \cdot \boldsymbol{\epsilon}_t \qquad \forall\, t \in 2, \ldots, T,$$

$$z_2 = \sqrt{1-\beta_2}\, z_1 + \sqrt{\beta_2} \cdot \epsilon_2$$

$$q(\mathbf{z}_1|\mathbf{x}) = \text{Norm}_{\mathbf{z}_1}\left[\sqrt{1-\beta_1}\mathbf{x}, \beta_1\mathbf{I}\right]$$

$$q(\mathbf{z}_t|\mathbf{z}_{t-1}) = \text{Norm}_{\mathbf{z}_t}\left[\sqrt{1-\beta_t}\mathbf{z}_{t-1}, \beta_t\mathbf{I}\right] \qquad \forall\, t \in \{2, \ldots, T\}.$$

Decoder

$$
\begin{aligned}
Pr(\mathbf{z}_T) &= \text{Norm}_{\mathbf{z}_T}[\mathbf{0}, \mathbf{I}] \\
Pr(\mathbf{z}_{t-1}|\mathbf{z}_t, \boldsymbol{\phi}_t) &= \text{Norm}_{\mathbf{z}_{t-1}}\Big[\mathbf{f}_t[\mathbf{z}_t, \boldsymbol{\phi}_t], \sigma_t^2\mathbf{I}\Big] \\
Pr(\mathbf{x}|\mathbf{z}_1, \boldsymbol{\phi}_1) &= \text{Norm}_{\mathbf{x}}\Big[\mathbf{f}_1[\mathbf{z}_1, \boldsymbol{\phi}_1], \sigma_1^2\mathbf{I}\Big],
\end{aligned}
$$

$$
\begin{aligned}
q(\mathbf{z}_1|\mathbf{x}) &= \text{Norm}_{\mathbf{z}_1}\Big[\sqrt{1-\beta_1}\mathbf{x}, \beta_1\mathbf{I}\Big] \\
q(\mathbf{z}_t|\mathbf{z}_{t-1}) &= \text{Norm}_{\mathbf{z}_t}\Big[\sqrt{1-\beta_t}\mathbf{z}_{t-1}, \beta_t\mathbf{I}\Big] \qquad \forall\, t \in \{2, \ldots, T\}.
\end{aligned}
$$

known

Now we replace the model $\hat{\mathbf{z}}_{t-1} = \mathbf{f}_t[\mathbf{z}_t, \boldsymbol{\phi}_t]$ with a new model $\hat{\boldsymbol{\epsilon}} = \mathbf{g}_t[\mathbf{z}_t, \boldsymbol{\phi}_t]$ which predicts the noise $\boldsymbol{\epsilon}$ that was mixed with $\mathbf{x}$ to create $\mathbf{z}_t$:

$$
\mathbf{f}_t[\mathbf{z}_t, \boldsymbol{\phi}_t] = \frac{1}{\sqrt{1-\beta_t}}\mathbf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}}\mathbf{g}_t[\mathbf{z}_t, \boldsymbol{\phi}_t].
$$

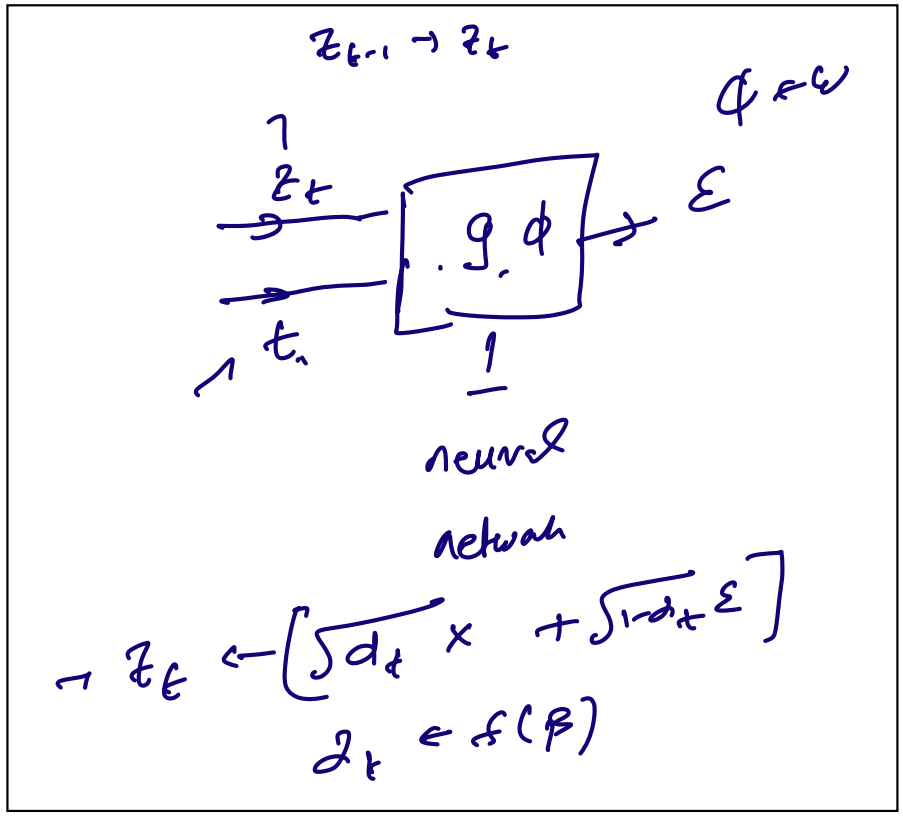
Training Algorithm

Algorithm 18.1: Diffusion model training

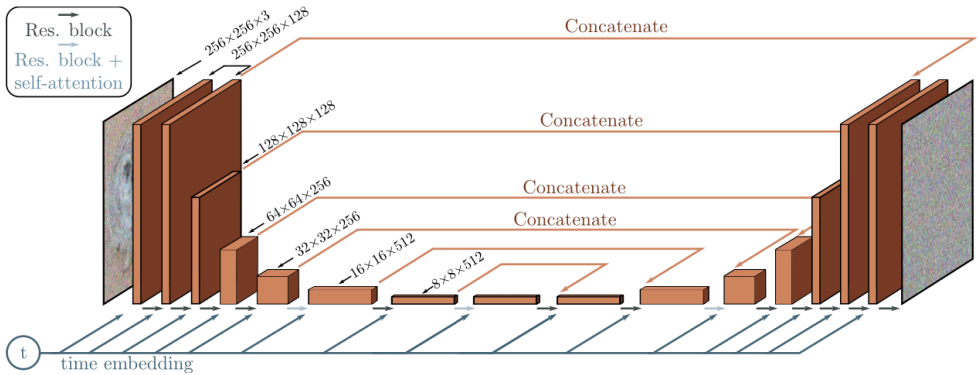
Input: Training data $\mathbf{x}$
Output: Model parameters ϕ_t
repeat
  for $i \in \mathcal{B}$ **do** // For every training example index in batch
    $t \sim \text{Uniform}[1, \ldots T]$ // Sample random timestep
    $\boldsymbol{\epsilon} \sim \text{Norm}[\mathbf{0}, \mathbf{I}]$ // Sample noise
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

Algorithm 18.2: Sampling

Input: Model, $\mathbf{g}_t[\bullet, \boldsymbol{\phi}_t]$
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**
  $\hat{\mathbf{z}}_{t-1} = \frac{1}{\sqrt{1-\beta_t}}\mathbf{z}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}\sqrt{1-\beta_t}}\mathbf{g}_t[\mathbf{z}_t, \boldsymbol{\phi}_t]$ // 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
$\mathbf{x} = \frac{1}{\sqrt{1-\beta_1}}\mathbf{z}_1 - \frac{\beta_1}{\sqrt{1-\alpha_1}\sqrt{1-\beta_1}}\mathbf{g}_1[\mathbf{z}_1, \boldsymbol{\phi}_1]$ // Generate sample from $\mathbf{z}_1$ without noise

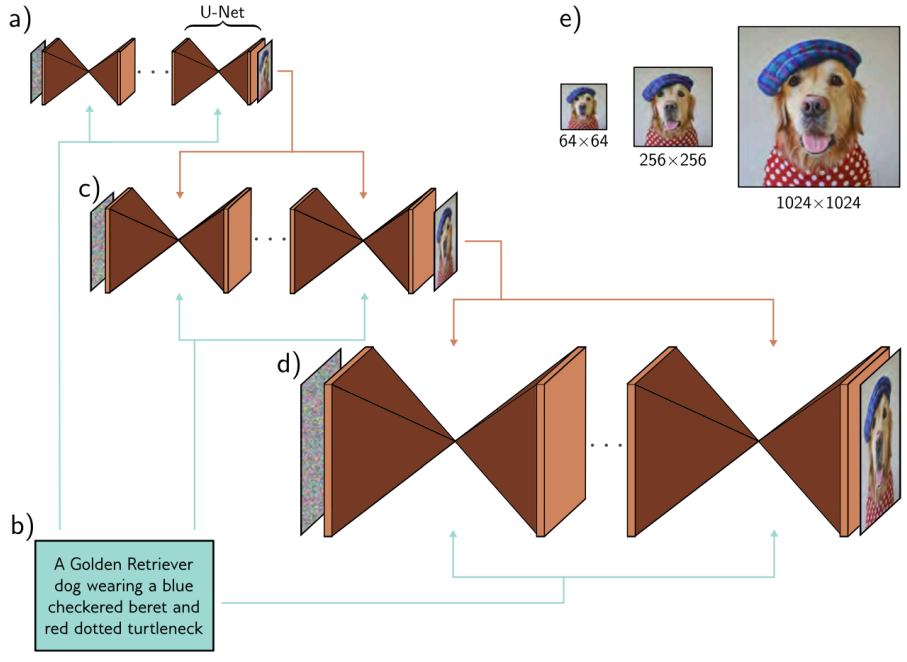


Network Architecture



- U-Net as used in diffusion models for images. The network aims to predict the noise that was added to the image.
- It consists of an encoder which reduces the scale and increases the number of channels and a decoder which in- creases the scale and reduces the number of channels.
- The encoder representations are concatenated to their partner in the decoder. Connections between adjacent representations consist of residual blocks, and periodic global self-attention in which every spatial position interacts with every other spatial position.
- A single network is used for all time steps, by passing a sinusoidal time embedding through a shallow neural network and adding the result to the channels at every spatial position at every stage of the U-Net.

Network Architecture-Conditional Generation



Cascaded conditional generation based on a text prompt.

a) A diffusion model consisting of a series of U-Nets is used to generate a 64×64 image.

b) This generation is conditioned on a sentence embedding computed by a language model.

c) A higher resolution 256×256 image is generated and conditioned on the smaller image and the text encoding.

d) This is repeated to create a 1024×1024 image. e) Final image sequence. Adapted from Saharia et al. (2022b).

Generated outputs - Conditional Generation