Autoencoders and Generative Models

CISC 7026: Introduction to Deep Learning

University of Macau

Agenda

1. Review
2. Compression
3. Autoencoders
4. Applications
5. Variational Modeling
6. VAE Implementation
7. Coding

Agenda

1. **Review**
2. Compression
3. Autoencoders
4. Applications
5. Variational Modeling
6. VAE Implementation
7. Coding

Review

Convolution works over inputs of any variables (time, space, etc)

Review

Convolution works over inputs of any variables (time, space, etc)

Recurrent neural networks only work with time

Review

Convolution works over inputs of any variables (time, space, etc)

Recurrent neural networks only work with time

Convolution makes use of locality and translation equivariance properties

Review

Convolution works over inputs of any variables (time, space, etc)

Recurrent neural networks only work with time

Convolution makes use of locality and translation equivariance properties

Recurrent models do not assume locality or equivariance

Review

Convolution works over inputs of any variables (time, space, etc)

Recurrent neural networks only work with time

Convolution makes use of locality and translation equivariance properties

Recurrent models do not assume locality or equivariance

Equivariance and locality make learning more efficient, but not all problems have this structure

Review

How do humans process temporal data?

Review

How do humans process temporal data?

Humans only perceive the present

Review

How do humans process temporal data?

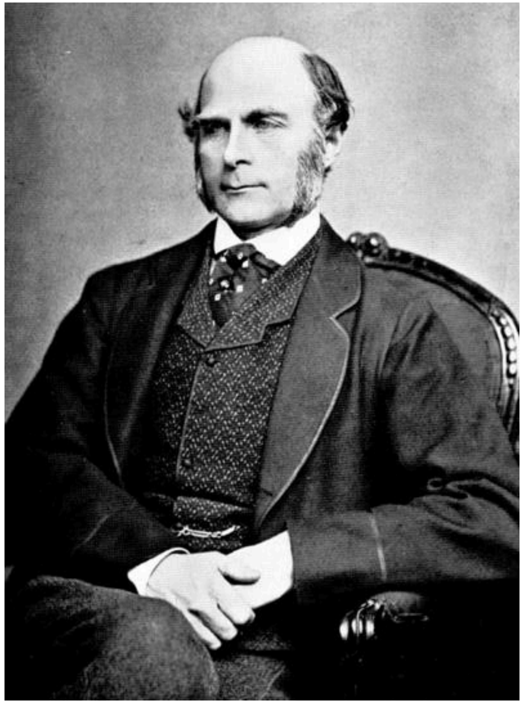
Humans only perceive the present

Humans process temporal data by storing and recalling memories

Review

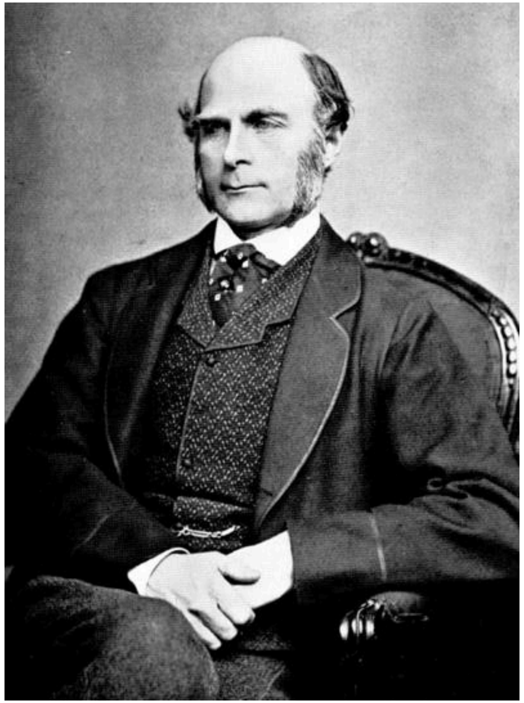
Francis Galton (1822-1911)

photo composite memory

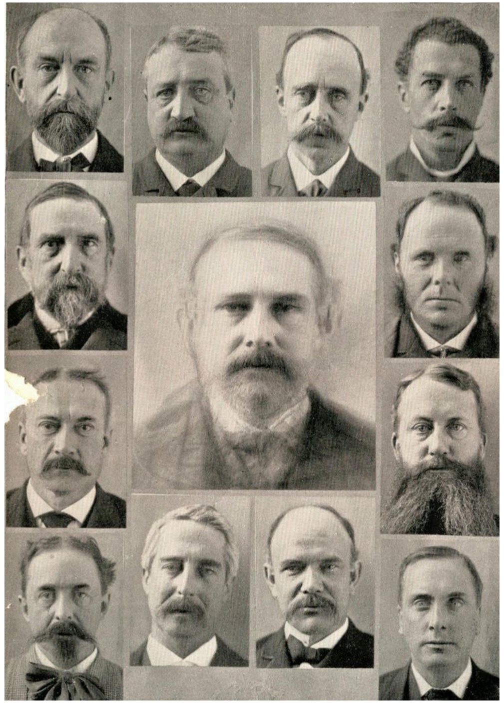


Review

Francis Galton (1822-1911) photo composite memory



Composite photo of members of a party



Review

Task: Find a mathematical model of how our mind represents memories

Review

Task: Find a mathematical model of how our mind represents memories

$$X : \mathbb{R}^{h \times w} \quad \text{People you see at the party}$$

Review

Task: Find a mathematical model of how our mind represents memories

$$X : \mathbb{R}^{h \times w} \quad \text{People you see at the party}$$

$$H : \mathbb{R}^{h \times w} \quad \text{The image in your mind}$$

$$f : X^T \times \Theta \mapsto H$$

Review

Task: Find a mathematical model of how our mind represents memories

$$X : \mathbb{R}^{h \times w} \quad \text{People you see at the party}$$

$$H : \mathbb{R}^{h \times w} \quad \text{The image in your mind}$$

$$f : X^T \times \Theta \mapsto H$$

Composite photography/memory uses a weighted sum

Review

Task: Find a mathematical model of how our mind represents memories

$$X : \mathbb{R}^{h \times w} \quad \text{People you see at the party}$$

$$H : \mathbb{R}^{h \times w} \quad \text{The image in your mind}$$

$$f : X^T \times \Theta \mapsto H$$

Composite photography/memory uses a weighted sum

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{i=1}^{T} \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_i$$

Review

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{i=1}^{T} \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_i$$

Review

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{i=1}^{T} \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_i$$

What if we see a new face?

Review

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{i=1}^{T} \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_i$$

What if we see a new face?

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \left(\sum_{i=1}^{T} \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_i \right) + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{\text{new}}$$

Review

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{i=1}^{T} \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_i$$

What if we see a new face?

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \left(\sum_{i=1}^{T} \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_i \right) + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{\text{new}}$$

We repeat the same process for each new face

Review

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{i=1}^{T} \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_i$$

What if we see a new face?

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \left(\sum_{i=1}^{T} \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_i \right) + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{\text{new}}$$

We repeat the same process for each new face

We can rewrite f as a **recurrent function**

Review

Let us rewrite composite memory as a recurrent function

Review

Let us rewrite composite memory as a recurrent function

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \underbrace{\left(\sum_{i=1}^{T} \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_i \right)}_{\boldsymbol{h}} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{\text{new}}$$

Review

Let us rewrite composite memory as a recurrent function

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \underbrace{\left(\sum_{i=1}^{T} \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_i \right)}_{\boldsymbol{h}} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{\text{new}}$$

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

Review

Let us rewrite composite memory as a recurrent function

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \underbrace{\left(\sum_{i=1}^{T} \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_i \right)}_{\boldsymbol{h}} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{\text{new}}$$

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

$$\boldsymbol{x} \in \mathbb{R}^{d_x}, \quad \boldsymbol{h} \in \mathbb{R}^{d_h}$$

Review

$$\boldsymbol{x} \in \mathbb{R}^{d_x}, \quad \boldsymbol{h} \in \mathbb{R}^{d_h}$$

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

Review

$$\boldsymbol{x} \in \mathbb{R}^{d_x}, \quad \boldsymbol{h} \in \mathbb{R}^{d_h} \qquad\qquad f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

$$\boldsymbol{h}_1 = f(\boldsymbol{0}, \boldsymbol{x}_1, \boldsymbol{\theta}) = \boldsymbol{0} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1$$

Review

$$\boldsymbol{x} \in \mathbb{R}^{d_x}, \quad \boldsymbol{h} \in \mathbb{R}^{d_h} \qquad\qquad f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

$$\boldsymbol{h}_1 = f(\boldsymbol{0}, \boldsymbol{x}_1, \boldsymbol{\theta}) = \boldsymbol{0} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1$$

$$\boldsymbol{h}_2 = f(\boldsymbol{h}_1, \boldsymbol{x}_2, \boldsymbol{\theta}) = \boldsymbol{h}_1 + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2$$

Review

$$\boldsymbol{x} \in \mathbb{R}^{d_x}, \quad \boldsymbol{h} \in \mathbb{R}^{d_h} \qquad\qquad f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

$$\boldsymbol{h}_1 = f(\boldsymbol{0}, \boldsymbol{x}_1, \boldsymbol{\theta}) = \boldsymbol{0} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1$$

$$\boldsymbol{h}_2 = f(\boldsymbol{h}_1, \boldsymbol{x}_2, \boldsymbol{\theta}) = \boldsymbol{h}_1 + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2$$

$$\boldsymbol{h}_3 = f(\boldsymbol{h}_2, \boldsymbol{x}_3, \boldsymbol{\theta}) = \boldsymbol{h}_2 + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3$$

Review

$$\boldsymbol{x} \in \mathbb{R}^{d_x}, \quad \boldsymbol{h} \in \mathbb{R}^{d_h} \qquad\qquad f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

$$\boldsymbol{h}_1 = f(\boldsymbol{0}, \boldsymbol{x}_1, \boldsymbol{\theta}) = \boldsymbol{0} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1$$

$$\boldsymbol{h}_2 = f(\boldsymbol{h}_1, \boldsymbol{x}_2, \boldsymbol{\theta}) = \boldsymbol{h}_1 + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2$$

$$\boldsymbol{h}_3 = f(\boldsymbol{h}_2, \boldsymbol{x}_3, \boldsymbol{\theta}) = \boldsymbol{h}_2 + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3$$

$$\vdots$$

$$\boldsymbol{h}_T = f(\boldsymbol{h}_{T-1}, \boldsymbol{x}_T, \boldsymbol{\theta}) = \boldsymbol{h}_{T-1} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_T$$

Review

$$\boldsymbol{x} \in \mathbb{R}^{d_x}, \quad \boldsymbol{h} \in \mathbb{R}^{d_h} \qquad f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

$$\boldsymbol{h}_1 = f(\boldsymbol{0}, \boldsymbol{x}_1, \boldsymbol{\theta}) = \boldsymbol{0} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1$$

$$\boldsymbol{h}_2 = f(\boldsymbol{h}_1, \boldsymbol{x}_2, \boldsymbol{\theta}) = \boldsymbol{h}_1 + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2$$

$$\boldsymbol{h}_3 = f(\boldsymbol{h}_2, \boldsymbol{x}_3, \boldsymbol{\theta}) = \boldsymbol{h}_2 + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3$$

$$\vdots$$

$$\boldsymbol{h}_T = f(\boldsymbol{h}_{T-1}, \boldsymbol{x}_T, \boldsymbol{\theta}) = \boldsymbol{h}_{T-1} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_T$$

Review

Right now, our model remembers everything

Review

Right now, our model remembers everything

But $\boldsymbol{h}$ is a fixed size, what if T is very large?

Review

Right now, our model remembers everything

But $\boldsymbol{h}$ is a fixed size, what if T is very large?

If we keep adding and adding $\boldsymbol{x}$ into $\boldsymbol{h}$, we will run out of space

Review

Right now, our model remembers everything

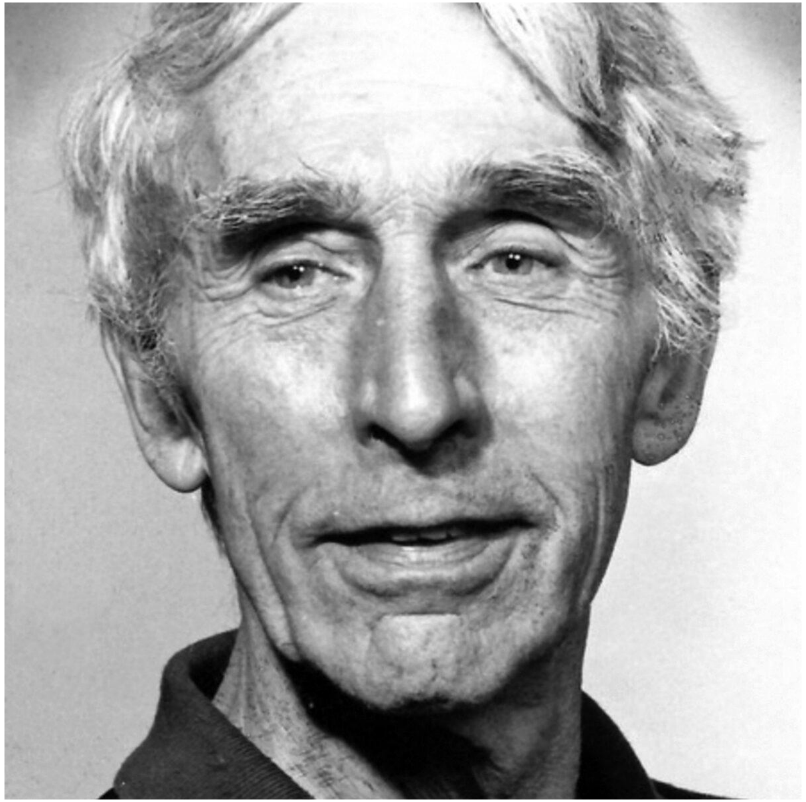
But $\boldsymbol{h}$ is a fixed size, what if T is very large?

If we keep adding and adding $\boldsymbol{x}$ into $\boldsymbol{h}$, we will run out of space

Humans forget old information

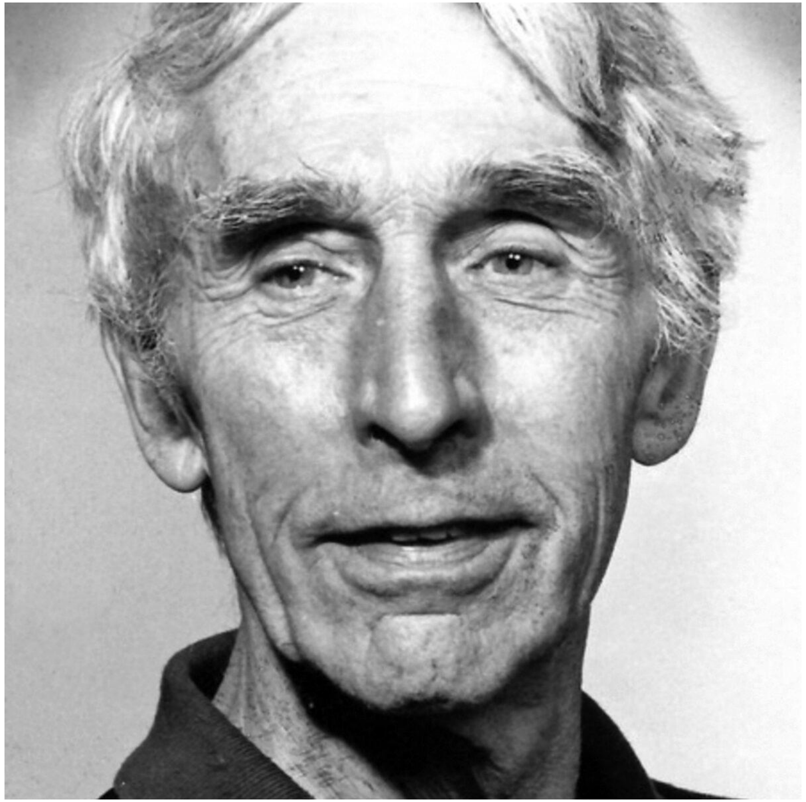
Review

Murdock (1982)



Review

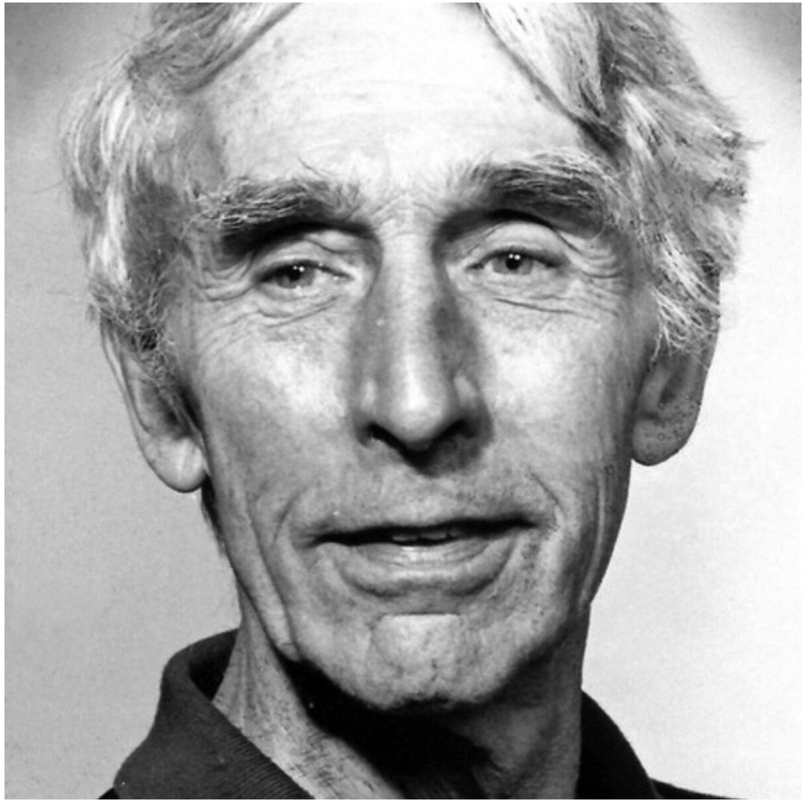
Murdock (1982)



$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = {\color{red}\gamma} \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}; \quad 0 < \gamma < 1$$

Review

Murdock (1982)

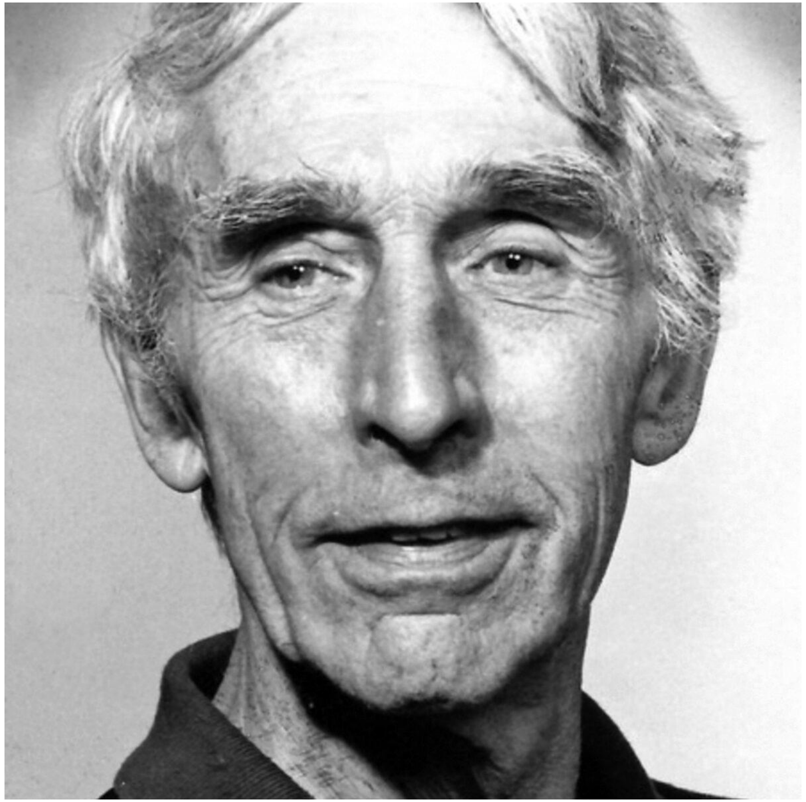


$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \gamma \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}; \quad 0 < \gamma < 1$$

Key Idea: $\lim_{T \to \infty} \gamma^T = 0$

Review

Murdock (1982)



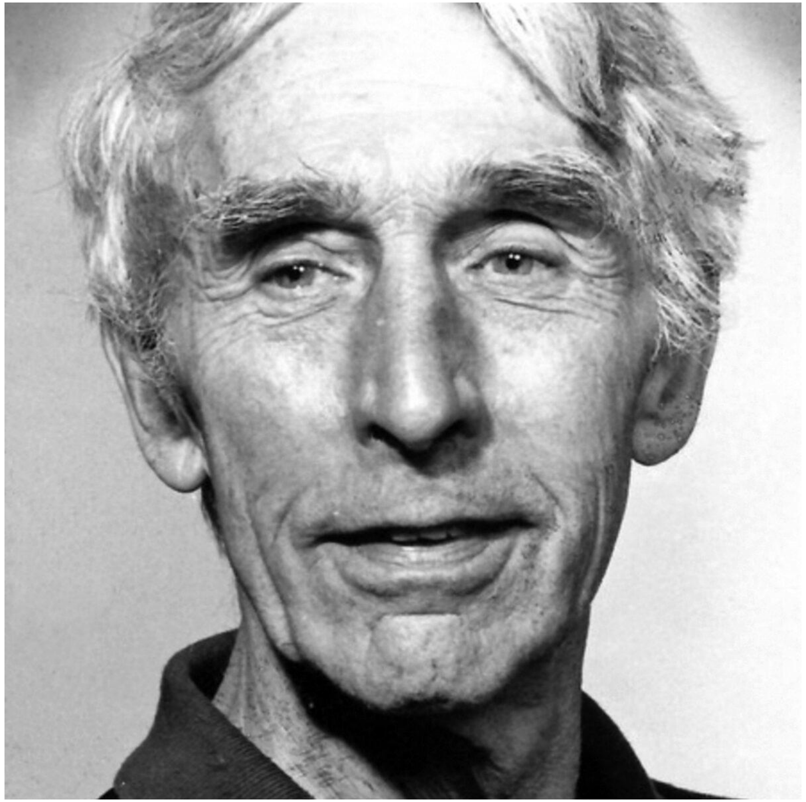
$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \gamma \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}; \quad 0 < \gamma < 1$$

Key Idea: $\lim_{T \to \infty} \gamma^T = 0$

Let $\gamma = 0.9$

Review

Murdock (1982)



$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \gamma \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}; \quad 0 < \gamma < 1$$

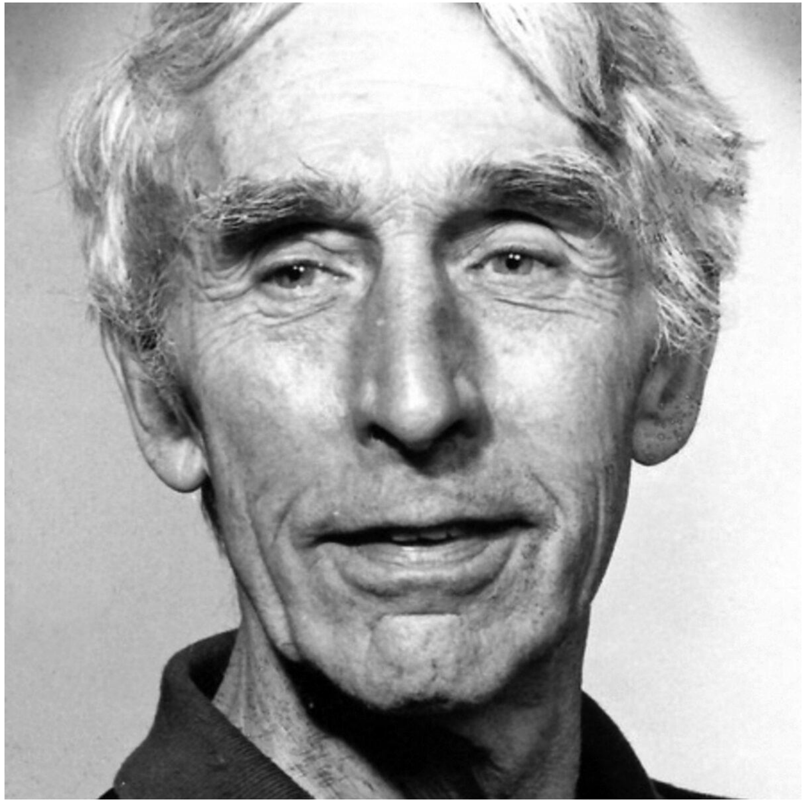
Key Idea: $\lim_{T \to \infty} \gamma^T = 0$

Let $\gamma = 0.9$

$$0.9 \cdot 0.9 \cdot \boldsymbol{h} = 0.81\boldsymbol{h}$$

Review

Murdock (1982)



$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \gamma \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}; \quad 0 < \gamma < 1$$

Key Idea: $\lim_{T \to \infty} \gamma^T = 0$

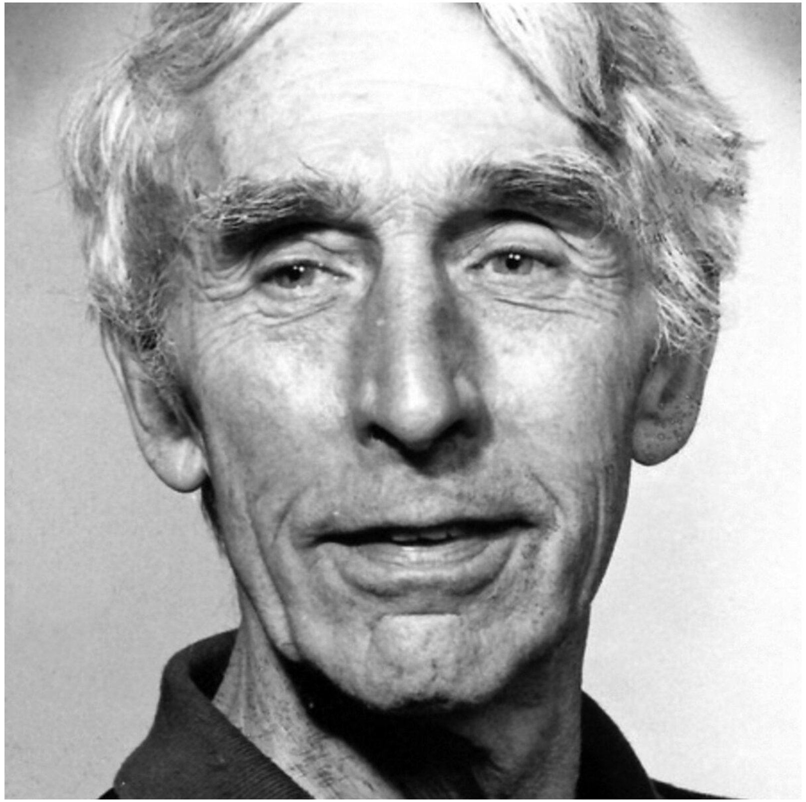
Let $\gamma = 0.9$

$$0.9 \cdot 0.9 \cdot \boldsymbol{h} = 0.81 \boldsymbol{h}$$

$$0.9 \cdot 0.9 \cdot 0.9 \cdot \boldsymbol{h} = 0.72 \boldsymbol{h}$$

Review

Murdock (1982)



$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \gamma \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}; \quad 0 < \gamma < 1$$

Key Idea: $\lim_{T \to \infty} \gamma^T = 0$

Let $\gamma = 0.9$

$$0.9 \cdot 0.9 \cdot \boldsymbol{h} = 0.81 \boldsymbol{h}$$

$$0.9 \cdot 0.9 \cdot 0.9 \cdot \boldsymbol{h} = 0.72 \boldsymbol{h}$$

$$0.9 \cdot 0.9 \cdot 0.9 \cdot \ldots \cdot \boldsymbol{h} = 0$$

Review

$$\boldsymbol{h}_T = \gamma^3 \boldsymbol{h}_{T-3} + \gamma^2 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{T-2} + \gamma \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{T-1} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_T$$

Review

$$\boldsymbol{h}_T = \gamma^3 \boldsymbol{h}_{T-3} + \gamma^2 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{T-2} + \gamma \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{T-1} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_T$$

$$\boldsymbol{h}_T = \gamma^T \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1 + \gamma^{T-1} \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2 + \ldots + \gamma^2 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{T-2} + \gamma \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{T-1} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_T$$

Review

Our function f is just defined for a single X

Review

Our function f is just defined for a single X

$$f : H \times X \times \Theta \mapsto H$$

Review

Our function f is just defined for a single X

$$f : H \times X \times \Theta \mapsto H$$

To extend f to sequences, we scan f over the inputs

$$\text{scan}(f) : H \times X^T \times \Theta \mapsto H^T$$

Review

Our function f is just defined for a single X

$$f : H \times X \times \Theta \mapsto H$$

To extend f to sequences, we scan f over the inputs

$$\text{scan}(f) : H \times X^T \times \Theta \mapsto H^T$$

$$\text{scan}(f)\left(\boldsymbol{h}_0, \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}\right) = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_T \end{bmatrix}$$

Review

$$f : H \times X \times \Theta \mapsto H, \quad \text{scan}(f) : H \times X^T \times \Theta \mapsto H^T$$

Review

$$f : H \times X \times \Theta \mapsto H, \quad \mathrm{scan}(f) : H \times X^T \times \Theta \mapsto H^T$$

$$\mathrm{scan}(f)\left(\boldsymbol{h}_0, \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}\right) = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_T \end{bmatrix}$$

Review

$$f : H \times X \times \Theta \mapsto H, \quad \mathrm{scan}(f) : H \times X^T \times \Theta \mapsto H^T$$

$$\mathrm{scan}(f)\left(\boldsymbol{h}_0, \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta} \right) = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_T \end{bmatrix}$$

$$\mathrm{scan}(f)\left(\boldsymbol{h}_0, \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta} \right) = \begin{bmatrix} f(h_0, x_1, \boldsymbol{\theta}) \\ f(h_1, x_2, \boldsymbol{\theta}) \\ \vdots \\ f(h_{T-1}, x_T, \boldsymbol{\theta}) \end{bmatrix} = \begin{bmatrix} f(h_0, x_1) \\ f(f(h_0, x_1), x_2) \\ \vdots \\ f(\dots f(h_0, x_1) \dots, x_T) \end{bmatrix}$$

Review

Let g define our memory recall function

Review

Let g define our memory recall function

$$g : H \times X \times \Theta \mapsto Y$$

Review

Let g define our memory recall function

$$g : H \times X \times \Theta \mapsto Y$$

g searches your memories h using the input x, to produce output y

Review

Let g define our memory recall function

$$g : H \times X \times \Theta \mapsto Y$$

g searches your memories h using the input x, to produce output y

$\boldsymbol{x}$: "What is your favorite ice cream flavor?"

Review

Let g define our memory recall function

$$g : H \times X \times \Theta \mapsto Y$$

g searches your memories h using the input x, to produce output y

$\boldsymbol{x}$: “What is your favorite ice cream flavor?”

$\boldsymbol{h}$: Everything you remember (hometown, birthday, etc)

Review

Let g define our memory recall function

$$g : H \times X \times \Theta \mapsto Y$$

g searches your memories h using the input x, to produce output y

$\boldsymbol{x}$: "What is your favorite ice cream flavor?"

$\boldsymbol{h}$: Everything you remember (hometown, birthday, etc)

g : Searches your memories for ice cream information, and responds "chocolate"

Review

Step 1: Perform scan to find recurrent states

Review

Step 1: Perform scan to find recurrent states

$$\begin{bmatrix} \boldsymbol{h}_1 \\ \vdots \\ \boldsymbol{h}_T \end{bmatrix} = \mathrm{scan}(f)\left(\boldsymbol{h}_0, \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}_f\right)$$

Review

Step 1: Perform scan to find recurrent states

$$\begin{bmatrix} \boldsymbol{h}_1 \\ \vdots \\ \boldsymbol{h}_T \end{bmatrix} = \mathrm{scan}(f) \left(\boldsymbol{h}_0, \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}_f \right)$$

Step 2: Perform recall on recurrent states

Review

Step 1: Perform scan to find recurrent states

$$\begin{bmatrix} \boldsymbol{h}_1 \\ \vdots \\ \boldsymbol{h}_T \end{bmatrix} = \text{scan}(f) \left(\boldsymbol{h}_0, \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}_f \right)$$

Step 2: Perform recall on recurrent states

$$\begin{bmatrix} \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_T \end{bmatrix} = \begin{bmatrix} g(\boldsymbol{h}_1, \boldsymbol{x}_1, \boldsymbol{\theta}_g) \\ \vdots \\ g(\boldsymbol{h}_T, \boldsymbol{x}_T, \boldsymbol{\theta}_g) \end{bmatrix}$$

Review

The simplest recurrent neural network is the **Elman Network**

Review

The simplest recurrent neural network is the **Elman Network**

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_1^\top \boldsymbol{h} + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}\right)$$

Review

The simplest recurrent neural network is the **Elman Network**

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}_1^\top \boldsymbol{h} + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}})$$

$$g(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{\theta}_3^\top \boldsymbol{h}$$

Review

Add forgetting

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_1^\top \boldsymbol{h} \odot f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}\right)$$

Review

Add forgetting

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_1^\top \boldsymbol{h} \odot f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}\right)$$

$$f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}} + \boldsymbol{\theta}_2^\top \boldsymbol{h}\right)$$

Review

Add forgetting

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_1^\top \boldsymbol{h} \odot f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}\right)$$

$$f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}} + \boldsymbol{\theta}_2^\top \boldsymbol{h}\right)$$

When $f_{\text{forget}} < 1$, we forget some of our memories!

Review

Add forgetting

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_1^\top \boldsymbol{h} \odot f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}\right)$$

$$f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}} + \boldsymbol{\theta}_2^\top \boldsymbol{h}\right)$$

When $f_{\text{forget}} < 1$, we forget some of our memories!

Through gradient descent, neural network learns which memories to forget

Review

Minimal gated unit (MGU) is a modern RNN

Review

Minimal gated unit (MGU) is a modern RNN

MGU defines two helper functions

Review

Minimal gated unit (MGU) is a modern RNN

MGU defines two helper functions

$$f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}} + \boldsymbol{\theta}_2^\top \boldsymbol{h})$$

Review

Minimal gated unit (MGU) is a modern RNN

MGU defines two helper functions

$$f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}} + \boldsymbol{\theta}_2^\top \boldsymbol{h})$$

$$f_2(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_3^\top \overline{\boldsymbol{x}} + \boldsymbol{\theta}_4^\top f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) \odot \boldsymbol{h}\right)$$

Review

Minimal gated unit (MGU) is a modern RNN

MGU defines two helper functions

$$f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}} + \boldsymbol{\theta}_2^\top \boldsymbol{h})$$

$$f_2(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_3^\top \overline{\boldsymbol{x}} + \boldsymbol{\theta}_4^\top f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) \odot \boldsymbol{h}\right)$$

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) \odot \boldsymbol{h} + \left(1 - f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta})\right) \odot f_2(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta})$$

Minimal gated unit (MGU) is a modern RNN

MGU defines two helper functions

$$f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}} + \boldsymbol{\theta}_2^\top \boldsymbol{h}\right)$$

$$f_2(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_3^\top \overline{\boldsymbol{x}} + \boldsymbol{\theta}_4^\top f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) \odot \boldsymbol{h}\right)$$

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) \odot \boldsymbol{h} + \left(1 - f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta})\right) \odot f_2(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta})$$

Left term forgets old, right term replaces forgotten memories

Review

Jax RNN https://colab.research.google.com/drive/147z7FNGyERV8oQ_4gZmxDVdeoNt0hKta#scrollTo=TUMonlJ1u8Va

Agenda

1. **Review**
2. Compression
3. Autoencoders
4. Applications
5. Variational Modeling
6. VAE Implementation
7. Coding

Agenda

1. Review
2. **Compression**
3. Autoencoders
4. Applications
5. Variational Modeling
6. VAE Implementation
7. Coding

Compression

Compression

Question: You watch a film. How do you communicate information about the film with a friend?

Compression

Question: You watch a film. How do you communicate information about the film with a friend?

Answer: An ogre and donkey rescue a princess, discovering friendship and love along the way.

Compression

Question: You watch a film. How do you communicate information about the film with a friend?

Answer: An ogre and donkey rescue a princess, discovering friendship and love along the way.

Question: What is missing?

Compression

Question: You watch a film. How do you communicate information about the film with a friend?

Answer: An ogre and donkey rescue a princess, discovering friendship and love along the way.

Question: What is missing?

Answer: Shrek lives in a swamp, Lord Farquaad, dragons, etc

Compression

When you discuss films with friends, you summarize them

Compression

When you discuss films with friends, you summarize them

This is a form of **compression**

Compression

When you discuss films with friends, you summarize them

This is a form of **compression**

$$f\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_n \end{bmatrix}\right) = \text{Green ogre and donkey save princess}$$

Compression

When you discuss films with friends, you summarize them

This is a form of **compression**

$$f\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_n \end{bmatrix}\right) = \text{Green ogre and donkey save princess}$$

In compression, we reduce the size of data by removing information

Compression

When you discuss films with friends, you summarize them

This is a form of **compression**

$$f\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_n \end{bmatrix}\right) = \text{Green ogre and donkey save princess}$$

In compression, we reduce the size of data by removing information

Let us examine a more principled form of video compression

Compression

Shrek in 4k UHD:

$$X \in \mathbb{Z}_{255}^{3 \times 3840 \times 2160}, X^{90 \times 60 \times 24}$$

Compression

Shrek in 4k UHD:

$$X \in \mathbb{Z}_{255}^{3 \times 3840 \times 2160}, X^{90 \times 60 \times 24}$$

Question: How many GB?

Compression

Shrek in 4k UHD:

$$X \in \mathbb{Z}_{255}^{3 \times 3840 \times 2160}, X^{90 \times 60 \times 24}$$

Question: How many GB?

Answer: 3000 GB

Compression

Shrek in 4k UHD:

$$X \in \mathbb{Z}_{255}^{3 \times 3840 \times 2160}, X^{90 \times 60 \times 24}$$

Question: How many GB?

Answer: 3000 GB

But if you download films, you know they are smaller than 3 TB

Compression

Shrek in 4k UHD:

$$X \in \mathbb{Z}_{255}^{3 \times 3840 \times 2160}, X^{90 \times 60 \times 24}$$

Question: How many GB?

Answer: 3000 GB

But if you download films, you know they are smaller than 3 TB

Today, we use the H.264 video **encoder** to transform videos into a more compact representation

Compression

H.264 encoder selects 16×16 pixel blocks, estimates shift between frames, applies cosine transform, ...

Compression

H.264 encoder selects 16×16 pixel blocks, estimates shift between frames, applies cosine transform, …

The result is an `.mp4` file Z

Compression

H.264 encoder selects 16×16 pixel blocks, estimates shift between frames, applies cosine transform, ...

The result is an `.mp4` file Z

$$f : X^t \mapsto Z \qquad\qquad Z \in \{0, 1\}^n$$

Compression

H.264 encoder selects 16×16 pixel blocks, estimates shift between frames, applies cosine transform, ...

The result is an `.mp4` file Z

$$f : X^t \mapsto Z \qquad\qquad Z \in \{0, 1\}^n$$

Question: What is the size of Z in GB?

Compression

H.264 encoder selects 16×16 pixel blocks, estimates shift between frames, applies cosine transform, ...

The result is an `.mp4` file Z

$$f : X^t \mapsto Z \qquad\qquad Z \in \{0, 1\}^n$$

Question: What is the size of Z in GB?

Answer: 60 GB, original size was 3000 GB

Compression

H.264 encoder selects 16×16 pixel blocks, estimates shift between frames, applies cosine transform, ...

The result is an `.mp4` file Z

$$f : X^t \mapsto Z \qquad\qquad Z \in \{0, 1\}^n$$

Question: What is the size of Z in GB?

Answer: 60 GB, original size was 3000 GB

We achieve a compression ratio of $3000 \text{ GB} \ / \ 60 \text{ GB} = 50$

Compression

What happens on your computer when you watch Shrek?

Compression

What happens on your computer when you watch Shrek?

Download $z \in Z$ from the internet

Compression

What happens on your computer when you watch Shrek?

Download $\boldsymbol{z} \in Z$ from the internet $\quad Z \in \{0, 1\}^n$

Compression

What happens on your computer when you watch Shrek?

Download $\boldsymbol{z} \in Z$ from the internet $\quad Z \in \{0, 1\}^n$

Information is no longer pixels, it is a string of bits

Compression

What happens on your computer when you watch Shrek?

Download $\boldsymbol{z} \in Z$ from the internet $\quad Z \in \{0, 1\}^n$

Information is no longer pixels, it is a string of bits

We must **decode** $\boldsymbol{z}$ back into pixels

Compression

What happens on your computer when you watch Shrek?

Download $\boldsymbol{z} \in Z$ from the internet $\quad Z \in \{0, 1\}^n$

Information is no longer pixels, it is a string of bits

We must **decode** $\boldsymbol{z}$ back into pixels

We need to undo (invert) the encoder f

$$f : X^t \mapsto Z$$

$$f^{-1} : Z \mapsto X^t$$

Compression

What happens on your computer when you watch Shrek?

Download $\boldsymbol{z} \in Z$ from the internet $\quad Z \in \{0, 1\}^n$

Information is no longer pixels, it is a string of bits

We must **decode** $\boldsymbol{z}$ back into pixels

We need to undo (invert) the encoder f

$$f : X^t \mapsto Z$$

$$f^{-1} : Z \mapsto X^t$$

You CPU has a H.264 decoder built in to make this fast

Compression

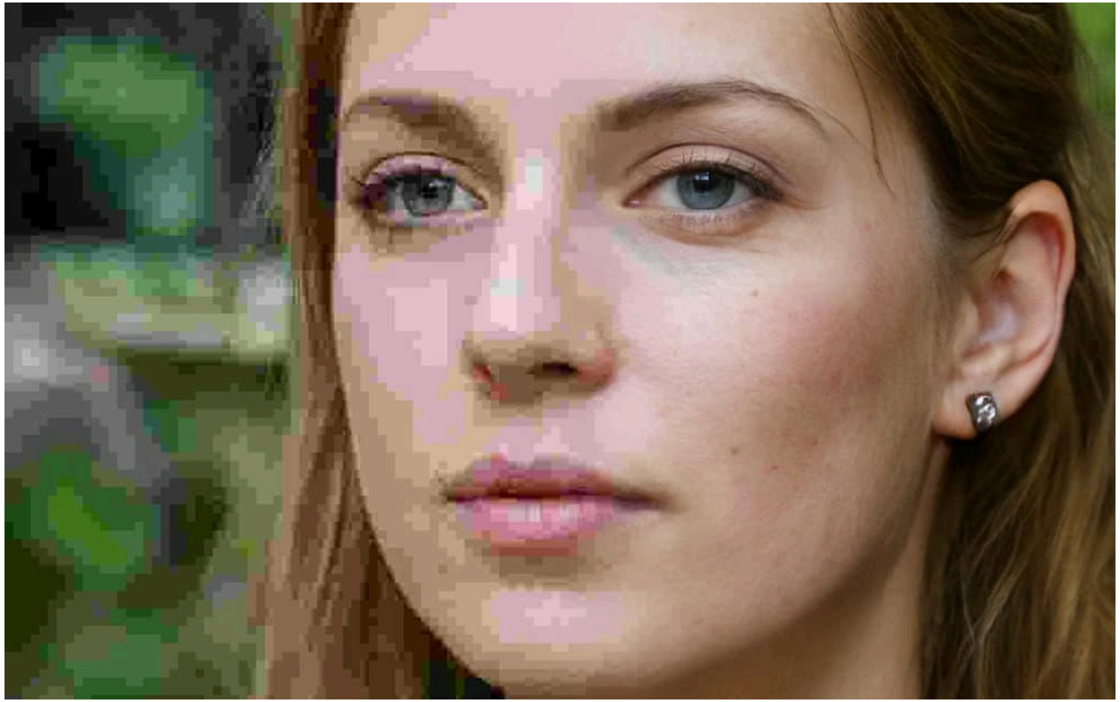
Compression may be **lossy**

Compression

Compression may be **lossy** or **lossless**

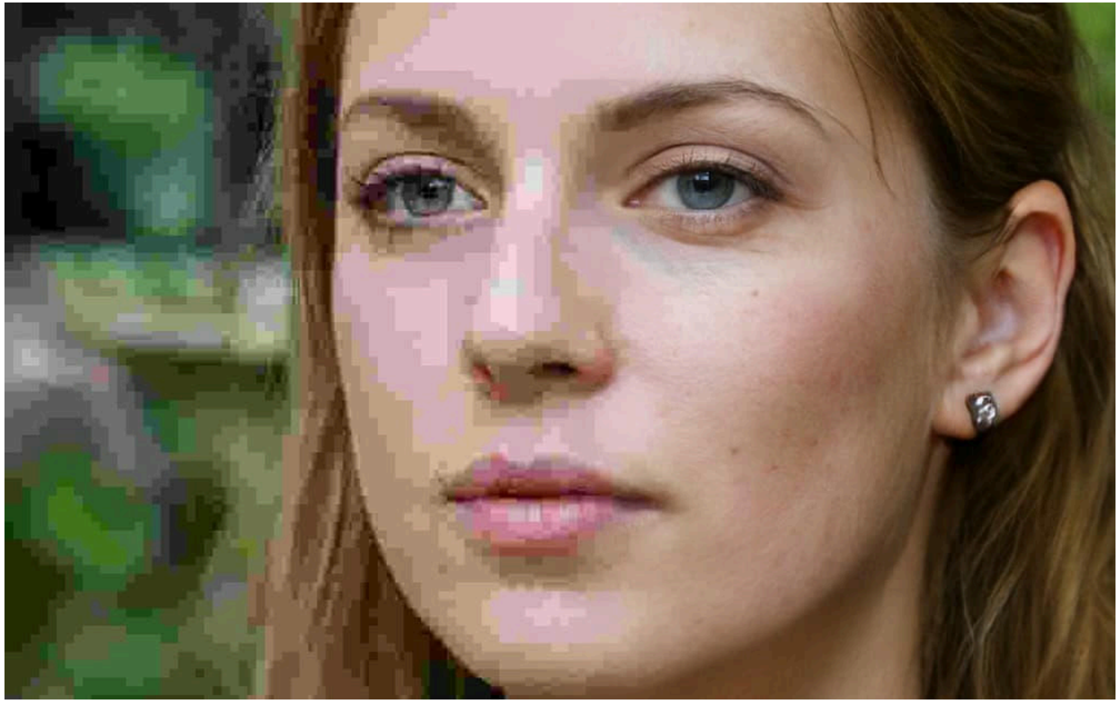
Compression

Compression may be **lossy** or **lossless**



Compression

Compression may be **lossy** or **lossless**



Question: Which is H.264?

Agenda

1. Review
2. **Compression**
3. Autoencoders
4. Applications
5. Variational Modeling
6. VAE Implementation
7. Coding

Agenda

1. Review
2. Compression
3. **Autoencoders**
4. Applications
5. Variational Modeling
6. VAE Implementation
7. Coding

Autoencoders

Encoders and decoders for images, videos, and music are functions

Autoencoders

Encoders and decoders for images, videos, and music are functions

Neural networks can represent any continuous function

Autoencoders

Encoders and decoders for images, videos, and music are functions

Neural networks can represent any continuous function

Let us learn neural network encoders and decoders

Autoencoders

Encoders and decoders for images, videos, and music are functions

Neural networks can represent any continuous function

Let us learn neural network encoders and decoders

$$f : X \times \Theta \mapsto Z$$

Autoencoders

Encoders and decoders for images, videos, and music are functions

Neural networks can represent any continuous function

Let us learn neural network encoders and decoders

$$f : X \times \Theta \mapsto Z$$

$$f^{-1} : Z \times \Theta \mapsto X$$

Autoencoders

Encoders and decoders for images, videos, and music are functions

Neural networks can represent any continuous function

Let us learn neural network encoders and decoders

$$f : X \times \Theta \mapsto Z$$

$$f^{-1} : Z \times \Theta \mapsto X$$

We call this an **autoencoder**

Autoencoders

Encoders and decoders for images, videos, and music are functions

Neural networks can represent any continuous function

Let us learn neural network encoders and decoders

$$f : X \times \Theta \mapsto Z$$

$$f^{-1} : Z \times \Theta \mapsto X$$

We call this an **autoencoder**

Notice there is no Y this time

Autoencoders

Encoders and decoders for images, videos, and music are functions

Neural networks can represent any continuous function

Let us learn neural network encoders and decoders

$$f : X \times \Theta \mapsto Z$$

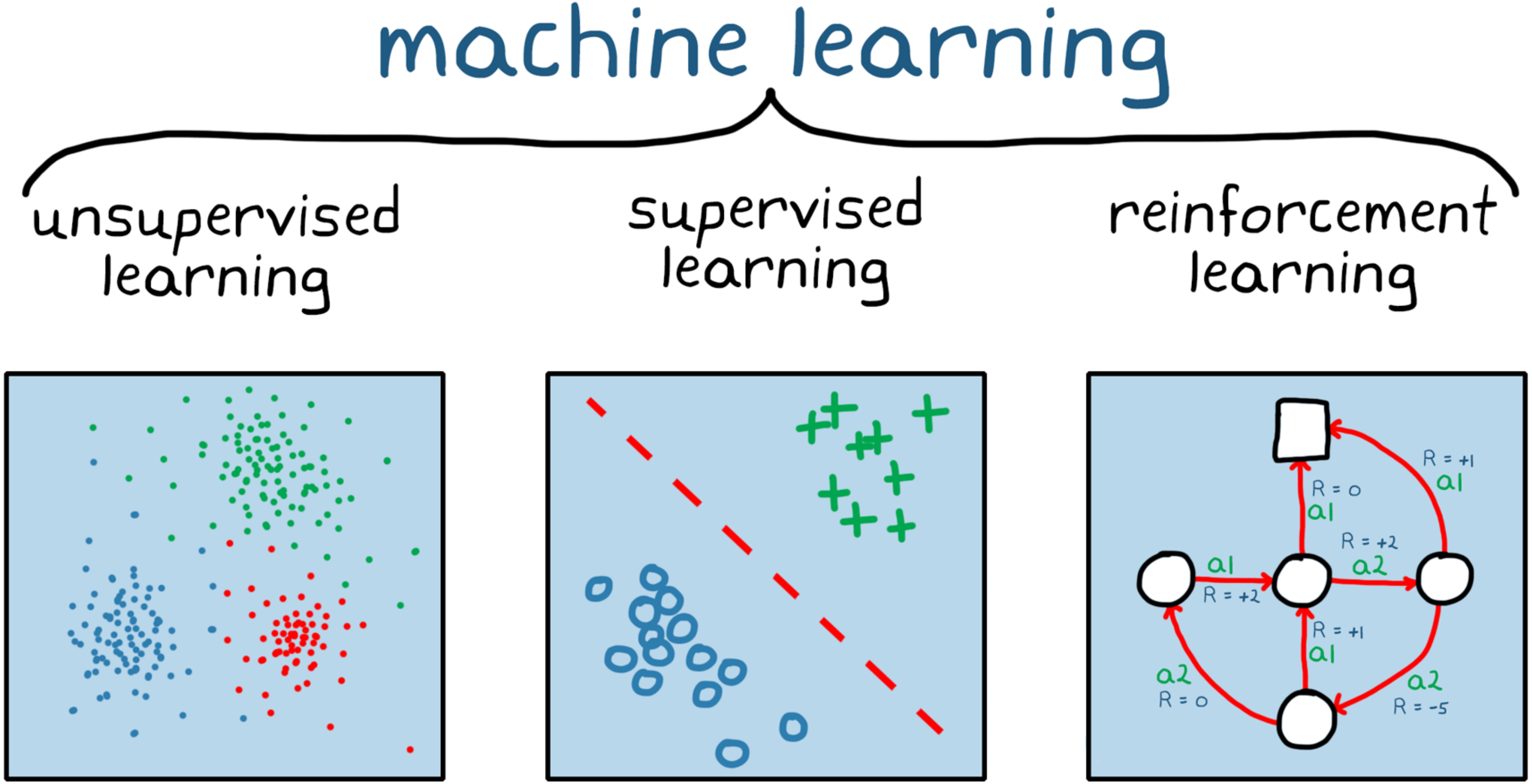
$$f^{-1} : Z \times \Theta \mapsto X$$

We call this an **autoencoder**

Notice there is no Y this time

Training autoencoders is different than what we have seen before

Autoencoders



Autoencoders

In supervised learning, humans provide the model with **inputs** $\boldsymbol{X}$ and corresponding **outputs** $\boldsymbol{Y}$

Autoencoders

In supervised learning, humans provide the model with **inputs** $\boldsymbol{X}$ and corresponding **outputs** $\boldsymbol{Y}$

$$\boldsymbol{X} = \begin{bmatrix} x_{[1]} & x_{[2]} & \dots & x_{[n]} \end{bmatrix}^\top \quad \boldsymbol{Y} = \begin{bmatrix} y_{[1]} & y_{[2]} & \dots & y_{[n]} \end{bmatrix}^\top$$

Autoencoders

In supervised learning, humans provide the model with **inputs** $\boldsymbol{X}$ and corresponding **outputs** $\boldsymbol{Y}$

$$\boldsymbol{X} = \begin{bmatrix} x_{[1]} & x_{[2]} & \dots & x_{[n]} \end{bmatrix}^\top \quad \boldsymbol{Y} = \begin{bmatrix} y_{[1]} & y_{[2]} & \dots & y_{[n]} \end{bmatrix}^\top$$

In unsupervised learning, humans only provide **input**

Autoencoders

In supervised learning, humans provide the model with **inputs** $\boldsymbol{X}$ and corresponding **outputs** $\boldsymbol{Y}$

$$\boldsymbol{X} = \begin{bmatrix} x_{[1]} & x_{[2]} & \dots & x_{[n]} \end{bmatrix}^\top \quad \boldsymbol{Y} = \begin{bmatrix} y_{[1]} & y_{[2]} & \dots & y_{[n]} \end{bmatrix}^\top$$

In unsupervised learning, humans only provide **input**

$$\boldsymbol{X} = \begin{bmatrix} x_{[1]} & x_{[2]} & \dots & x_{[n]} \end{bmatrix}^\top$$

Autoencoders

In supervised learning, humans provide the model with **inputs** $\boldsymbol{X}$ and corresponding **outputs** $\boldsymbol{Y}$

$$\boldsymbol{X} = \begin{bmatrix} x_{[1]} & x_{[2]} & \dots & x_{[n]} \end{bmatrix}^\top \quad \boldsymbol{Y} = \begin{bmatrix} y_{[1]} & y_{[2]} & \dots & y_{[n]} \end{bmatrix}^\top$$

In unsupervised learning, humans only provide **input**

$$\boldsymbol{X} = \begin{bmatrix} x_{[1]} & x_{[2]} & \dots & x_{[n]} \end{bmatrix}^\top$$

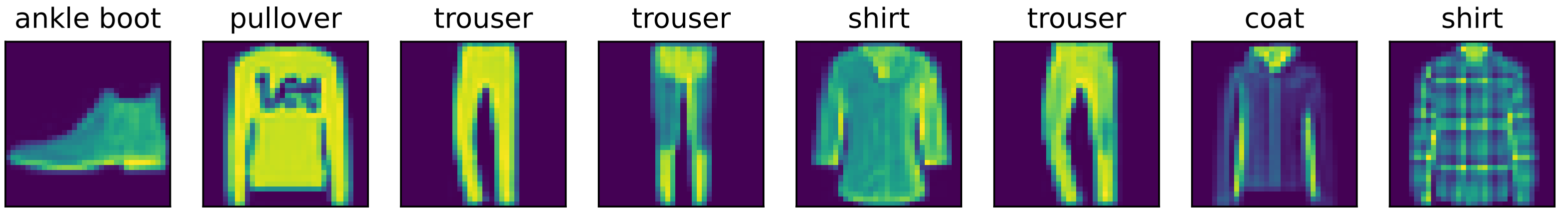
The training algorithm may generate labels

Autoencoders

Task: Compress images for your clothing website to save on costs

Autoencoders

Task: Compress images for your clothing website to save on costs

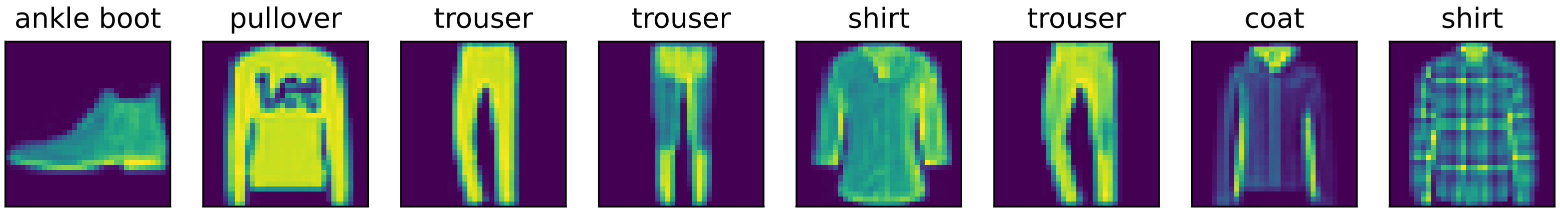


$$d_x : 28 \times 28 \qquad d_z : 4$$

$$X \in [0, 1]^{d_x} \qquad Z \in \mathbb{R}^{d_z}$$

Autoencoders

Task: Compress images for your clothing website to save on costs



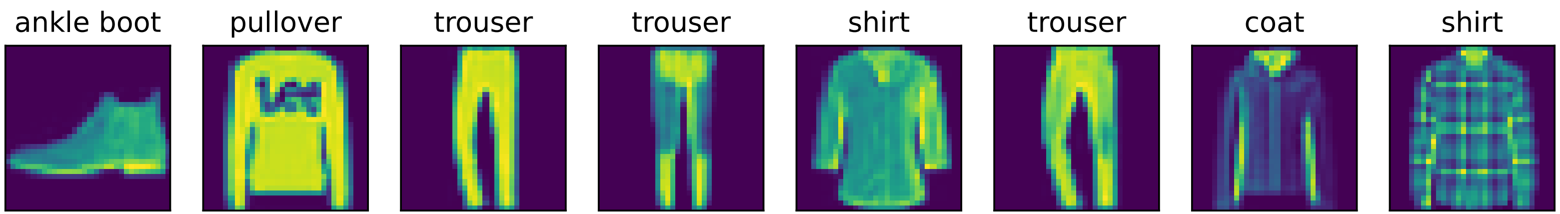
$$d_x : 28 \times 28 \qquad\qquad d_z : 4$$

$$X \in [0, 1]^{d_x} \qquad\qquad Z \in \mathbb{R}^{d_z}$$

$$f : X \times \Theta \mapsto Z \qquad\qquad f^{-1} : Z \times \Theta \mapsto X$$

Autoencoders

Task: Compress images for your clothing website to save on costs



$$d_x : 28 \times 28 \qquad d_z : 4$$

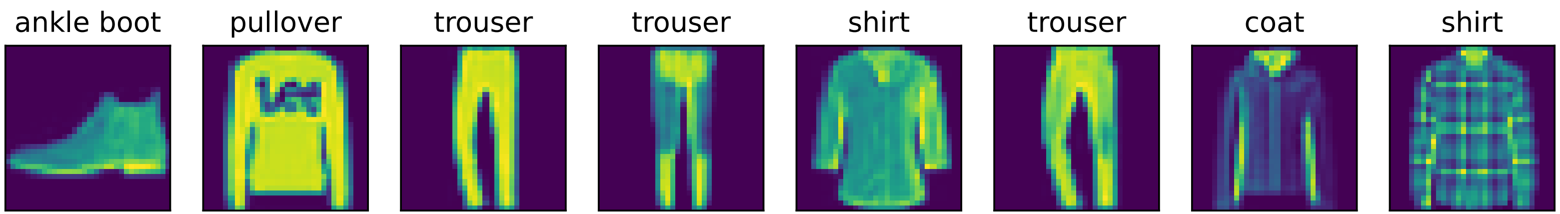
$$X \in [0, 1]^{d_x} \qquad Z \in \mathbb{R}^{d_z}$$

$$f : X \times \Theta \mapsto Z \qquad f^{-1} : Z \times \Theta \mapsto X$$

What is the structure of f, f^{-1}?

Autoencoders

Task: Compress images for your clothing website to save on costs



$d_x : 28 \times 28$

$d_z : 4$

$X \in [0, 1]^{d_x}$

$Z \in \mathbb{R}^{d_z}$

$f : X \times \Theta \mapsto Z$

$f^{-1} : Z \times \Theta \mapsto X$

What is the structure of f, f^{-1}?

How do we find $\boldsymbol{\theta}$?

Autoencoders

Let us find f, then find the inverse f^{-1}

Autoencoders

Let us find f, then find the inverse f^{-1}

Start with a perceptron

Autoencoders

Let us find f, then find the inverse f^{-1}

Start with a perceptron

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^\top \overline{\boldsymbol{x}}); \quad \boldsymbol{\theta} \in \mathbb{R}^{d_x \times d_z}$$

Autoencoders

Let us find f, then find the inverse f^{-1}

Start with a perceptron

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^\top \overline{\boldsymbol{x}}); \quad \boldsymbol{\theta} \in \mathbb{R}^{d_x \times d_z}$$

$$\boldsymbol{z} = \sigma(\boldsymbol{\theta}^\top \overline{\boldsymbol{x}})$$

Autoencoders

Let us find f, then find the inverse f^{-1}

Start with a perceptron

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^\top \overline{\boldsymbol{x}}); \quad \boldsymbol{\theta} \in \mathbb{R}^{d_x \times d_z}$$

$$\boldsymbol{z} = \sigma(\boldsymbol{\theta}^\top \overline{\boldsymbol{x}})$$

Solve for $\boldsymbol{x}$ to find the inverse

Autoencoders

Let us find f, then find the inverse f^{-1}

Start with a perceptron

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^\top \overline{\boldsymbol{x}}); \quad \boldsymbol{\theta} \in \mathbb{R}^{d_x \times d_z}$$

$$\boldsymbol{z} = \sigma(\boldsymbol{\theta}^\top \overline{\boldsymbol{x}})$$

Solve for $\boldsymbol{x}$ to find the inverse

$$\sigma^{-1}(\boldsymbol{z}) = \sigma^{-1}(\sigma(\boldsymbol{\theta}^\top \overline{\boldsymbol{x}}))$$

Autoencoders

Let us find f, then find the inverse f^{-1}

Start with a perceptron

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^\top \overline{\boldsymbol{x}}); \quad \boldsymbol{\theta} \in \mathbb{R}^{d_x \times d_z}$$

$$\boldsymbol{z} = \sigma(\boldsymbol{\theta}^\top \overline{\boldsymbol{x}})$$

Solve for $\boldsymbol{x}$ to find the inverse

$$\sigma^{-1}(\boldsymbol{z}) = \sigma^{-1}(\sigma(\boldsymbol{\theta}^\top \overline{\boldsymbol{x}}))$$

$$\sigma^{-1}(\boldsymbol{z}) = \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

Autoencoders

$$\sigma^{-1}(\boldsymbol{z}) = \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

Autoencoders

$$\sigma^{-1}(\boldsymbol{z}) = \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

$$\left(\boldsymbol{\theta}^\top\right)^{-1} \sigma^{-1}(\boldsymbol{z}) = \left(\boldsymbol{\theta}^\top\right)^{-1} \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

Autoencoders

$$\sigma^{-1}(\boldsymbol{z}) = \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

$$\left(\boldsymbol{\theta}^\top\right)^{-1} \sigma^{-1}(\boldsymbol{z}) = \left(\boldsymbol{\theta}^\top\right)^{-1} \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

$$\left(\boldsymbol{\theta}^\top\right)^{-1} \sigma^{-1}(\boldsymbol{z}) = \boldsymbol{I} \overline{\boldsymbol{x}}$$

Autoencoders

$$\sigma^{-1}(\boldsymbol{z}) = \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

$$\left(\boldsymbol{\theta}^\top\right)^{-1} \sigma^{-1}(\boldsymbol{z}) = \left(\boldsymbol{\theta}^\top\right)^{-1} \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

$$\left(\boldsymbol{\theta}^\top\right)^{-1} \sigma^{-1}(\boldsymbol{z}) = \boldsymbol{I} \overline{\boldsymbol{x}}$$

$$\left(\boldsymbol{\theta}^\top\right)^{-1} \sigma^{-1}(\boldsymbol{z}) = \overline{\boldsymbol{x}}$$

Autoencoders

$$\sigma^{-1}(\boldsymbol{z}) = \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

$$\left(\boldsymbol{\theta}^\top\right)^{-1} \sigma^{-1}(\boldsymbol{z}) = \left(\boldsymbol{\theta}^\top\right)^{-1} \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

$$\left(\boldsymbol{\theta}^\top\right)^{-1} \sigma^{-1}(\boldsymbol{z}) = \boldsymbol{I} \overline{\boldsymbol{x}}$$

$$\left(\boldsymbol{\theta}^\top\right)^{-1} \sigma^{-1}(\boldsymbol{z}) = \overline{\boldsymbol{x}}$$

$$f^{-1}(\boldsymbol{z}, \boldsymbol{\theta}) = \left(\boldsymbol{\theta}^\top\right)^{-1} \sigma^{-1}(\boldsymbol{z})$$

Autoencoders

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^\top \overline{\boldsymbol{x}}) \quad f^{-1}(\boldsymbol{z}, \boldsymbol{\theta}) = \left(\boldsymbol{\theta}^\top\right)^{-1} \sigma^{-1}(\boldsymbol{z}) \qquad \boldsymbol{\theta} \in \mathbb{R}^{d_x \times d_z}$$

Autoencoders

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^\top \overline{\boldsymbol{x}}) \quad f^{-1}(\boldsymbol{z}, \boldsymbol{\theta}) = \left(\boldsymbol{\theta}^\top\right)^{-1} \sigma^{-1}(\boldsymbol{z}) \qquad \boldsymbol{\theta} \in \mathbb{R}^{d_x \times d_z}$$

Question: Any issues?

Hint: What if $d_x \neq d_z$?

Autoencoders

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^\top \overline{\boldsymbol{x}}) \quad f^{-1}(\boldsymbol{z}, \boldsymbol{\theta}) = (\boldsymbol{\theta}^\top)^{-1} \sigma^{-1}(\boldsymbol{z}) \qquad \boldsymbol{\theta} \in \mathbb{R}^{d_x \times d_z}$$

Question: Any issues?

Hint: What if $d_x \neq d_z$?

Answer: Can only invert square matrices, $\boldsymbol{\theta}^\top$ only invertible if $d_z = d_x$

Autoencoders

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^\top \overline{\boldsymbol{x}}) \quad f^{-1}(\boldsymbol{z}, \boldsymbol{\theta}) = \left(\boldsymbol{\theta}^\top\right)^{-1} \sigma^{-1}(\boldsymbol{z}) \qquad \boldsymbol{\theta} \in \mathbb{R}^{d_x \times d_z}$$

Question: Any issues? **Hint:** What if $d_x \neq d_z$?

Answer: Can only invert square matrices, $\boldsymbol{\theta}^\top$ only invertible if $d_z = d_x$

Question: What kind of compression can we achieve if $d_z = d_x$?

Autoencoders

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^\top \overline{\boldsymbol{x}}) \quad f^{-1}(\boldsymbol{z}, \boldsymbol{\theta}) = \left(\boldsymbol{\theta}^\top\right)^{-1} \sigma^{-1}(\boldsymbol{z}) \qquad \boldsymbol{\theta} \in \mathbb{R}^{d_x \times d_z}$$

Question: Any issues? **Hint:** What if $d_x \neq d_z$?

Answer: Can only invert square matrices, $\boldsymbol{\theta}^\top$ only invertible if $d_z = d_x$

Question: What kind of compression can we achieve if $d_z = d_x$?

Answer: None! We need $d_z < d_x$ for compression

Autoencoders

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^\top \overline{\boldsymbol{x}}) \quad f^{-1}(\boldsymbol{z}, \boldsymbol{\theta}) = (\boldsymbol{\theta}^\top)^{-1} \sigma^{-1}(\boldsymbol{z}) \qquad \boldsymbol{\theta} \in \mathbb{R}^{d_x \times d_z}$$

Question: Any issues? **Hint:** What if $d_x \neq d_z$?

Answer: Can only invert square matrices, $\boldsymbol{\theta}^\top$ only invertible if $d_z = d_x$

Question: What kind of compression can we achieve if $d_z = d_x$?

Answer: None! We need $d_z < d_x$ for compression

Look for another solution

Autoencoders

Let us try another way

Autoencoders

Let us try another way

$$\boldsymbol{z} = f(\boldsymbol{x}, \boldsymbol{\theta}_e) = \sigma\left(\boldsymbol{\theta}_e^\top \overline{\boldsymbol{x}}\right)$$

$$\boldsymbol{x} = f^{-1}(\boldsymbol{z}, \boldsymbol{\theta}_d) = \sigma\left(\boldsymbol{\theta}_d^\top \overline{\boldsymbol{z}}\right)$$

Autoencoders

Let us try another way

$$\boldsymbol{z} = f(\boldsymbol{x}, \boldsymbol{\theta}_e) = \sigma(\boldsymbol{\theta}_e^\top \overline{\boldsymbol{x}})$$

$$\boldsymbol{x} = f^{-1}(\boldsymbol{z}, \boldsymbol{\theta}_d) = \sigma(\boldsymbol{\theta}_d^\top \overline{\boldsymbol{z}})$$

What if we plug $\boldsymbol{z}$ into the second equation?

Autoencoders

Let us try another way

$$\boldsymbol{z} = f(\boldsymbol{x}, \boldsymbol{\theta}_e) = \sigma(\boldsymbol{\theta}_e^\top \overline{\boldsymbol{x}})$$

$$\boldsymbol{x} = f^{-1}(\boldsymbol{z}, \boldsymbol{\theta}_d) = \sigma(\boldsymbol{\theta}_d^\top \overline{\boldsymbol{z}})$$

What if we plug $\boldsymbol{z}$ into the second equation?

$$\boldsymbol{x} = f^{-1}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), \boldsymbol{\theta}_d) = \sigma(\boldsymbol{\theta}_d^\top \boldsymbol{\sigma}(\boldsymbol{\theta}_e^\top \overline{\boldsymbol{x}}))$$

Autoencoders

$$\boldsymbol{x} = f^{-1}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), \boldsymbol{\theta}_d) = \sigma\left(\boldsymbol{\theta}_d^\top \boldsymbol{\sigma}\left(\boldsymbol{\theta}_{\boldsymbol{e}}^\top \overline{\boldsymbol{x}}\right)\right)$$

Autoencoders

$$\boldsymbol{x} = f^{-1}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), \boldsymbol{\theta}_d) = \sigma\left(\boldsymbol{\theta}_d^\top \boldsymbol{\sigma}(\boldsymbol{\theta}_{\boldsymbol{e}}^\top \overline{\boldsymbol{x}})\right)$$

More generally, f, f^{-1} may be any neural network

Autoencoders

$$\boldsymbol{x} = f^{-1}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), \boldsymbol{\theta}_d) = \sigma\left(\boldsymbol{\theta}_d^\top \boldsymbol{\sigma}\left(\boldsymbol{\theta}_{\boldsymbol{e}}^\top \overline{\boldsymbol{x}}\right)\right)$$

More generally, f, f^{-1} may be any neural network

$$\boldsymbol{x} = f^{-1}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), \boldsymbol{\theta}_d)$$

Autoencoders

$$\boldsymbol{x} = f^{-1}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), \boldsymbol{\theta}_d) = \sigma\left(\boldsymbol{\theta}_d^\top \boldsymbol{\sigma}(\boldsymbol{\theta}_{\boldsymbol{e}}^\top \overline{\boldsymbol{x}})\right)$$

More generally, f, f^{-1} may be any neural network

$$\boldsymbol{x} = f^{-1}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), \boldsymbol{\theta}_d)$$

Turn this into a loss function using the square error

Autoencoders

$$\boldsymbol{x} = f^{-1}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), \boldsymbol{\theta}_d) = \sigma\left(\boldsymbol{\theta}_d^\top \boldsymbol{\sigma}\left(\boldsymbol{\theta}_{\boldsymbol{e}}^\top \overline{\boldsymbol{x}}\right)\right)$$

More generally, f, f^{-1} may be any neural network

$$\boldsymbol{x} = f^{-1}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), \boldsymbol{\theta}_d)$$

Turn this into a loss function using the square error

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{j=1}^{d_x} \left(x_j - f^{-1}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), \boldsymbol{\theta}_d)_j\right)^2$$

Autoencoders

$$\boldsymbol{x} = f^{-1}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), \boldsymbol{\theta}_d) = \sigma\left(\boldsymbol{\theta}_d^\top \boldsymbol{\sigma}\left(\boldsymbol{\theta}_{\boldsymbol{e}}^\top \overline{\boldsymbol{x}}\right)\right)$$

More generally, f, f^{-1} may be any neural network

$$\boldsymbol{x} = f^{-1}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), \boldsymbol{\theta}_d)$$

Turn this into a loss function using the square error

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{j=1}^{d_x} \left(x_j - f^{-1}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), \boldsymbol{\theta}_d)_j \right)^2$$

Forces the networks to compress and reconstruct $\boldsymbol{x}$

Autoencoders

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{j=1}^{d_x} \left(x_j - f^{-1}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), \boldsymbol{\theta}_d)_j \right)^2$$

Autoencoders

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{j=1}^{d_x} \left(x_j - f^{-1}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), \boldsymbol{\theta}_d)_j \right)^2$$

Define over the entire dataset

$$\mathcal{L}(\boldsymbol{X}, \boldsymbol{\theta}) = \sum_{i=1}^{n} \sum_{j=1}^{d_x} \left(x_{[i],j} - f^{-1}\left(f\left(\boldsymbol{x}_{[i]}, \boldsymbol{\theta}_e\right), \boldsymbol{\theta}_d\right)_j \right)^2$$

Autoencoders

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{j=1}^{d_x} \left(x_j - f^{-1}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), \boldsymbol{\theta}_d)_j \right)^2$$

Define over the entire dataset

$$\mathcal{L}(\boldsymbol{X}, \boldsymbol{\theta}) = \sum_{i=1}^{n} \sum_{j=1}^{d_x} \left(x_{[i],j} - f^{-1}\left(f\left(\boldsymbol{x}_{[i]}, \boldsymbol{\theta}_e\right), \boldsymbol{\theta}_d\right)_j \right)^2$$

We call this the **reconstruction loss**

Autoencoders

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{j=1}^{d_x} \left(x_j - f^{-1}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), \boldsymbol{\theta}_d)_j \right)^2$$

Define over the entire dataset

$$\mathcal{L}(\boldsymbol{X}, \boldsymbol{\theta}) = \sum_{i=1}^{n} \sum_{j=1}^{d_x} \left(x_{[i],j} - f^{-1}\left(f\left(\boldsymbol{x}_{[i]}, \boldsymbol{\theta}_e\right), \boldsymbol{\theta}_d\right)_j \right)^2$$

We call this the **reconstruction loss**

It is an unsupervised loss because we only provide $\boldsymbol{X}$ and not $\boldsymbol{Y}$!

Autoencoders

First coding exercise

https://colab.research.google.com/drive/1UyR_W6NDIujaJXYlHZh6O3NfaCAMscpH#scrollTo=nmyQ8aE2pSbb

https://www.youtube.com/watch?v=UZDiGooFs54

Agenda

1. Review
2. Compression
3. **Autoencoders**
4. Applications
5. Variational Modeling
6. VAE Implementation
7. Coding

Agenda

1. Review
2. Compression
3. Autoencoders
4. **Applications**
5. Variational Modeling
6. VAE Implementation
7. Coding

Applications

We can use autoencoders for more than compression

Applications

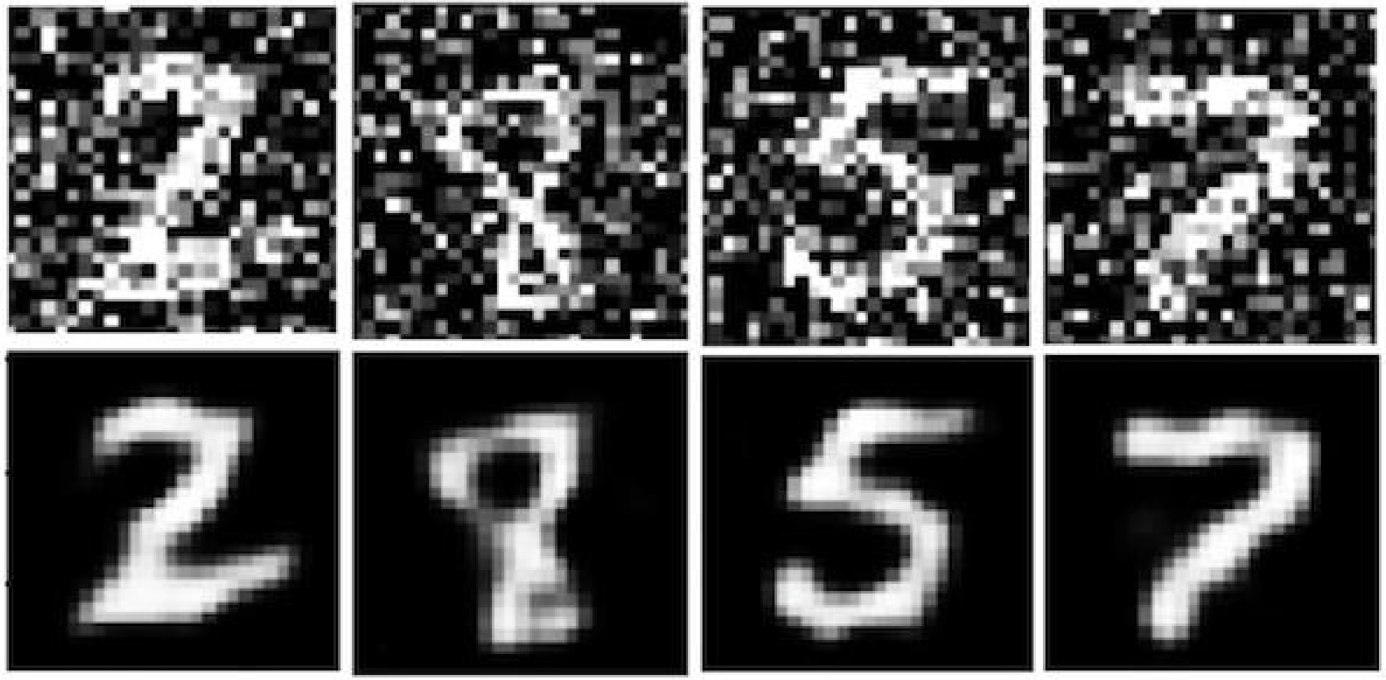
We can use autoencoders for more than compression

We can make **denoising autoencoders** that remove noise

Applications

We can use autoencoders for more than compression

We can make **denoising autoencoders** that remove noise



Applications

Generate some noise $\qquad \varepsilon \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

Applications

Generate some noise $\varepsilon \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

Add noise to the image $\boldsymbol{x} + \varepsilon$

Applications

Generate some noise $\varepsilon \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

Add noise to the image $\boldsymbol{x} + \varepsilon$

Original loss $\mathcal{L}(\boldsymbol{X}, \boldsymbol{\theta}) = \sum_{i=1}^{n} \sum_{j=1}^{d_x} \left(x_{[i],j} - f^{-1}\left(f\left(\boldsymbol{x}_{[i]}, \boldsymbol{\theta}_e \right), \boldsymbol{\theta}_d \right)_j \right)^2$

Applications

Generate some noise $\qquad \boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

Add noise to the image $\qquad \boldsymbol{x} + \boldsymbol{\varepsilon}$

Original loss $\qquad \mathcal{L}(\boldsymbol{X}, \boldsymbol{\theta}) = \sum_{i=1}^{n} \sum_{j=1}^{d_x} \left(x_{[i],j} - f^{-1}\left(f\left(\boldsymbol{x}_{[i]}, \boldsymbol{\theta}_e\right), \boldsymbol{\theta}_d\right)_j \right)^2$

Denoising loss $\qquad \mathcal{L}(\boldsymbol{X}, \boldsymbol{\theta}) = \sum_{i=1}^{n} \sum_{j=1}^{d_x} \left(x_{[i],j} - f^{-1}\left(f\left(\boldsymbol{x}_{[i]} + \boldsymbol{\varepsilon}, \boldsymbol{\theta}_e\right), \boldsymbol{\theta}_d\right)_j \right)^2$

Applications

Generate some noise $\varepsilon \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

Add noise to the image $\boldsymbol{x} + \varepsilon$

Original loss $$\mathcal{L}(\boldsymbol{X}, \boldsymbol{\theta}) = \sum_{i=1}^{n} \sum_{j=1}^{d_x} \left(x_{[i],j} - f^{-1}\left(f\left(\boldsymbol{x}_{[i]}, \boldsymbol{\theta}_e \right), \boldsymbol{\theta}_d \right)_j \right)^2$$

Denoising loss $$\mathcal{L}(\boldsymbol{X}, \boldsymbol{\theta}) = \sum_{i=1}^{n} \sum_{j=1}^{d_x} \left(x_{[i],j} - f^{-1}\left(f\left(\boldsymbol{x}_{[i]} + \varepsilon, \boldsymbol{\theta}_e \right), \boldsymbol{\theta}_d \right)_j \right)^2$$

Autoencoder will learn to remove noise when reconstructing image

Applications

We can add camera blur too

Applications

We can add camera blur too

Applications

$$\mathrm{blur}(\boldsymbol{x} + \boldsymbol{\varepsilon})$$

Applications

$$\text{blur}(\boldsymbol{x} + \boldsymbol{\varepsilon})$$

Denoising deblur loss

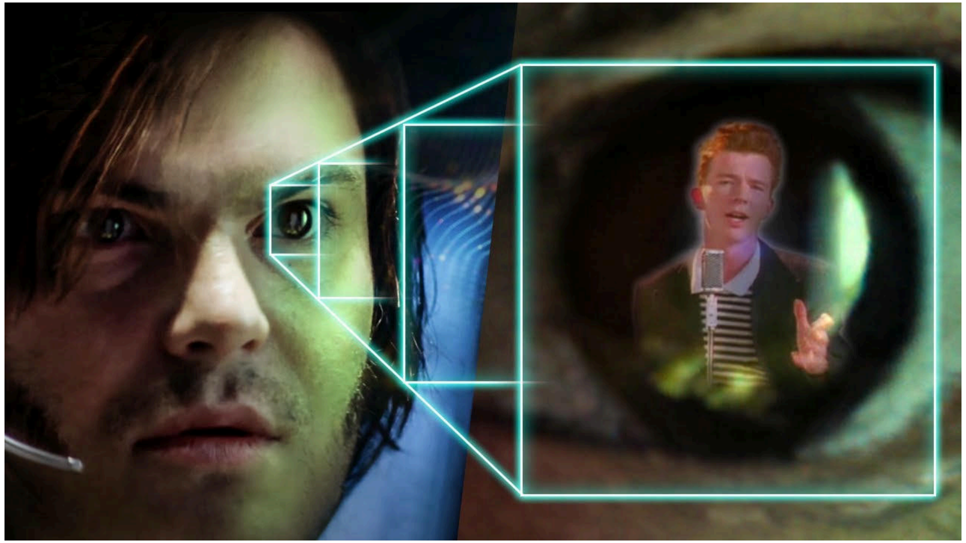
$$\mathcal{L}(\boldsymbol{X}, \boldsymbol{\theta}) = \sum_{i=1}^{n} \sum_{j=1}^{d_x} \left(x_{[i],j} - f^{-1}\left(f\left(\text{blur}\left(\boldsymbol{x}_{[i]} + \boldsymbol{\varepsilon}\right), \boldsymbol{\theta}_e\right), \boldsymbol{\theta}_d\right)_j \right)^2$$

Applications

Now we can "enhance" images like in crime tv shows

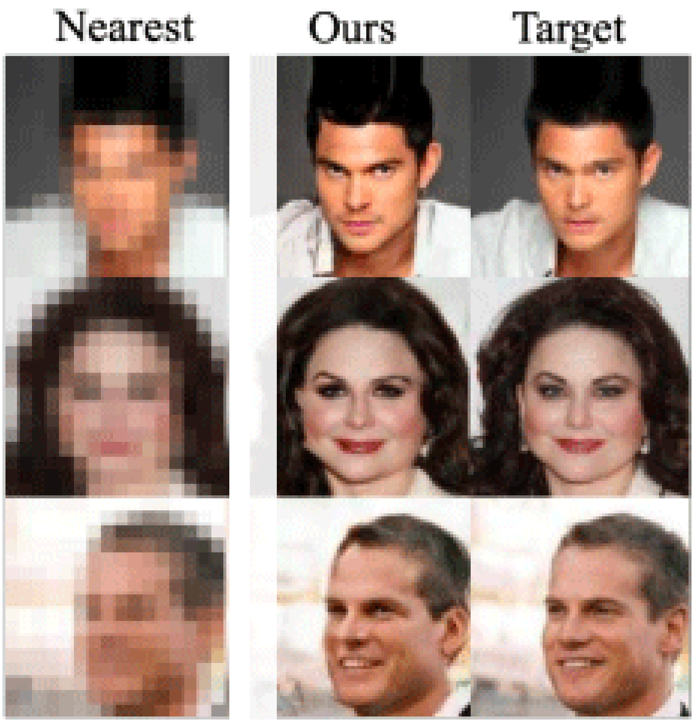
Applications

Now we can "enhance" images like in crime tv shows



Applications

We can deblur faces from security cameras



Applications

We can even hide parts of the image

Applications

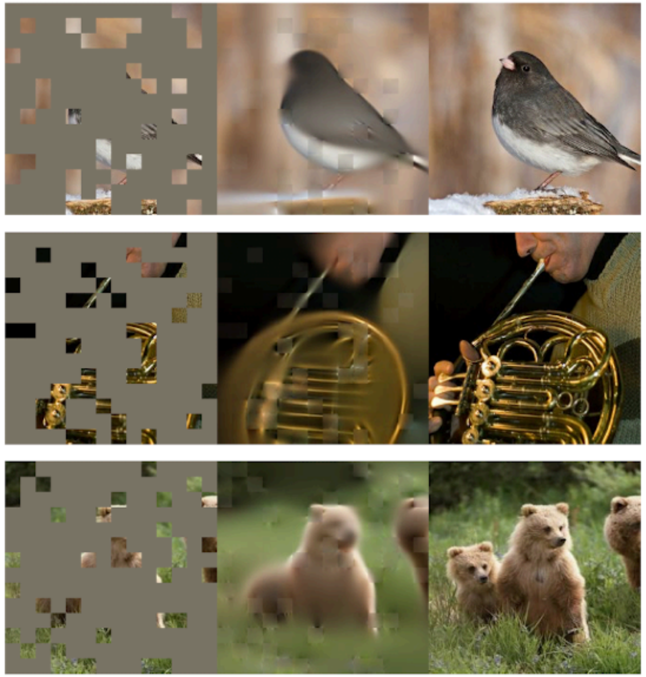
We can even hide parts of the image

A **masked autoencoder** will reconstruct the missing data

Applications

We can even hide parts of the image

A **masked autoencoder** will reconstruct the missing data



Applications

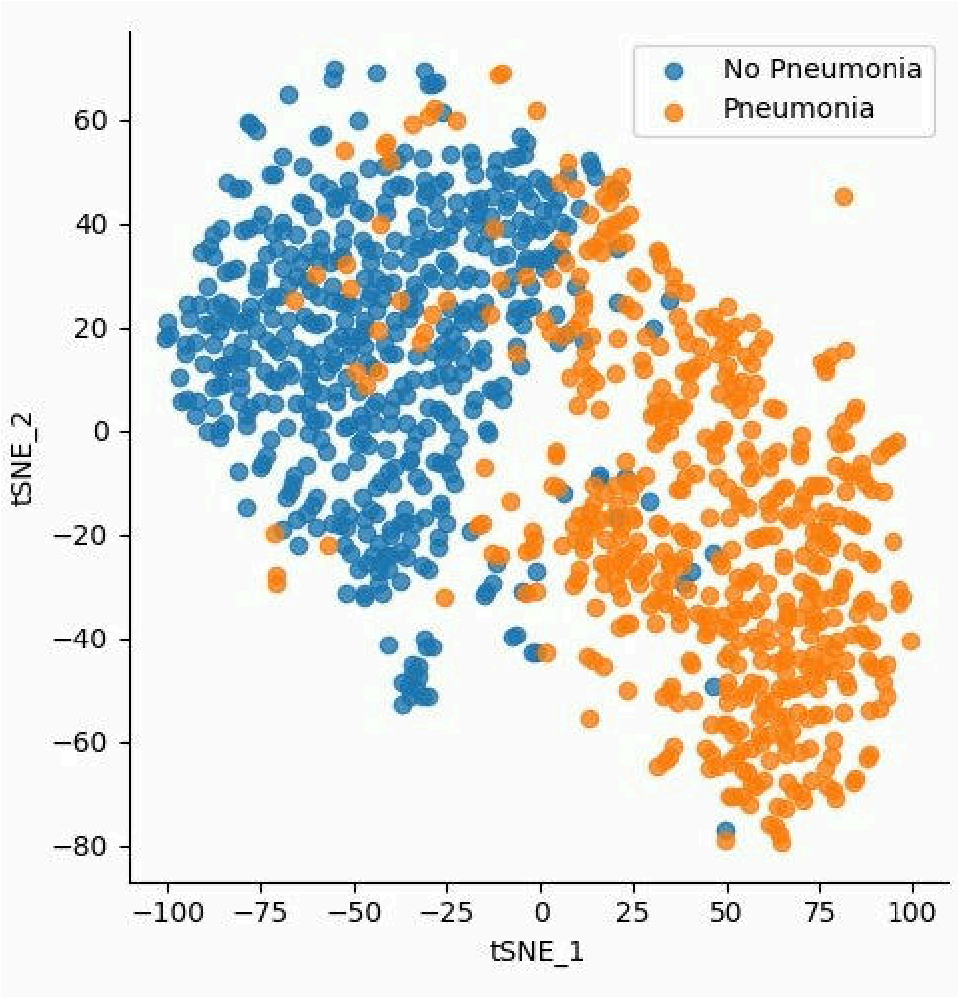
What is happening here? How can these models do this?

Applications

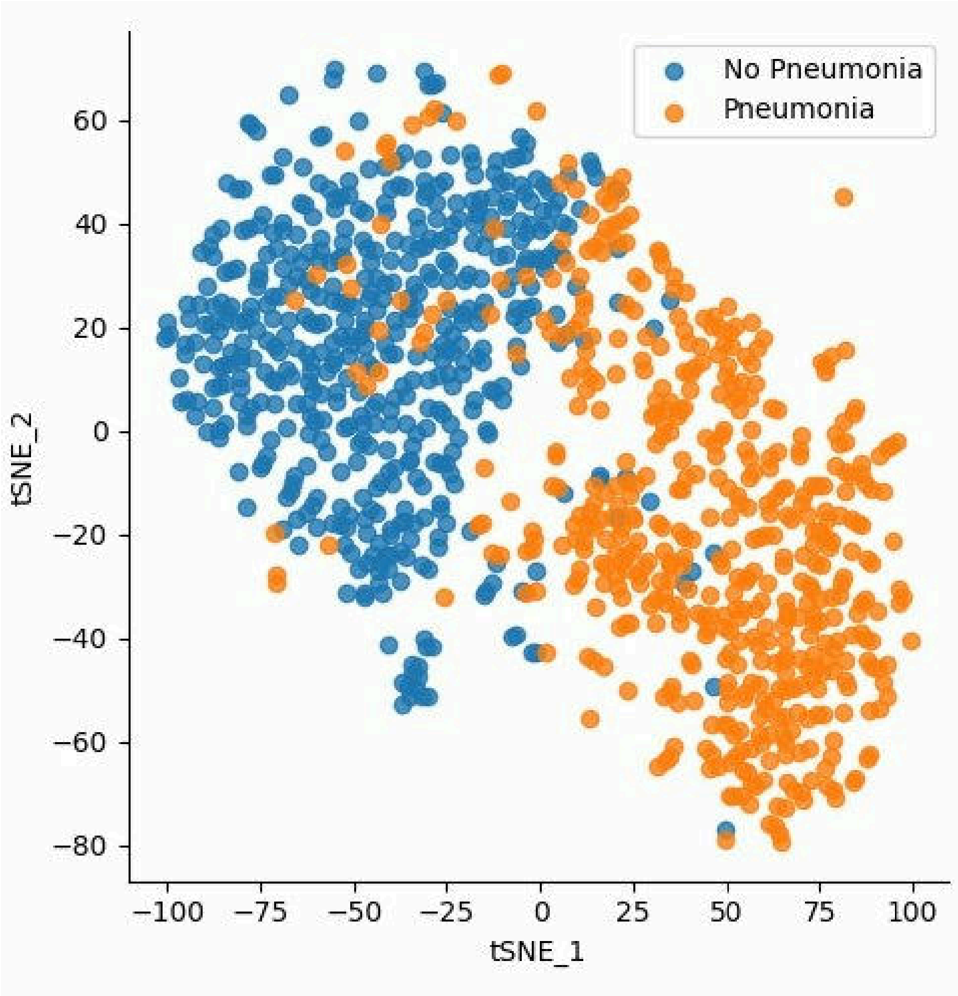
What is happening here? How can these models do this?

Autoencoders learn the structure of the dataset

Applications

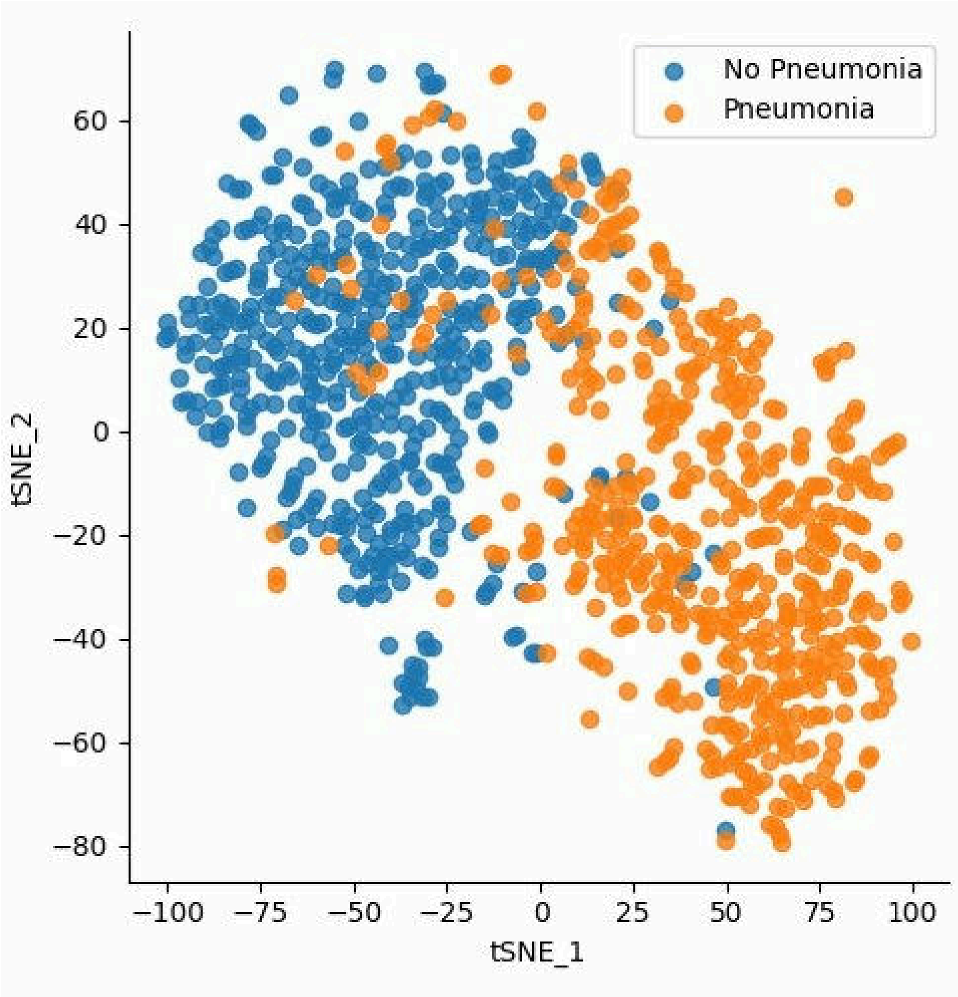


Applications



X : Pictures of human lungs

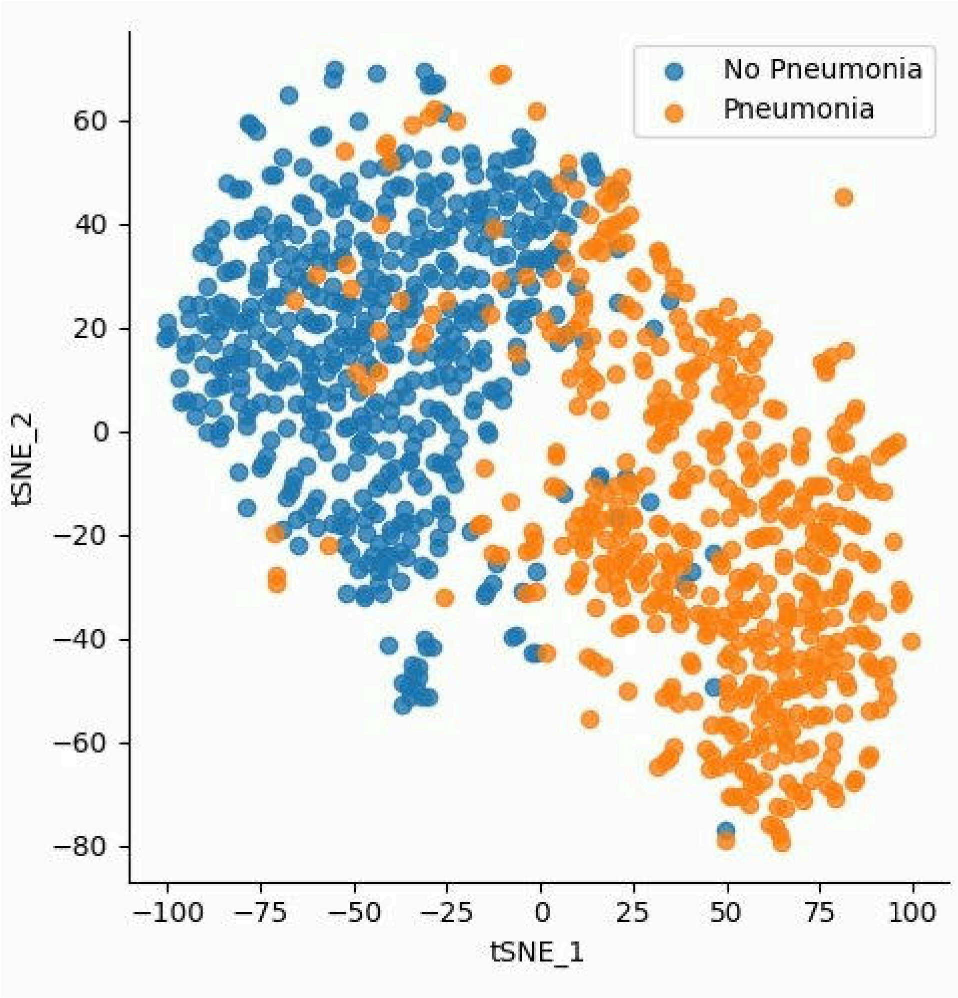
Applications



X : Pictures of human lungs

$Z \in \mathbb{R}^2$

Applications

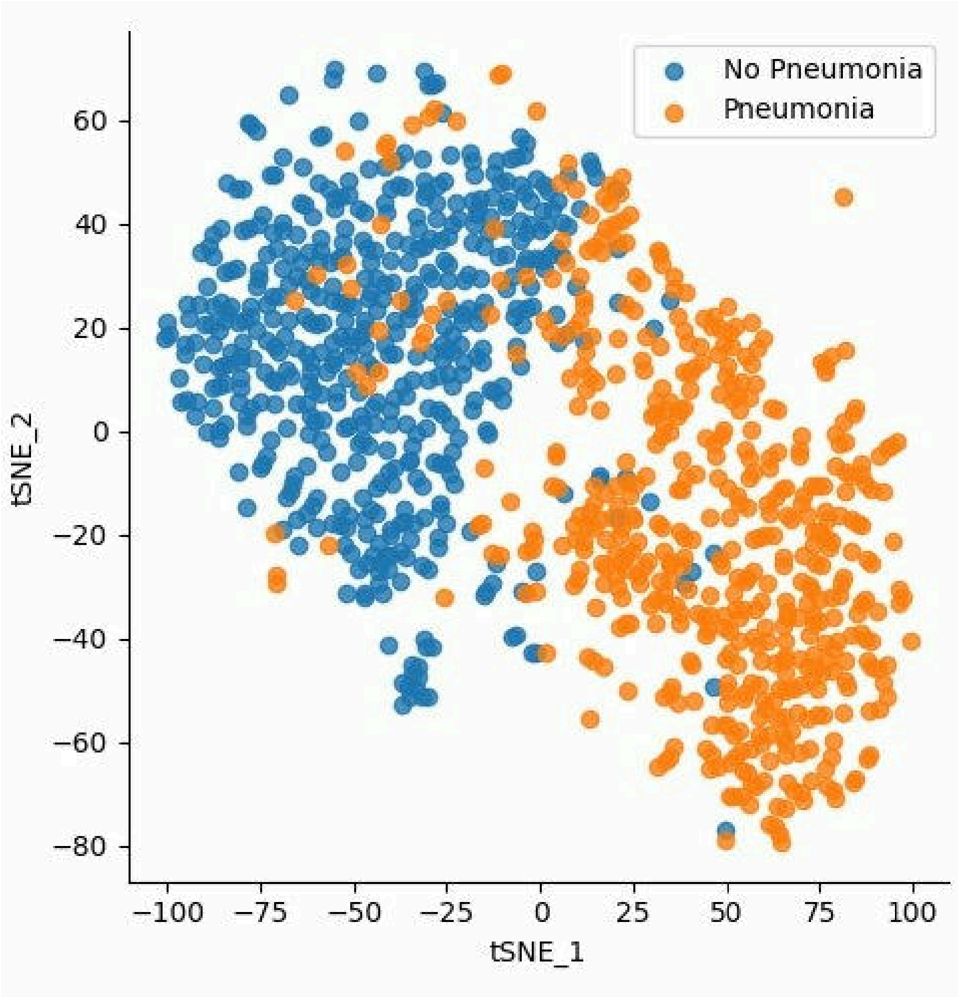


X : Pictures of human lungs

$Z \in \mathbb{R}^2$

Learns the structure of lungs from images

Applications



X : Pictures of human lungs

$Z \in \mathbb{R}^2$

Learns the structure of lungs from images

Differentiates sick and healthy lungs without being told

Applications

If the dataset is lung images, the model learns the structure of lungs

Applications

If the dataset is lung images, the model learns the structure of lungs

If the dataset is pictures from our world, then the autoencoders learn the structure of the world

Applications

If the dataset is lung images, the model learns the structure of lungs

If the dataset is pictures from our world, then the autoencoders learn the structure of the world

Nobody tells them what a dog or cat is

Applications

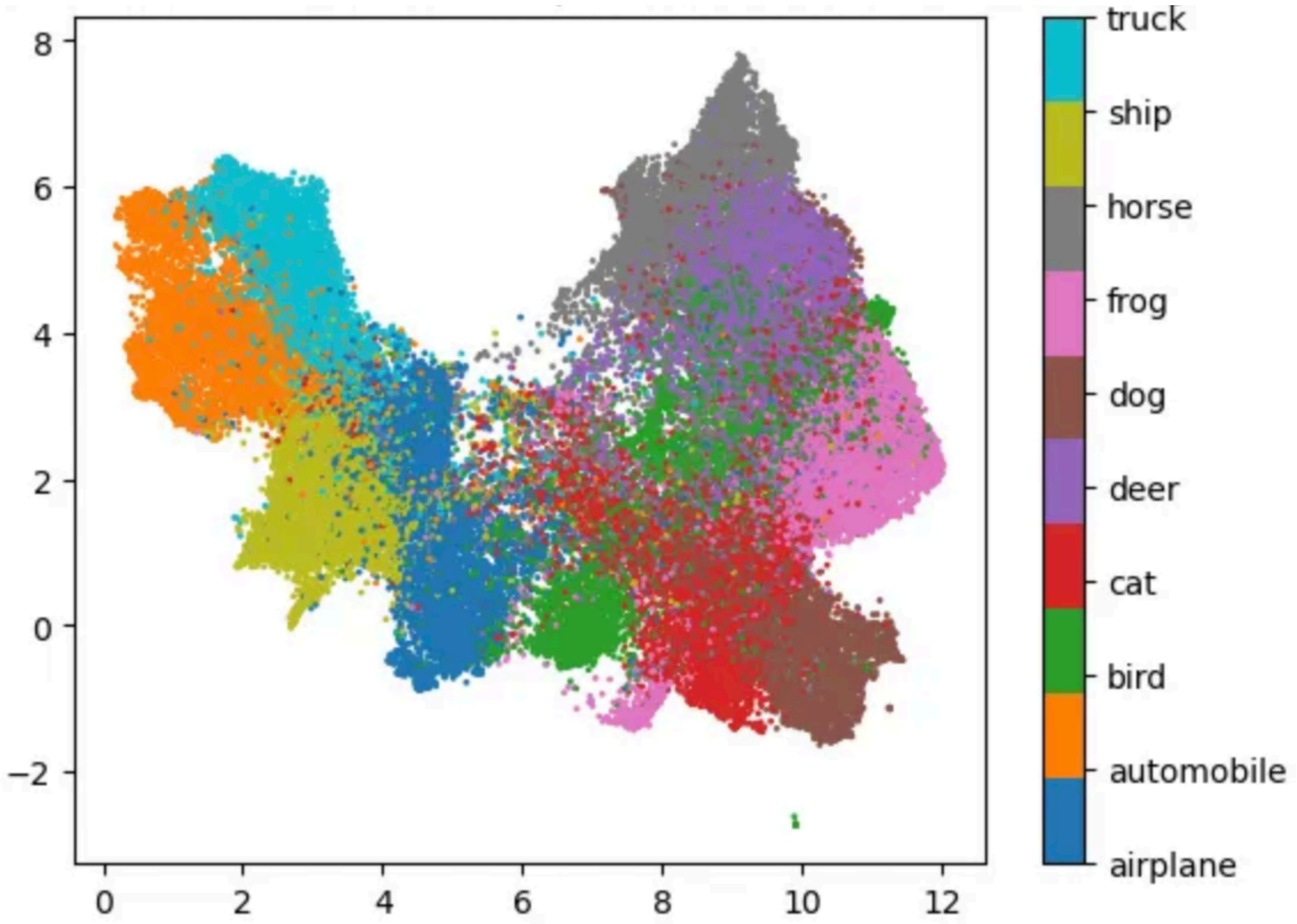
If the dataset is lung images, the model learns the structure of lungs

If the dataset is pictures from our world, then the autoencoders learn the structure of the world

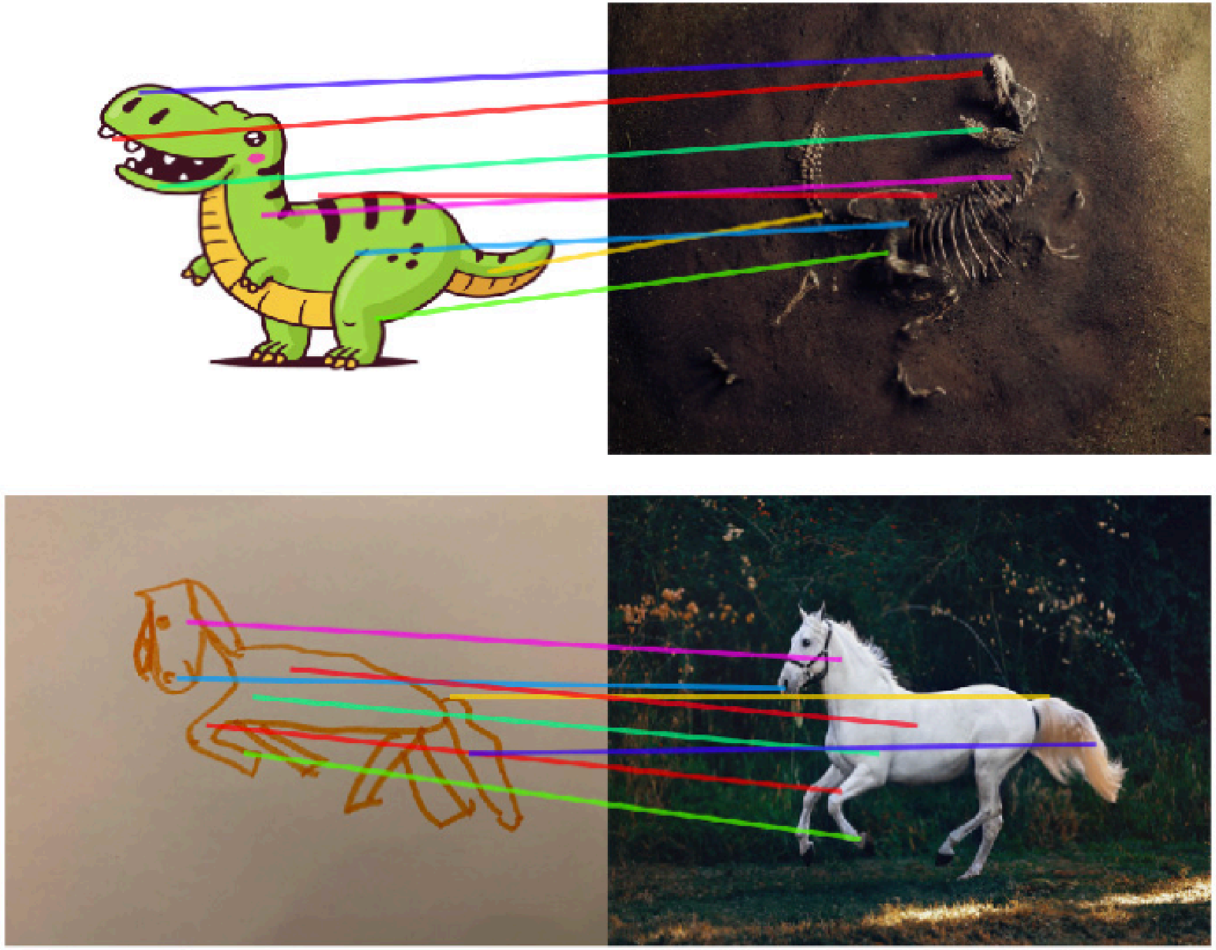
Nobody tells them what a dog or cat is

They learn that on their own

Applications



Applications



Applications

Some say “neural networks do not understand”, they just learn patterns

Applications

Some say “neural networks do not understand”, they just learn patterns

Humans are also pattern recognition machines

Applications

Some say "neural networks do not understand", they just learn patterns

Humans are also pattern recognition machines

Applications

Some say "neural networks do not understand", they just learn patterns

Humans are also pattern recognition machines

These networks can understand our world

Agenda

1. Review
2. Compression
3. Autoencoders
4. **Applications**
5. Variational Modeling
6. VAE Implementation
7. Coding

Agenda

1. Review
2. Compression
3. Autoencoders
4. Applications
5. **Variational Modeling**
6. VAE Implementation
7. Coding

Variational Modeling

Variational Modeling

These pictures were created by a **variational** autoencoder

Variational Modeling

These pictures were created by a **variational** autoencoder

But these people do not exist!

Variational Modeling

Autoencoders are useful for compression and denoising

Variational Modeling

Autoencoders are useful for compression and denoising

But we can also use them as **generative models**

Variational Modeling

Autoencoders are useful for compression and denoising

But we can also use them as **generative models**

A generative model learns the structure of data

Variational Modeling

Autoencoders are useful for compression and denoising

But we can also use them as **generative models**

A generative model learns the structure of data

Using this structure, it generates **new** data

Variational Modeling

Autoencoders are useful for compression and denoising

But we can also use them as **generative models**

A generative model learns the structure of data

Using this structure, it generates **new** data

- Train on face dataset, generate **new** pictures

Variational Modeling

Autoencoders are useful for compression and denoising

But we can also use them as **generative models**

A generative model learns the structure of data

Using this structure, it generates **new** data

- Train on face dataset, generate **new** pictures
- Train on book dataset, write a **new** book

Variational Modeling

Autoencoders are useful for compression and denoising

But we can also use them as **generative models**

A generative model learns the structure of data

Using this structure, it generates **new** data

- Train on face dataset, generate **new** pictures
- Train on book dataset, write a **new** book
- Train on protein dataset, create **new** proteins

Variational Modeling

Autoencoders are useful for compression and denoising

But we can also use them as **generative models**

A generative model learns the structure of data

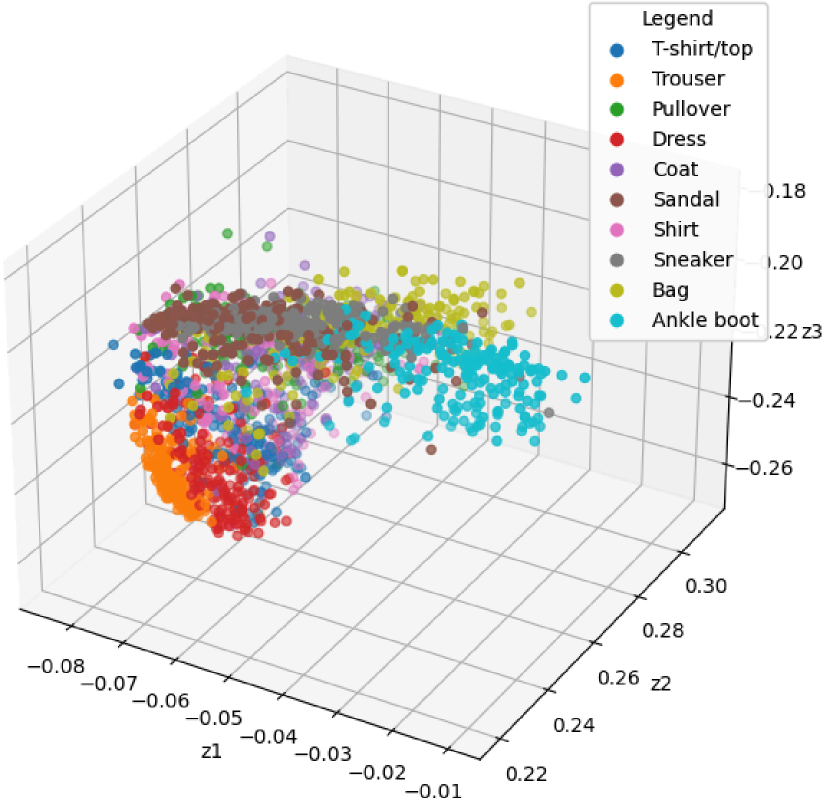
Using this structure, it generates **new** data

- Train on face dataset, generate **new** pictures
- Train on book dataset, write a **new** book
- Train on protein dataset, create **new** proteins

How does this work?

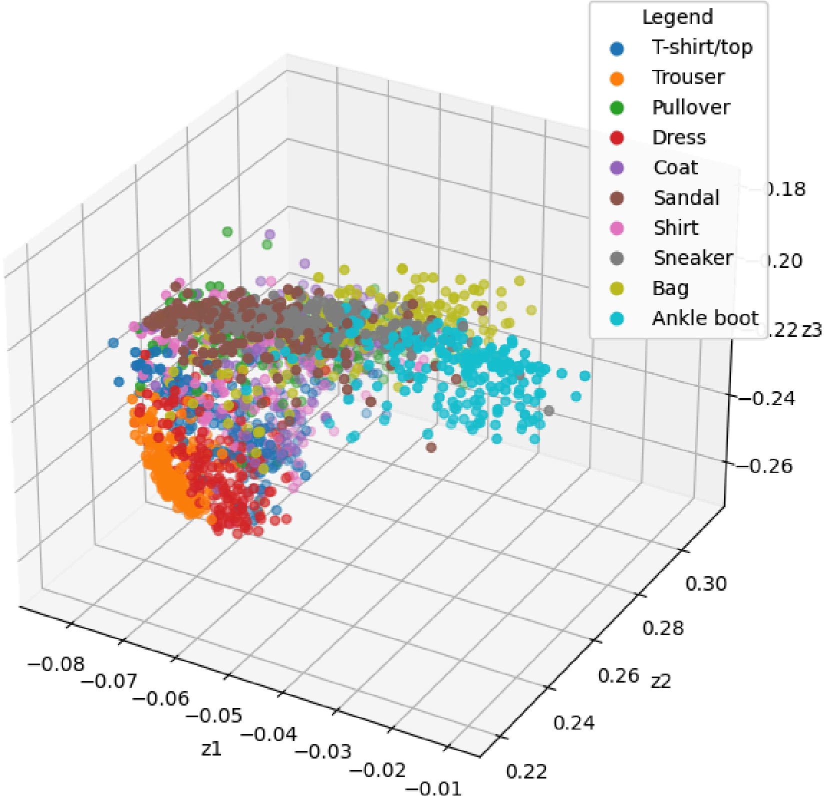
Variational Modeling

Latent space Z after training on the clothes dataset with $d_z = 3$

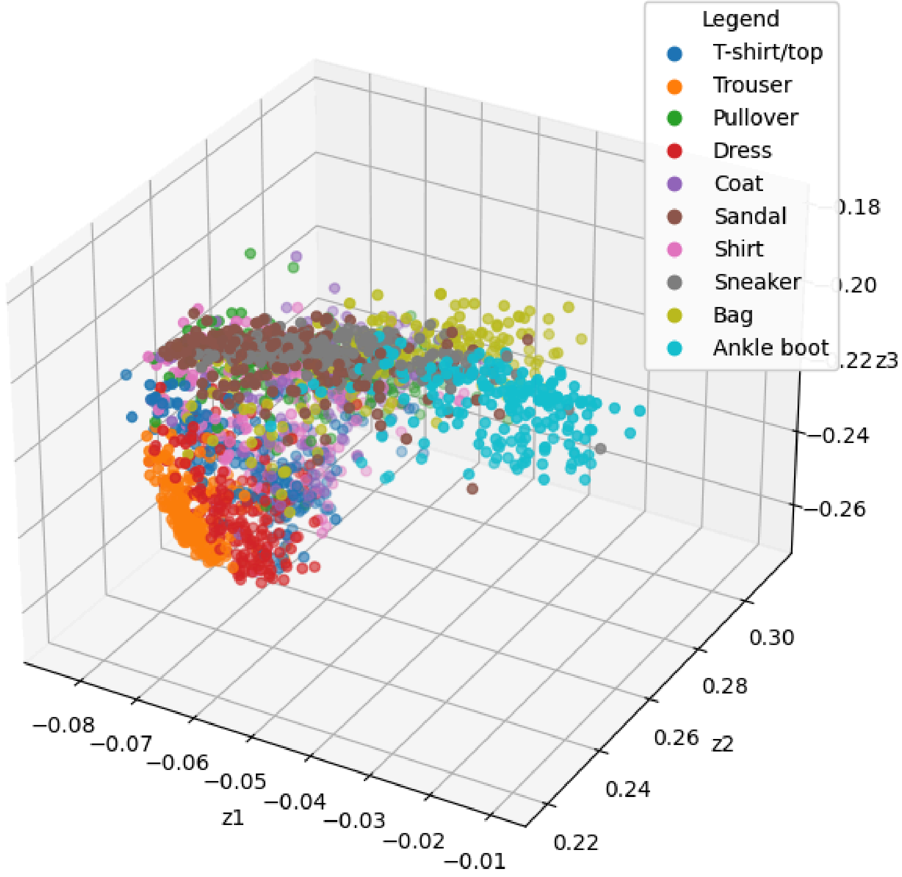


Variational Modeling

What happens if we decode a new point?

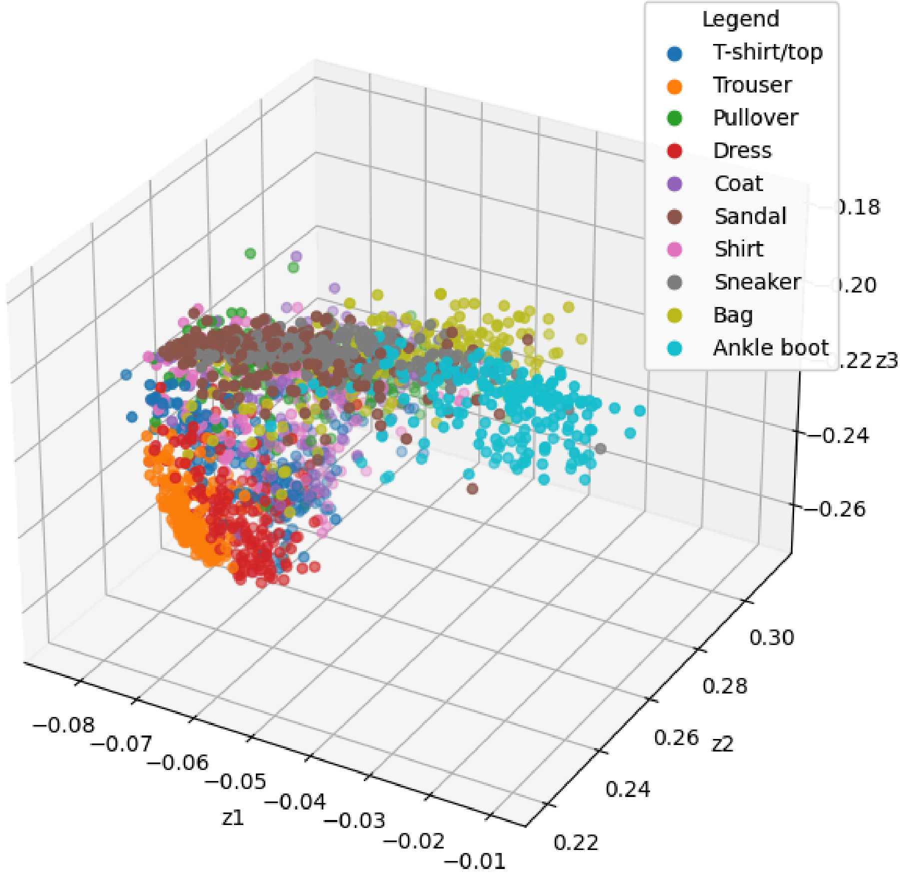


Variational Modeling



Autoencoder generative model:

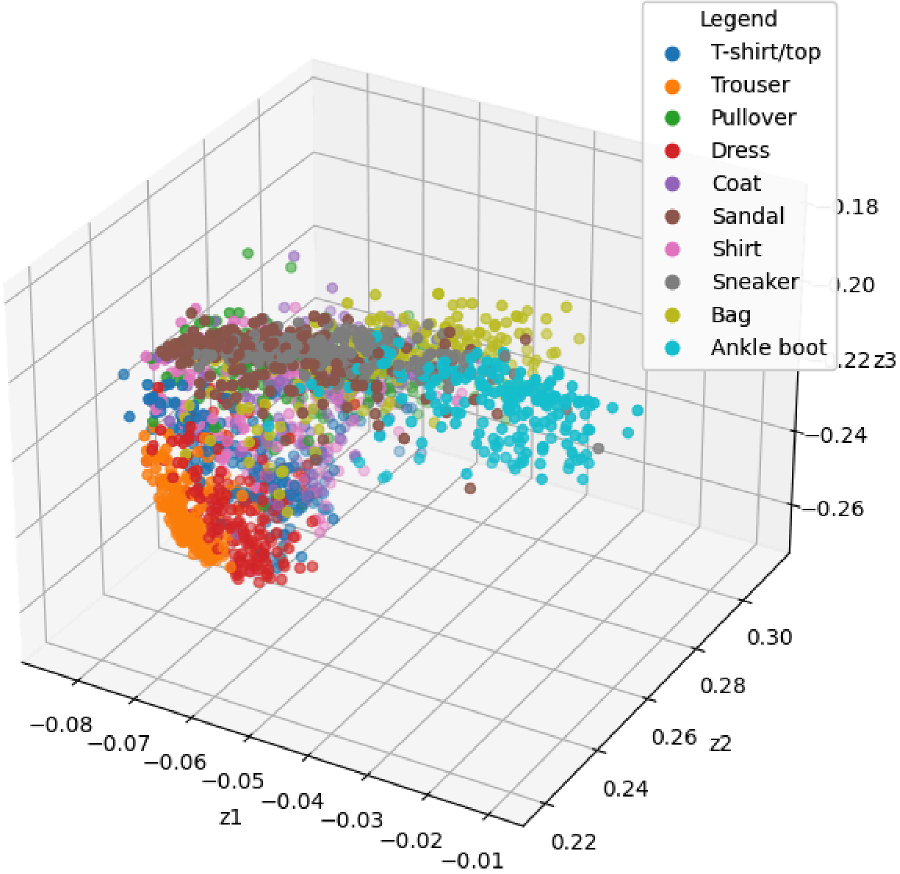
Variational Modeling



Autoencoder generative model:

$$\text{Encode} \begin{bmatrix} \boldsymbol{x}_{[1]} \\ \vdots \\ \boldsymbol{x}_{[n]} \end{bmatrix} \text{ into } \begin{bmatrix} \boldsymbol{z}_{[1]} \\ \vdots \\ \boldsymbol{z}_{[n]} \end{bmatrix}$$

Variational Modeling

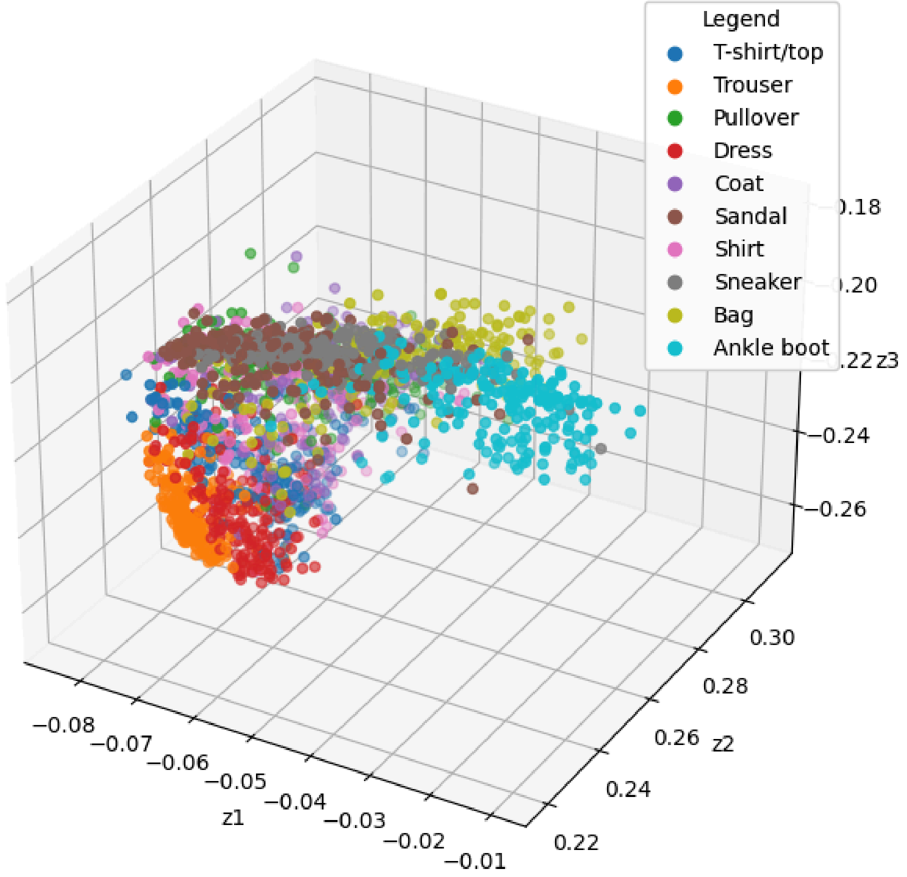


Autoencoder generative model:

$$\text{Encode} \begin{bmatrix} \boldsymbol{x}_{[1]} \\ \vdots \\ \boldsymbol{x}_{[n]} \end{bmatrix} \text{into} \begin{bmatrix} \boldsymbol{z}_{[1]} \\ \vdots \\ \boldsymbol{z}_{[n]} \end{bmatrix}$$

Pick a point $\boldsymbol{z}_{[k]}$

Variational Modeling



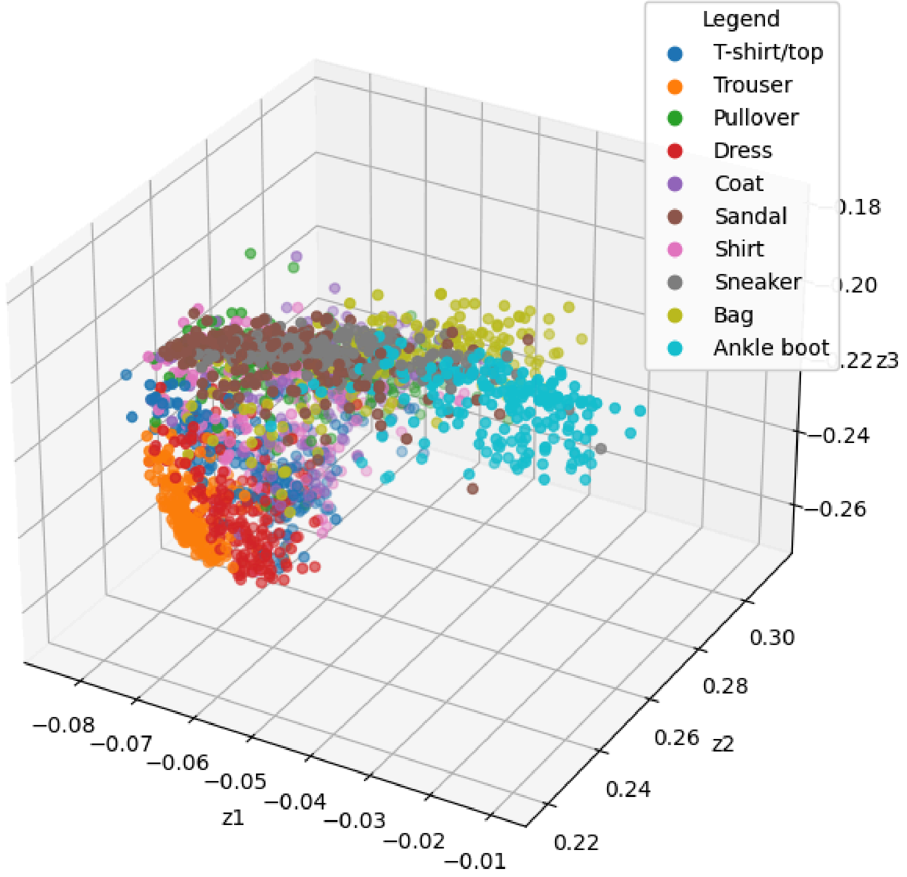
Autoencoder generative model:

Encode $\begin{bmatrix} \boldsymbol{x}_{[1]} \\ \vdots \\ \boldsymbol{x}_{[n]} \end{bmatrix}$ into $\begin{bmatrix} \boldsymbol{z}_{[1]} \\ \vdots \\ \boldsymbol{z}_{[n]} \end{bmatrix}$

Pick a point $\boldsymbol{z}_{[k]}$

Add some noise $\boldsymbol{z}_{\text{new}} = \boldsymbol{z}_{[k]} + \boldsymbol{\varepsilon}$

Variational Modeling



Autoencoder generative model:

Encode $\begin{bmatrix} \boldsymbol{x}_{[1]} \\ \vdots \\ \boldsymbol{x}_{[n]} \end{bmatrix}$ into $\begin{bmatrix} \boldsymbol{z}_{[1]} \\ \vdots \\ \boldsymbol{z}_{[n]} \end{bmatrix}$

Pick a point $\boldsymbol{z}_{[k]}$

Add some noise $\boldsymbol{z}_{\text{new}} = \boldsymbol{z}_{[k]} + \boldsymbol{\varepsilon}$

Decode $\boldsymbol{z}_{\text{new}}$ into $\boldsymbol{x}_{\text{new}}$

Variational Modeling

Variational Modeling

$$f^{-1}(\boldsymbol{z}_k + \varepsilon, \boldsymbol{\theta}_d)$$

But there is a problem, the **curse of dimensionality**

Variational Modeling

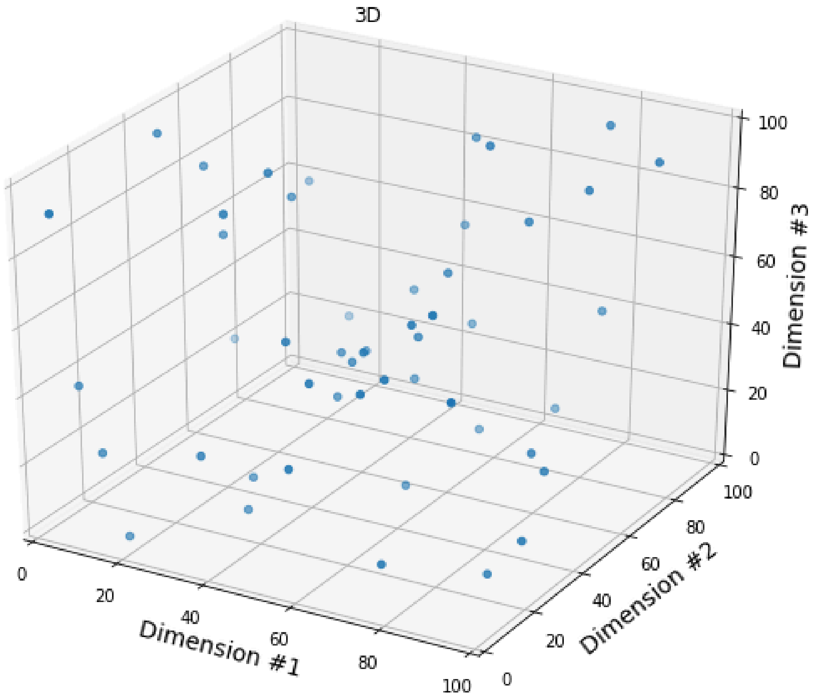
But there is a problem, the **curse of dimensionality**

As d_z increases, points move further and further apart

Variational Modeling

But there is a problem, the **curse of dimensionality**

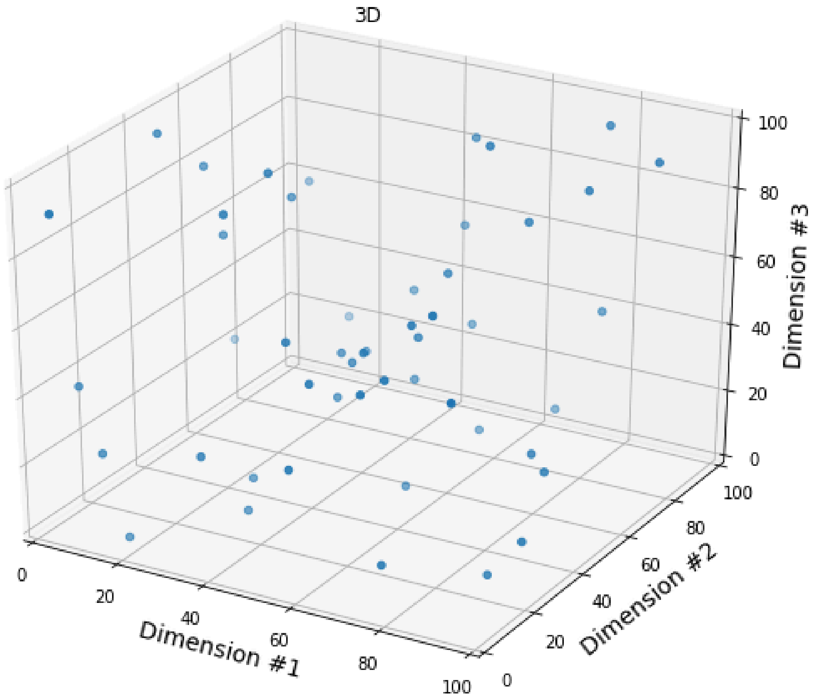
As d_z increases, points move further and further apart



Variational Modeling

But there is a problem, the **curse of dimensionality**

As d_z increases, points move further and further apart



$f^{-1}(\boldsymbol{z} + \varepsilon)$ will produce either garbage, or $\boldsymbol{z}$

Variational Modeling

Question: What can we do?

Variational Modeling

Question: What can we do?

Answer: Force the points to be close together!

Variational Modeling

Question: What can we do?

Answer: Force the points to be close together!

We will use a **variational autoencoder** (VAE)

Variational Modeling

VAE discovered by Diederik Kingma (also adam optimizer)

Variational Modeling

VAE discovered by Diederik Kingma (also adam optimizer)

Variational Modeling

Variational autoencoders (VAEs) do three things:

Variational Modeling

Variational autoencoders (VAEs) do three things:

1. Make it easy to sample random $\boldsymbol{z}$

Variational Modeling

Variational autoencoders (VAEs) do three things:

1. Make it easy to sample random $\boldsymbol{z}$
2. Keep all $\boldsymbol{z}_{[1]}, ...\boldsymbol{z}_{[n]}$ close together in a small region

Variational Modeling

Variational autoencoders (VAEs) do three things:

1. Make it easy to sample random $\boldsymbol{z}$
2. Keep all $\boldsymbol{z}_{[1]}, ...\boldsymbol{z}_{[n]}$ close together in a small region
3. Ensure that $\boldsymbol{z} + \boldsymbol{\varepsilon}$ is always meaningful

Variational Modeling

Variational autoencoders (VAEs) do three things:

1. Make it easy to sample random $\boldsymbol{z}$
2. Keep all $\boldsymbol{z}_{[1]}, ...\boldsymbol{z}_{[n]}$ close together in a small region
3. Ensure that $\boldsymbol{z} + \boldsymbol{\varepsilon}$ is always meaningful

How?

Variational Modeling

Variational autoencoders (VAEs) do three things:

1. Make it easy to sample random $\boldsymbol{z}$
2. Keep all $\boldsymbol{z}_{[1]}, ...\boldsymbol{z}_{[n]}$ close together in a small region
3. Ensure that $\boldsymbol{z} + \boldsymbol{\varepsilon}$ is always meaningful

How?

Make $\boldsymbol{z}_{[1]}, ..., \boldsymbol{z}_{[n]}$ normally distributed

Variational Modeling

Variational autoencoders (VAEs) do three things:

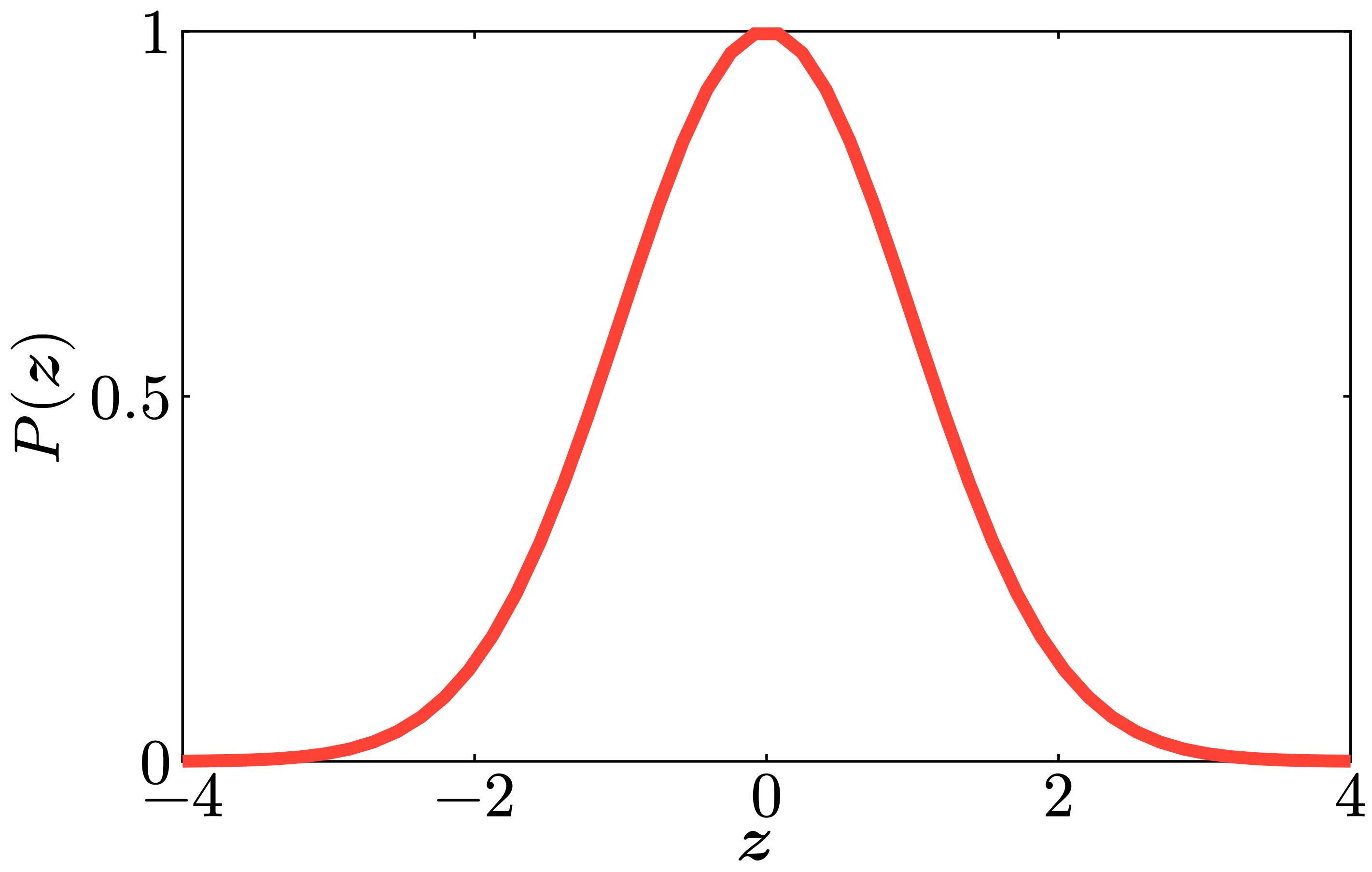
1. Make it easy to sample random $\boldsymbol{z}$
2. Keep all $\boldsymbol{z}_{[1]}, ...\boldsymbol{z}_{[n]}$ close together in a small region
3. Ensure that $\boldsymbol{z} + \boldsymbol{\varepsilon}$ is always meaningful

How?

Make $\boldsymbol{z}_{[1]}, ..., \boldsymbol{z}_{[n]}$ normally distributed

$$\boldsymbol{z} \sim \mathcal{N}(\mu, \sigma), \quad \mu = 0, \sigma = 1$$

Variational Modeling



Variational Modeling

If $\boldsymbol{z}_{[1]}, ..., \boldsymbol{z}_{[n]}$ are distributed following $\mathcal{N}(0, 1)$:

Variational Modeling

If $\boldsymbol{z}_{[1]}, ..., \boldsymbol{z}_{[n]}$ are distributed following $\mathcal{N}(0, 1)$:

1. 99.7% of $\boldsymbol{z}_{[1]}, ..., \boldsymbol{z}_{[n]}$ lie within $3\sigma = [-3, 3]$

Variational Modeling

If $\boldsymbol{z}_{[1]}, ..., \boldsymbol{z}_{[n]}$ are distributed following $\mathcal{N}(0, 1)$:

1. 99.7% of $\boldsymbol{z}_{[1]}, ..., \boldsymbol{z}_{[n]}$ lie within $3\sigma = [-3, 3]$
2. Make it easy to generate new $\boldsymbol{z}$, just sample $\boldsymbol{z} \sim \mathcal{N}(0, 1)$

Variational Modeling

So how do we ensure that $\boldsymbol{z}_{[1]}, ..., \boldsymbol{z}_{[n]}$ are normally distributed?

Variational Modeling

So how do we ensure that $\boldsymbol{z}_{[1]}, ..., \boldsymbol{z}_{[n]}$ are normally distributed?

We have to remember conditional probabilities

Variational Modeling

So how do we ensure that $\boldsymbol{z}_{[1]}, ..., \boldsymbol{z}_{[n]}$ are normally distributed?

We have to remember conditional probabilities

$$P(\text{rain} \mid \text{cloud}) = \text{Probability of rain, given that it is cloudy}$$

Variational Modeling

So how do we ensure that $\boldsymbol{z}_{[1]}, ..., \boldsymbol{z}_{[n]}$ are normally distributed?

We have to remember conditional probabilities

$$P(\text{rain} \mid \text{cloud}) = \text{Probability of rain, given that it is cloudy}$$

Variational Modeling

Key idea 1: We want to model the distribution over the dataset X

$$P(\boldsymbol{x}; \boldsymbol{\theta}), \quad \boldsymbol{x} \sim \boldsymbol{X}$$

Variational Modeling

Key idea 1: We want to model the distribution over the dataset X

$$P(\boldsymbol{x}; \boldsymbol{\theta}), \quad \boldsymbol{x} \sim \boldsymbol{X}$$

We want to learn $\boldsymbol{\theta}$ that best models the distribution of possible faces

Variational Modeling

Key idea 1: We want to model the distribution over the dataset X

$$P(\boldsymbol{x}; \boldsymbol{\theta}), \quad \boldsymbol{x} \sim \boldsymbol{X}$$

We want to learn $\boldsymbol{\theta}$ that best models the distribution of possible faces

Large $P(\boldsymbol{x}; \boldsymbol{\theta})$

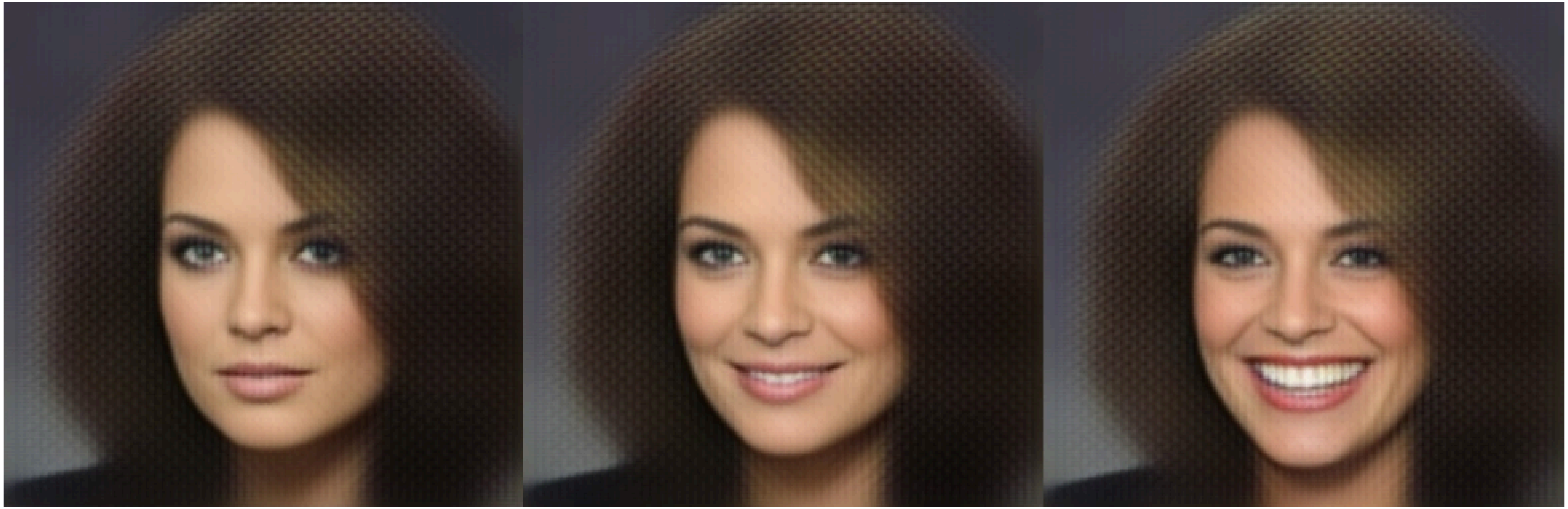
$P(\boldsymbol{x}; \boldsymbol{\theta}) \approx 0$

Variational Modeling

Key idea 2: There is some latent variable $\boldsymbol{z}$ which generates data $\boldsymbol{x}$

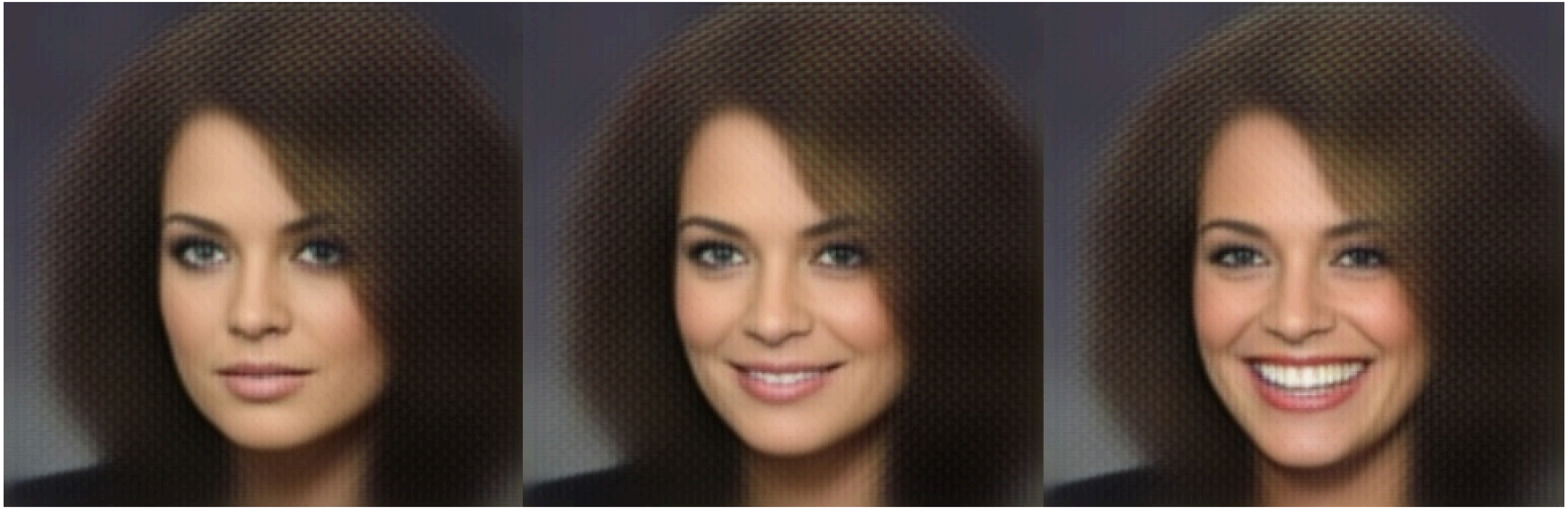
Variational Modeling

Key idea 2: There is some latent variable $\boldsymbol{z}$ which generates data $\boldsymbol{x}$

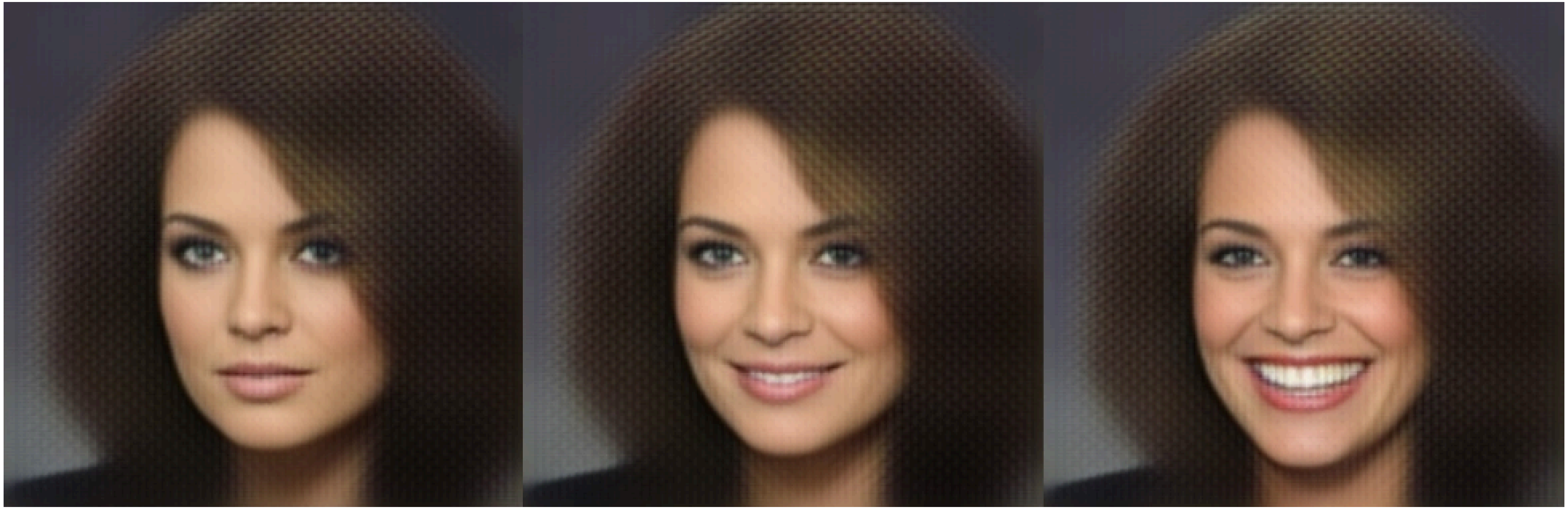
x : 

$$z : [\text{woman brown hair (frown} \mid \text{smile)}]$$

Variational Modeling

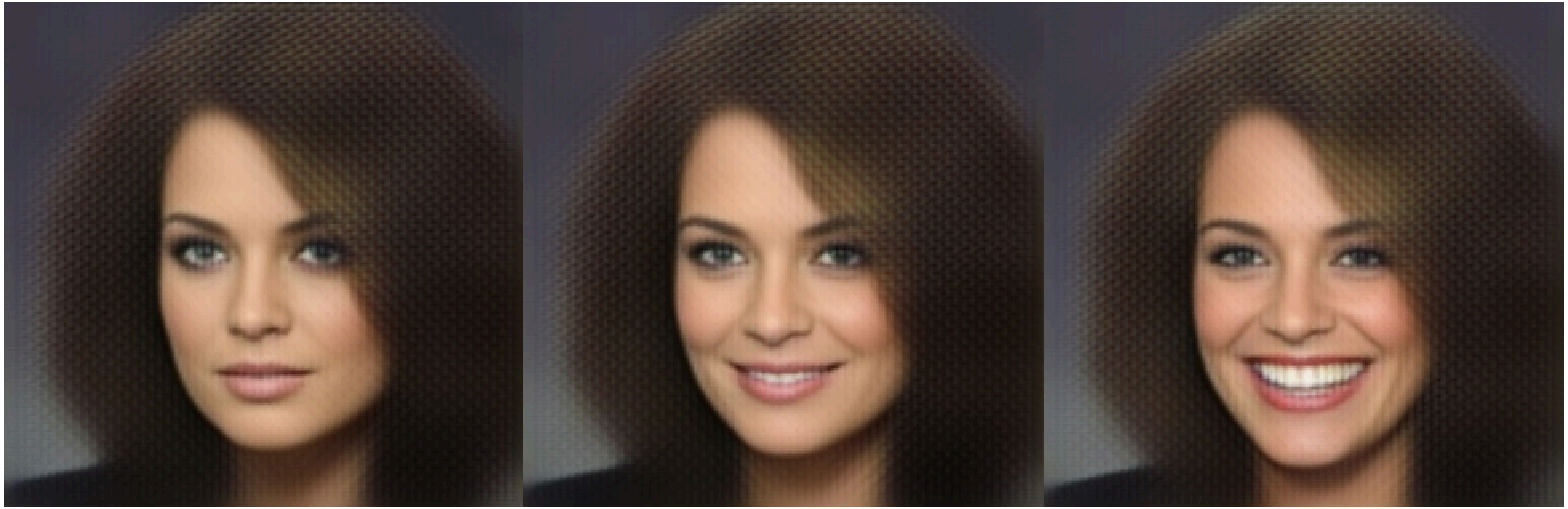


Variational Modeling



Network can only see $\boldsymbol{x}$, it cannot directly observe $\boldsymbol{z}$

Variational Modeling



Network can only see $\boldsymbol{x}$, it cannot directly observe $\boldsymbol{z}$

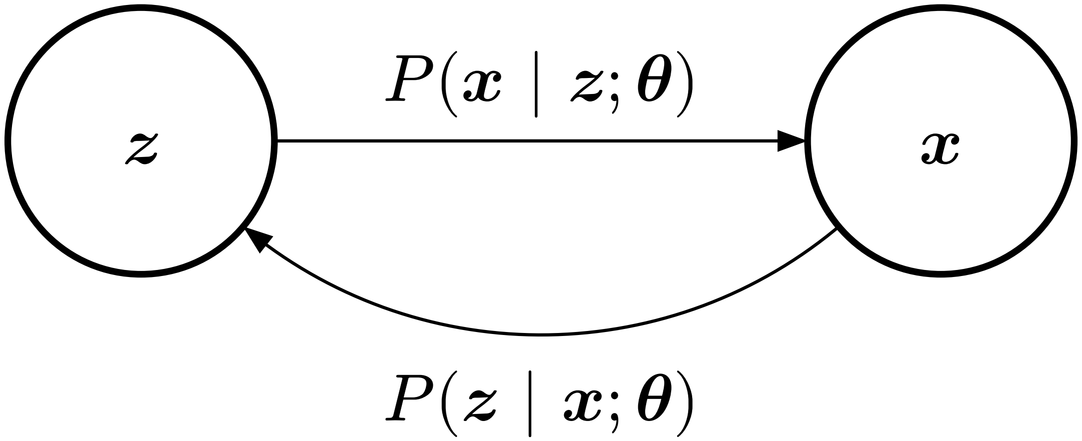
Given $\boldsymbol{x}$, find the probability that the person is smiling $P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta})$

Variational Modeling

We cast the autoencoding task as a **variational inference** problem

Variational Modeling

We cast the autoencoding task as a **variational inference** problem



Decoder

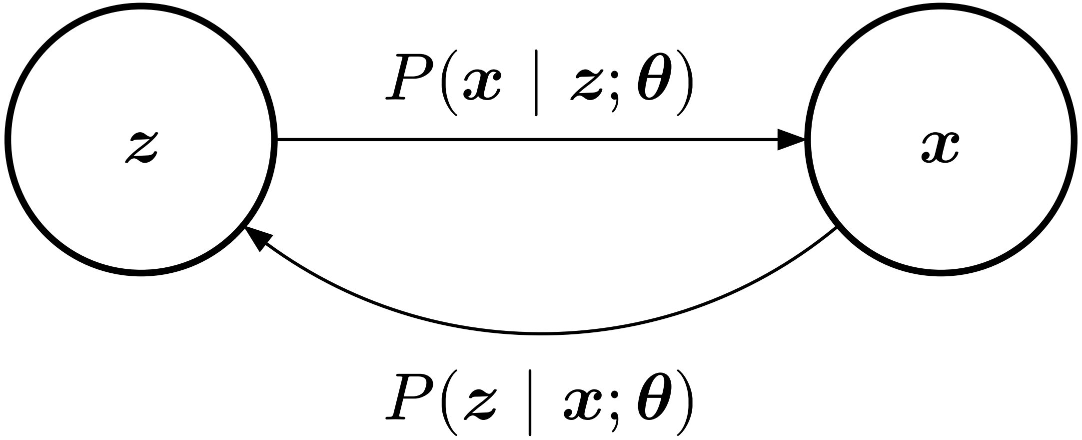
$$P(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta})$$

Encoder

$$P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta})$$

Variational Modeling

We cast the autoencoding task as a **variational inference** problem



Decoder	Encoder
$P(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta})$	$P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta})$

We want to learn both the encoder and decoder: $P(\boldsymbol{z}, \boldsymbol{x}; \boldsymbol{\theta})$

Variational Modeling

$$P(\boldsymbol{z}, \boldsymbol{x}; \boldsymbol{\theta}) = P(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta}) \ P(\boldsymbol{z}; \boldsymbol{\theta})$$

Variational Modeling

$$P(\boldsymbol{z}, \boldsymbol{x}; \boldsymbol{\theta}) = P(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta}) \ P(\boldsymbol{z}; \boldsymbol{\theta})$$

We can choose any distribution for $P(\boldsymbol{z})$

Variational Modeling

$$P(\boldsymbol{z}, \boldsymbol{x}; \boldsymbol{\theta}) = P(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta}) \; P(\boldsymbol{z}; \boldsymbol{\theta})$$

We can choose any distribution for $P(\boldsymbol{z})$

$$P(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{1})$$

Variational Modeling

$$P(\boldsymbol{z}, \boldsymbol{x}; \boldsymbol{\theta}) = P(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta}) \; P(\boldsymbol{z}; \boldsymbol{\theta})$$

We can choose any distribution for $P(\boldsymbol{z})$

$$P(\boldsymbol{z}) = \mathcal{N}(\mathbf{0}, \mathbf{1})$$

We can generate all possible $\boldsymbol{x}$ by sampling $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$

Variational Modeling

$$P(\boldsymbol{z}, \boldsymbol{x}; \boldsymbol{\theta}) = P(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta}) \ P(\boldsymbol{z}; \boldsymbol{\theta})$$

We can choose any distribution for $P(\boldsymbol{z})$

$$P(\boldsymbol{z}) = \mathcal{N}(\mathbf{0}, \mathbf{1})$$

We can generate all possible $\boldsymbol{x}$ by sampling $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$

We can randomly generate $\boldsymbol{z}$, which we can decode into new $\boldsymbol{x}$!

Variational Modeling

Now, all we must do is find $\boldsymbol{\theta}$ that best explains the dataset distribution

Variational Modeling

Now, all we must do is find $\boldsymbol{\theta}$ that best explains the dataset distribution

Learned distribution $P(\boldsymbol{x}; \boldsymbol{\theta})$ to be close to dataset $P(\boldsymbol{x}), \quad \boldsymbol{x} \sim X$

Variational Modeling

Now, all we must do is find $\boldsymbol{\theta}$ that best explains the dataset distribution

Learned distribution $P(\boldsymbol{x}; \boldsymbol{\theta})$ to be close to dataset $P(\boldsymbol{x}), \quad \boldsymbol{x} \sim X$

We need some error function between $P(\boldsymbol{x}; \boldsymbol{\theta})$ and $P(\boldsymbol{x})$

Variational Modeling

Now, all we must do is find $\boldsymbol{\theta}$ that best explains the dataset distribution

Learned distribution $P(\boldsymbol{x}; \boldsymbol{\theta})$ to be close to dataset $P(\boldsymbol{x}), \quad \boldsymbol{x} \sim X$

We need some error function between $P(\boldsymbol{x}; \boldsymbol{\theta})$ and $P(\boldsymbol{x})$

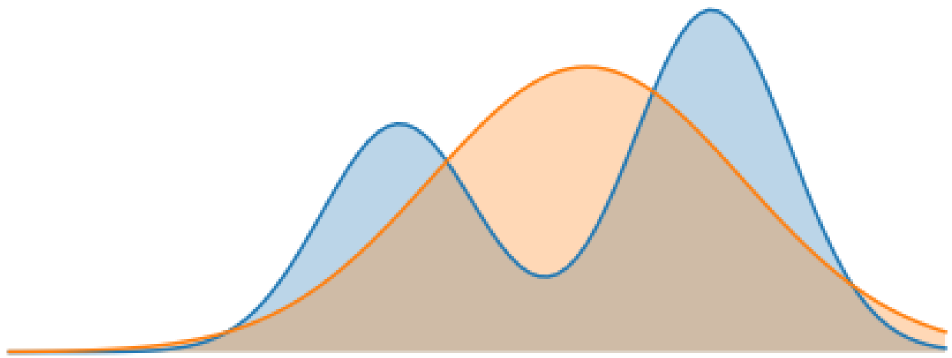
Question: How do we measure the distance between probability distributions?

Variational Modeling

Answer: KL divergence

Variational Modeling

Answer: KL divergence



$$\mathrm{KL}(P, Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

Variational Modeling

Learn the parameters for our model

Variational Modeling

Learn the parameters for our model

$$\arg\min_{\boldsymbol{\theta}} \mathrm{KL}(P(\boldsymbol{x}), P(\boldsymbol{x}; \boldsymbol{\theta}))$$

Variational Modeling

Learn the parameters for our model

$$\arg\min_{\boldsymbol{\theta}} \mathrm{KL}(P(\boldsymbol{x}), P(\boldsymbol{x}; \boldsymbol{\theta}))$$

Unfortunately, this objective is intractable to optimize

Variational Modeling

Learn the parameters for our model

$$\arg \min_{\boldsymbol{\theta}} \mathrm{KL}(P(\boldsymbol{x}), P(\boldsymbol{x}; \boldsymbol{\theta}))$$

Unfortunately, this objective is intractable to optimize

The paper provides surrogate objective

$$\arg \min_{\boldsymbol{\theta}} [- \log P(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta}) + \mathrm{KL}(P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta}), P(\boldsymbol{z}))]$$

Variational Modeling

Learn the parameters for our model

$$\arg\min_{\boldsymbol{\theta}} \mathrm{KL}(P(\boldsymbol{x}), P(\boldsymbol{x}; \boldsymbol{\theta}))$$

Unfortunately, this objective is intractable to optimize

The paper provides surrogate objective

$$\arg\min_{\boldsymbol{\theta}} [-\log P(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta}) + \mathrm{KL}(P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta}), P(\boldsymbol{z}))]$$

We call this the **Evidence Lower Bound Objective** (ELBO)

Variational Modeling

$$\arg\min_{\boldsymbol{\theta}} [- \log P(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta}) + \mathrm{KL}(P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta}), P(\boldsymbol{z}))]$$

Variational Modeling

$$\arg\min_{\boldsymbol{\theta}}[-\log P(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta}) + \mathrm{KL}(P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta}), P(\boldsymbol{z}))]$$

How is this ELBO helpful?

Variational Modeling

$$\arg\min_{\boldsymbol{\theta}} [-\log P(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta}) + \mathrm{KL}(P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta}), P(\boldsymbol{z}))]$$

How is this ELBO helpful?

Decoder	Encoder	Prior
$P(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta})$	$P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta})$	$P(\boldsymbol{z}) = \mathcal{N}(\mathbf{0}, \mathbf{1})$

Variational Modeling

$$\arg \min_{\boldsymbol{\theta}} [- \log P(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta}) + \mathrm{KL}(P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta}), P(\boldsymbol{z}))]$$

How is this ELBO helpful?

Decoder	Encoder	Prior
$P(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta})$	$P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta})$	$P(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{1})$

$$\arg \min_{\boldsymbol{\theta}} \left[\underbrace{- \log P(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta})}_{\text{Reconstruction error}} + \underbrace{\mathrm{KL}(P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta}), P(\boldsymbol{z}))}_{\text{Constrain latent}} \right]$$

Variational Modeling

$$\arg\min_{\boldsymbol{\theta}} [- \log P(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta}) + \mathrm{KL}(P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta}), P(\boldsymbol{z}))]$$

How is this ELBO helpful?

Decoder	Encoder	Prior
$P(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta})$	$P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta})$	$P(\boldsymbol{z}) = \mathcal{N}(\mathbf{0}, \mathbf{1})$

$$\arg\min_{\boldsymbol{\theta}} \left[\underbrace{- \log P(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta})}_{\text{Reconstruction error}} + \underbrace{\mathrm{KL}(P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta}), P(\boldsymbol{z}))}_{\text{Constrain latent}} \right]$$

Now we know how to train our autoencoder!

Agenda

1. Review
2. Compression
3. Autoencoders
4. Applications
5. **Variational Modeling**
6. VAE Implementation
7. Coding

Agenda

1. Review
2. Compression
3. Autoencoders
4. Applications
5. Variational Modeling
6. **VAE Implementation**
7. Coding

VAE Implementation

How do we implement f (i.e., $P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta})$)?

VAE Implementation

How do we implement f (i.e., $P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta})$)?

$$f : X \times \Theta \mapsto \Delta Z$$

VAE Implementation

How do we implement f (i.e., $P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta})$)?

$$f : X \times \Theta \mapsto \Delta Z$$

Normal distribution has a mean $\mu \in \mathbb{R}$ and standard deviation $\sigma \in \mathbb{R}_+$

VAE Implementation

How do we implement f (i.e., $P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta})$)?

$$f : X \times \Theta \mapsto \Delta Z$$

Normal distribution has a mean $\mu \in \mathbb{R}$ and standard deviation $\sigma \in \mathbb{R}_+$

Our encoder should output d_z means and d_z standard deviations

VAE Implementation

How do we implement f (i.e., $P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta})$)?

$$f : X \times \Theta \mapsto \Delta Z$$

Normal distribution has a mean $\mu \in \mathbb{R}$ and standard deviation $\sigma \in \mathbb{R}_+$

Our encoder should output d_z means and d_z standard deviations

$$f : X \times \Theta \mapsto \mathbb{R}^{d_z} \times \mathbb{R}_+^{d_z}$$

VAE Implementation

```
core = nn.Sequential(...)
mu_layer = nn.Linear(d_h, d_z)
# Neural networks output real numbers
# But sigma must be positive
# So we output log sigma, because e^(sigma) is always
positive
log_sigma_layer = nn.Linear(d_h, d_z)

tmp = core(x)
mu = mu_layer(tmp)
log_sigma = log_sigma_layer(tmp)
distribution = (mu, exp(sigma))
```

VAE Implementation

We covered the encoder

$$f : X \times \Theta \mapsto \Delta Z$$

We can use the same decoder as a standard autoencoder

VAE Implementation

We covered the encoder

$$f : X \times \Theta \mapsto \Delta Z$$

We can use the same decoder as a standard autoencoder

$$f^{-1} : Z \times \Theta \mapsto X$$

Question: Any issues?

VAE Implementation

We covered the encoder

$$f : X \times \Theta \mapsto \Delta Z$$

We can use the same decoder as a standard autoencoder

$$f^{-1} : Z \times \Theta \mapsto X$$

Question: Any issues?

Answer: Encoder outputs a distribution ΔZ but decoder input is Z

VAE Implementation

We can sample from the distribution

$$\boldsymbol{\mu}, \boldsymbol{\sigma} = f(\boldsymbol{x}, \boldsymbol{\theta}_e)$$

$$\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$$

VAE Implementation

We can sample from the distribution

$$\boldsymbol{\mu}, \boldsymbol{\sigma} = f(\boldsymbol{x}, \boldsymbol{\theta}_e)$$

$$\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$$

But there is a problem! Sampling is not differentiable

VAE Implementation

We can sample from the distribution

$$\boldsymbol{\mu}, \boldsymbol{\sigma} = f(\boldsymbol{x}, \boldsymbol{\theta}_e)$$
$$\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$$

But there is a problem! Sampling is not differentiable

Question: Why does this matter?

VAE Implementation

We can sample from the distribution

$$\boldsymbol{\mu}, \boldsymbol{\sigma} = f(\boldsymbol{x}, \boldsymbol{\theta}_e)$$
$$\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$$

But there is a problem! Sampling is not differentiable

Question: Why does this matter?

Answer: Must be differentiable for gradient descent

VAE Implementation

VAE paper proposes the **reparameterization trick**

VAE Implementation

VAE paper proposes the **reparameterization trick**

$$\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$$

$$\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\varepsilon} \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$$

VAE Implementation

VAE paper proposes the **reparameterization trick**

$$\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$$

$$\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\varepsilon} \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$$

Gradient can flow through $\boldsymbol{\mu}, \boldsymbol{\sigma}$

VAE Implementation

VAE paper proposes the **reparameterization trick**

$$\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$$

$$\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\varepsilon} \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$$

Gradient can flow through $\boldsymbol{\mu}, \boldsymbol{\sigma}$

We can sample and use gradient descent

VAE Implementation

VAE paper proposes the **reparameterization trick**

$$\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$$

$$\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\varepsilon} \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$$

Gradient can flow through $\boldsymbol{\mu}, \boldsymbol{\sigma}$

We can sample and use gradient descent

This trick only works with certain distributions

VAE Implementation

Put it all together

VAE Implementation

Put it all together

Step 1: Encode the input to a normal distribution

$$\boldsymbol{\mu}, \boldsymbol{\sigma} = f(\boldsymbol{x}, \boldsymbol{\theta}_e)$$

VAE Implementation

Put it all together

Step 1: Encode the input to a normal distribution

$$\boldsymbol{\mu}, \boldsymbol{\sigma} = f(\boldsymbol{x}, \boldsymbol{\theta}_e)$$

Step 2: Generate a sample from distribution

$$\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\varepsilon}$$

VAE Implementation

Put it all together

Step 1: Encode the input to a normal distribution

$$\boldsymbol{\mu}, \boldsymbol{\sigma} = f(\boldsymbol{x}, \boldsymbol{\theta}_e)$$

Step 2: Generate a sample from distribution

$$\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\varepsilon}$$

Step 3: Decode the sample

$$\boldsymbol{x} = f^{-1}(\boldsymbol{z}, \boldsymbol{\theta}_d)$$

VAE Implementation

One last thing, implement the loss function

VAE Implementation

One last thing, implement the loss function

Decoder	Encoder	Prior
$P(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta})$	$P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta})$	$P(\boldsymbol{z}) = \mathcal{N}(\mathbf{0}, \mathbf{1})$

VAE Implementation

One last thing, implement the loss function

Decoder	Encoder	Prior
$P(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta})$	$P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta})$	$P(\boldsymbol{z}) = \mathcal{N}(\mathbf{0}, \mathbf{1})$

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}) = \arg \min_{\boldsymbol{\theta}} \left[\underbrace{- \log P(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta})}_{\text{Reconstruction error}} + \underbrace{\mathrm{KL}(P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta}), P(\boldsymbol{z}))}_{\text{Constrain latent}} \right]$$

VAE Implementation

One last thing, implement the loss function

Decoder	Encoder	Prior
$P(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta})$	$P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta})$	$P(\boldsymbol{z}) = \mathcal{N}(\mathbf{0}, \mathbf{1})$

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} \left[\underbrace{- \log P(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta})}_{\text{Reconstruction error}} + \underbrace{\mathrm{KL}(P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta}), P(\boldsymbol{z}))}_{\text{Constrain latent}} \right]$$

Start with the KL term first

VAE Implementation

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}}[-\log P(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta}) + \mathrm{KL}(P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta}), P(\boldsymbol{z}))]$$

VAE Implementation

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} [- \log P(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta}) + \mathrm{KL}(P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta}), P(\boldsymbol{z}))]$$

First, rewrite KL term using our encoder f

VAE Implementation

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} [-\log P(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta}) + \mathrm{KL}(P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta}), P(\boldsymbol{z}))]$$

First, rewrite KL term using our encoder f

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} [-\log P(\boldsymbol{x} \mid \boldsymbol{z}) + \mathrm{KL}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), P(\boldsymbol{z}))]$$

VAE Implementation

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} [-\log P(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta}) + \mathrm{KL}(P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta}), P(\boldsymbol{z}))]$$

First, rewrite KL term using our encoder f

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} [-\log P(\boldsymbol{x} \mid \boldsymbol{z}) + \mathrm{KL}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), P(\boldsymbol{z}))]$$

$P(\boldsymbol{z})$ and $f(\boldsymbol{x}, \boldsymbol{\theta}_e)$ are Gaussian, we can simplify KL term

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}) = \underbrace{\log P(\boldsymbol{x} \mid \boldsymbol{z})}_{\text{Reconstruction error}} - \left(\sum_{j=1}^{d_z} \mu_j^2 + \sigma_j^2 - \log(\sigma^2) - 1 \right)$$

VAE Implementation

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}) = \underbrace{\log P(\boldsymbol{x} \mid \boldsymbol{z})}_{\text{Reconstruction error}} - \left(\sum_{j=1}^{d_z} \mu_j^2 + \sigma_j^2 - \log(\sigma^2) - 1 \right)$$

VAE Implementation

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}) = \underbrace{\log P(\boldsymbol{x} \mid \boldsymbol{z})}_{\text{Reconstruction error}} - \left(\sum_{j=1}^{d_z} \mu_j^2 + \sigma_j^2 - \log(\sigma^2) - 1 \right)$$

Next, plug in square error for reconstruction error

VAE Implementation

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}) = \underbrace{\log P(\boldsymbol{x} \mid \boldsymbol{z})}_{\text{Reconstruction error}} - \left(\sum_{j=1}^{d_z} \mu_j^2 + \sigma_j^2 - \log(\sigma^2) - 1 \right)$$

Next, plug in square error for reconstruction error

$$= \sum_{j=1}^{d_z} \left(x_j - f^{-1}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), \boldsymbol{\theta}_d)_j \right)^2 - \left(\sum_{j=1}^{d_z} \mu_j^2 + \sigma_j^2 - \log(\sigma_j^2) - 1 \right)$$

VAE Implementation

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}) = \underbrace{\log P(\boldsymbol{x} \mid \boldsymbol{z})}_{\text{Reconstruction error}} - \left(\sum_{j=1}^{d_z} \mu_j^2 + \sigma_j^2 - \log(\sigma^2) - 1 \right)$$

Next, plug in square error for reconstruction error

$$= \sum_{j=1}^{d_z} \left(x_j - f^{-1}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), \boldsymbol{\theta}_d)_j \right)^2 - \left(\sum_{j=1}^{d_z} \mu_j^2 + \sigma_j^2 - \log(\sigma_j^2) - 1 \right)$$

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{j=1}^{d_z} \left(x_j - f^{-1}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), \boldsymbol{\theta}_d)_j \right)^2 - \left(\sum_{j=1}^{d_z} \mu_j^2 + \sigma_j^2 - \log(\sigma_j^2) - 1 \right)$$

VAE Implementation

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{j=1}^{d_z} \left(x_j - f^{-1}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), \boldsymbol{\theta}_d)_j\right)^2 - \left(\sum_{j=1}^{d_z} \mu_j^2 + \sigma_j^2 - \log(\sigma_j^2) - 1\right)$$

VAE Implementation

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{j=1}^{d_z} \left(x_j - f^{-1}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), \boldsymbol{\theta}_d)_j \right)^2 - \left(\sum_{j=1}^{d_z} \mu_j^2 + \sigma_j^2 - \log(\sigma_j^2) - 1 \right)$$

Finally, define over the entire dataset

$$\mathcal{L}(\boldsymbol{X}, \boldsymbol{\theta}) = \sum_{i=1}^{n} \sum_{j=1}^{d_z} \left(x_{[i],j} - f^{-1}\left(f\left(\boldsymbol{x}_{[i]}, \boldsymbol{\theta}_e\right), \boldsymbol{\theta}_d\right)_j \right)^2 -$$

$$\left(\sum_{i=1}^{n} \sum_{j=1}^{d_z} \mu_{[i],j}^2 + \sigma_{[i],j}^2 - \log\left(\sigma_{[i],j}^2\right) - 1 \right)$$

VAE Implementation

VAE Implementation

$$\mathcal{L}(\boldsymbol{X}, \boldsymbol{\theta}) = \sum_{i=1}^{n} \sum_{j=1}^{d_z} \left(x_{[i],j} - f^{-1}\left(f\left(\boldsymbol{x}_{[i]}, \boldsymbol{\theta}_e\right), \boldsymbol{\theta}_d\right)_j \right)^2 -$$

$$\left(\sum_{i=1}^{n} \sum_{j=1}^{d_z} \mu_{[i],j}^2 + \sigma_{[i],j}^2 - \log\left(\sigma_{[i],j}^2\right) - 1 \right)$$

Scale of two terms can vary, we do not want one term to dominate

VAE Implementation

Paper suggests using minibatch size m and dataset size n

VAE Implementation

Paper suggests using minibatch size m and dataset size n

$$\mathcal{L}(\boldsymbol{X}, \boldsymbol{\theta}) = \frac{m}{n} \sum_{i=1}^{n} \sum_{j=1}^{d_z} \left(x_{[i],j} - f^{-1}\left(f\left(\boldsymbol{x}_{[i]}, \boldsymbol{\theta}_e \right), \boldsymbol{\theta}_d \right)_j \right)^2 -$$

$$\left(\sum_{i=1}^{n} \sum_{j=1}^{d_z} \mu_{[i],j}^2 + \sigma_{[i],j}^2 - \log\left(\sigma_{[i],j}^2 \right) - 1 \right)$$

VAE Implementation

Another paper finds hyperparameter β also helps

VAE Implementation

Another paper finds hyperparameter β also helps

$$\mathcal{L}(\boldsymbol{X}, \boldsymbol{\theta}) = \frac{m}{n} \sum_{i=1}^{n} \sum_{j=1}^{d_z} \left(x_{[i],j} - f^{-1}\left(f\left(\boldsymbol{x}_{[i]}, \boldsymbol{\theta}_e \right), \boldsymbol{\theta}_d \right)_j \right)^2 -$$

$$\beta \left(\sum_{i=1}^{n} \sum_{j=1}^{d_z} \mu_{[i],j}^2 + \sigma_{[i],j}^2 - \log\left(\sigma_{[i],j}^2 \right) - 1 \right)$$

VAE Implementation

```
def L(model, x, m, n, beta, key):
  mu, sigma = model.f(x)
  epsilon = jax.random.normal(key, x.shape[0])
  z = mu + sigma * epsilon
  pred_x = model.f_inverse(z)

  recon = jnp.sum((x - pred_x) ** 2)
  kl = jnp.sum(mu ** 2 + sigma ** 2 - jnp.log(sigma) - 1)

  return m / n * recon + beta * kl
```

VAE Implementation

Coding VAE

https://colab.research.google.com/drive/1UyR_W6NDIujaJXYlHZh6O3NfaCAMscpH#scrollTo=nmyQ8aE2pSbb

https://www.youtube.com/watch?v=UZDiGooFs54