Autoencoders and Generative Models

CISC 7026: Introduction to Deep Learning

University of Macau

Agenda

1. Review
2. Compression
3. Autoencoders
4. Applications
5. Variational Models
6. Coding

Agenda

1. **Review**
2. Compression
3. Autoencoders
4. Applications
5. Variational Models
6. Coding

Review

Agenda

1. **Review**
2. Compression
3. Autoencoders
4. Applications
5. Variational Models
6. Coding

Agenda

1. Review
2. **Compression**
3. Autoencoders
4. Applications
5. Variational Models
6. Coding

Compression

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 concepts with friends (paintings, music, films, etc), you summarize them

Compression

When you discuss concepts with friends (paintings, music, films, etc), you summarize them

This is a form of **compression**

Compression

When you discuss concepts with friends (paintings, music, films, etc), 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 concepts with friends (paintings, music, films, etc), 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 take some data and reduce its size by removing unnecessary information

Compression

When you discuss concepts with friends (paintings, music, films, etc), 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 take some data and reduce its size by removing unnecessary 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

Fortunately, very talented engineers created the H.264 video codec

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

Fortunately, very talented engineers created the H.264 video codec

H.264 MPEG-AVC **encoder** transforms 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

We download Z from the internet

Compression

We download Z from the internet $\quad Z \in \{0, 1\}^n$

Compression

We download Z from the internet $\quad Z \in \{0, 1\}^n$

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

Compression

We download Z from the internet $\quad Z \in \{0, 1\}^n$

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

Question: How do we watch the video?

Compression

We download Z from the internet $Z \in \{0, 1\}^n$

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

Question: How do we watch the video?

Answer: Transform or **decode** Z back into pixels

Compression

We download Z from the internet $\quad Z \in \{0, 1\}^n$

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

Question: How do we watch the video?

Answer: Transform or **decode** Z back into pixels

We need to undo (invert) the encoding function f

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

Compression

We download Z from the internet $\quad Z \in \{0, 1\}^n$

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

Question: How do we watch the video?

Answer: Transform or **decode** Z back into pixels

We need to undo (invert) the encoding function f

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

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

Compression

To summarize:

Compression

To summarize:

We encode pixels into a bit string to save space

Compression

To summarize:

We encode pixels into a bit string to save space

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

Compression

To summarize:

We encode pixels into a bit string to save space

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

We store the film as a bit string on websites or your computer

Compression

To summarize:

We encode pixels into a bit string to save space

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

We store the film as a bit string on websites or your computer

When you want to watch, we decode the string back into pixels

Compression

To summarize:

We encode pixels into a bit string to save space

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

We store the film as a bit string on websites or your computer

When you want to watch, we decode the string back into pixels

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

Compression

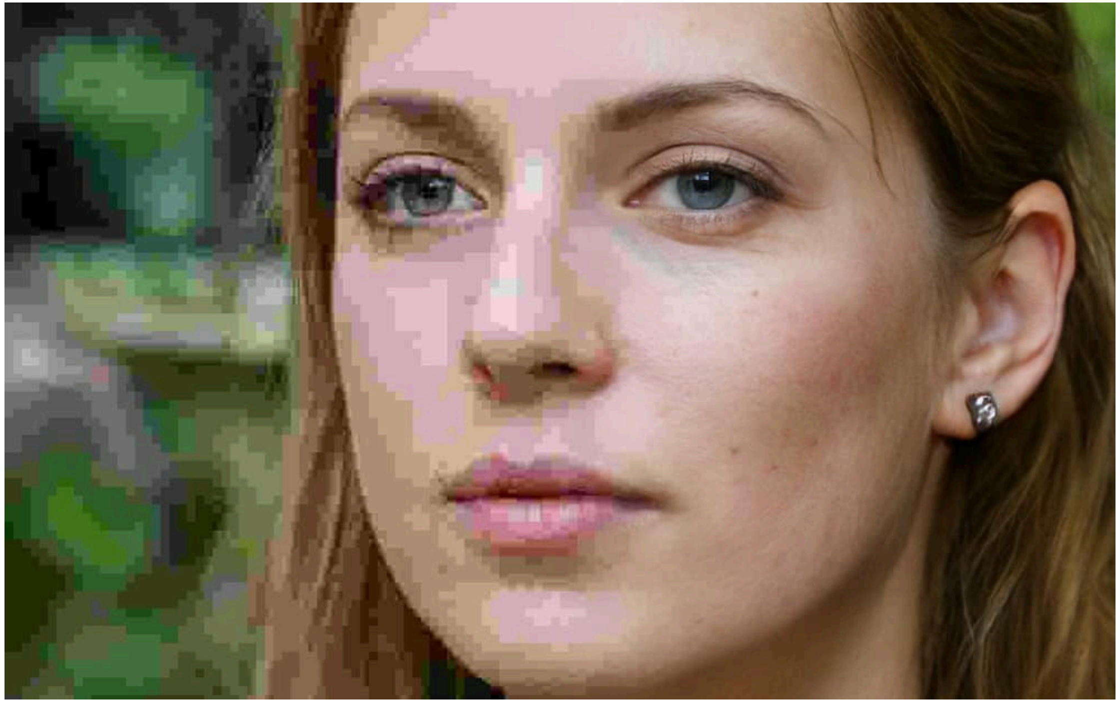
Compression may be **lossy**

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 Models
6. Coding

Agenda

1. Review
2. Compression
3. **Autoencoders**
4. Applications
5. Variational Models
6. Coding

Autoencoders

The encoders and decoders for images, videos, etc are very complex

Autoencoders

The encoders and decoders for images, videos, etc are very complex

Neural networks can represent any continuous function

Autoencoders

The encoders and decoders for images, videos, etc are very complex

Neural networks can represent any continuous function

Let us learn neural network encoders and decoders

Autoencoders

The encoders and decoders for images, videos, etc are very complex

Neural networks can represent any continuous function

Let us learn neural network encoders and decoders

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

Autoencoders

The encoders and decoders for images, videos, etc are very complex

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

The encoders and decoders for images, videos, etc are very complex

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

The encoders and decoders for images, videos, etc are very complex

Neural networks can represent any continuous function

Let us learn neural network encoders and decoders

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

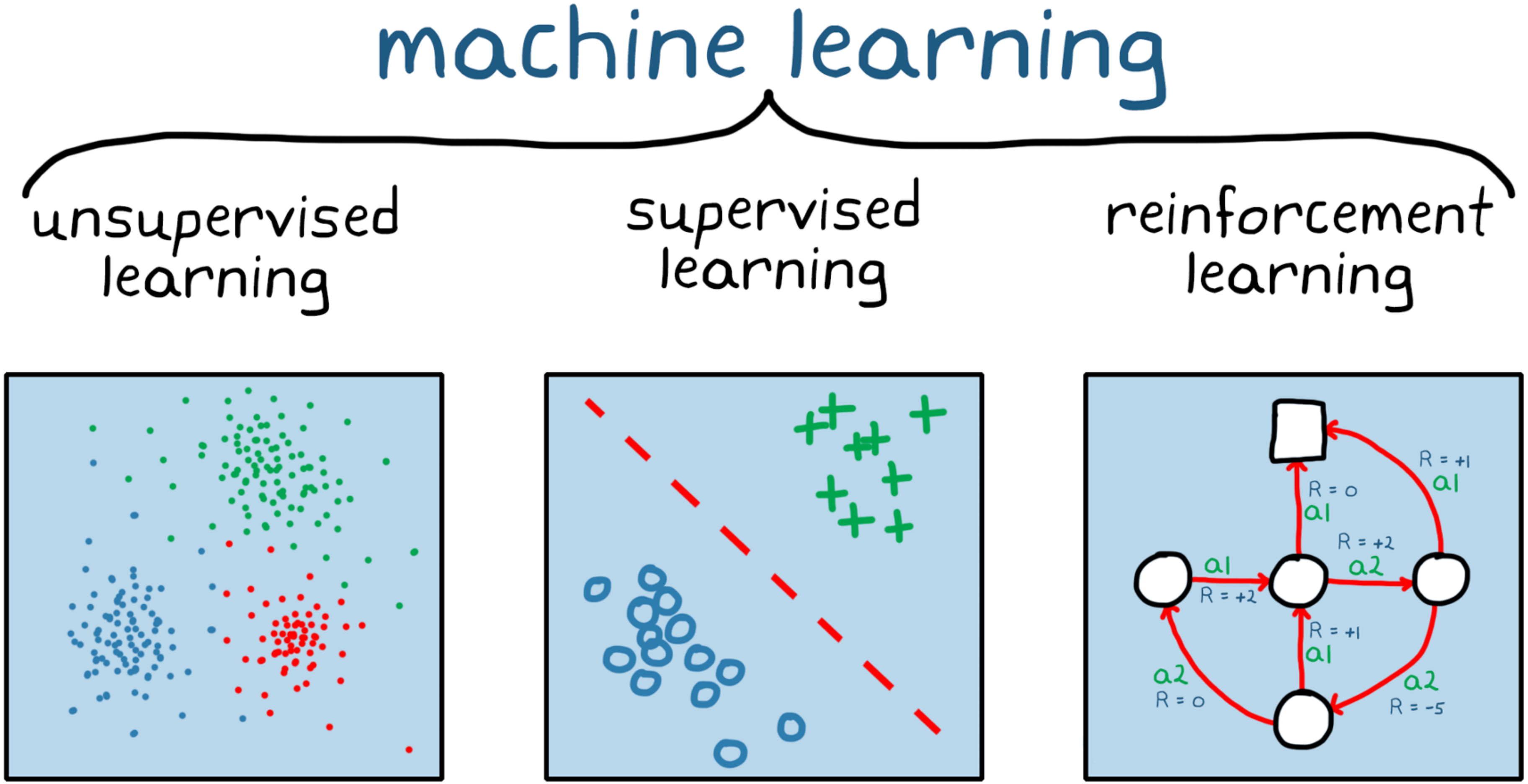
$$f^{-1} : Z \times \Theta \mapsto X$$

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**

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

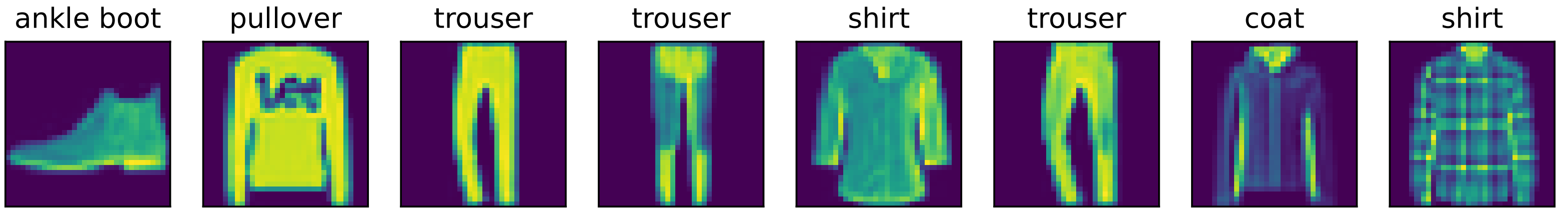
The training algorithm may generate labels

Autoencoders

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

Autoencoders

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

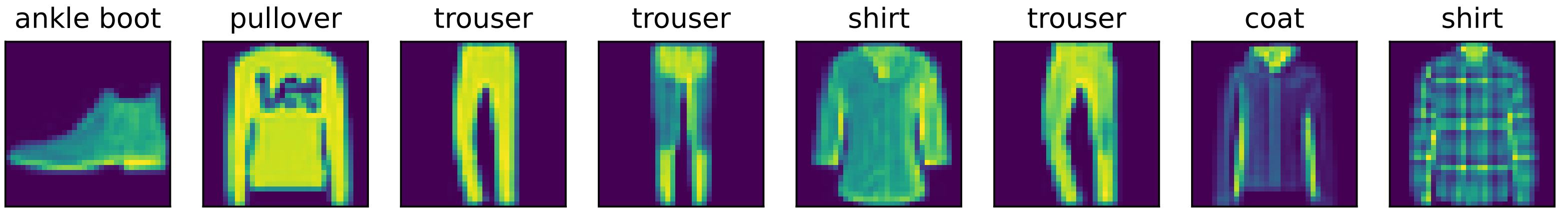


$$d_x : 28 \times 28 \qquad d_z : 4$$

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

Autoencoders

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



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

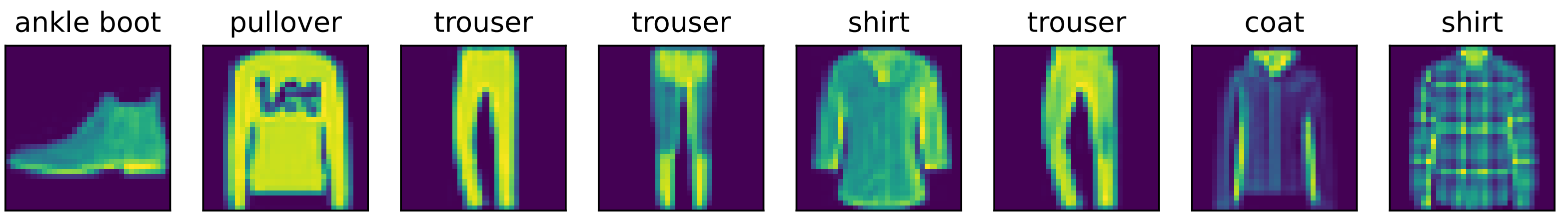
$X : [0, 1]^{d_x}$ $\quad$ $Z : \mathbb{R}^{d_z}$

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

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

Autoencoders

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



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

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

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

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

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

Autoencoders

Let us find f, then find the inverse

Autoencoders

Let us find f, then find the inverse

Start with a perceptron

Autoencoders

Let us find f, then find the inverse

Start with a perceptron

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

Autoencoders

Let us find f, then find the inverse

Start with a perceptron

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

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

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

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

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

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

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

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

Autoencoders

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

Autoencoders

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

Question: Any issues?

Hint: What if $d_x \neq d_z$?

Autoencoders

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

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

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

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

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

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

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

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

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

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

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

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

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

Autoencoders

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

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

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

Autoencoders

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

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

Autoencoders

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

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

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

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 Models
6. Coding

Agenda

1. Review
2. Compression
3. Autoencoders
4. **Applications**
5. Variational Models
6. Coding

Applications

We can use autoencoders for other tasks too

Applications

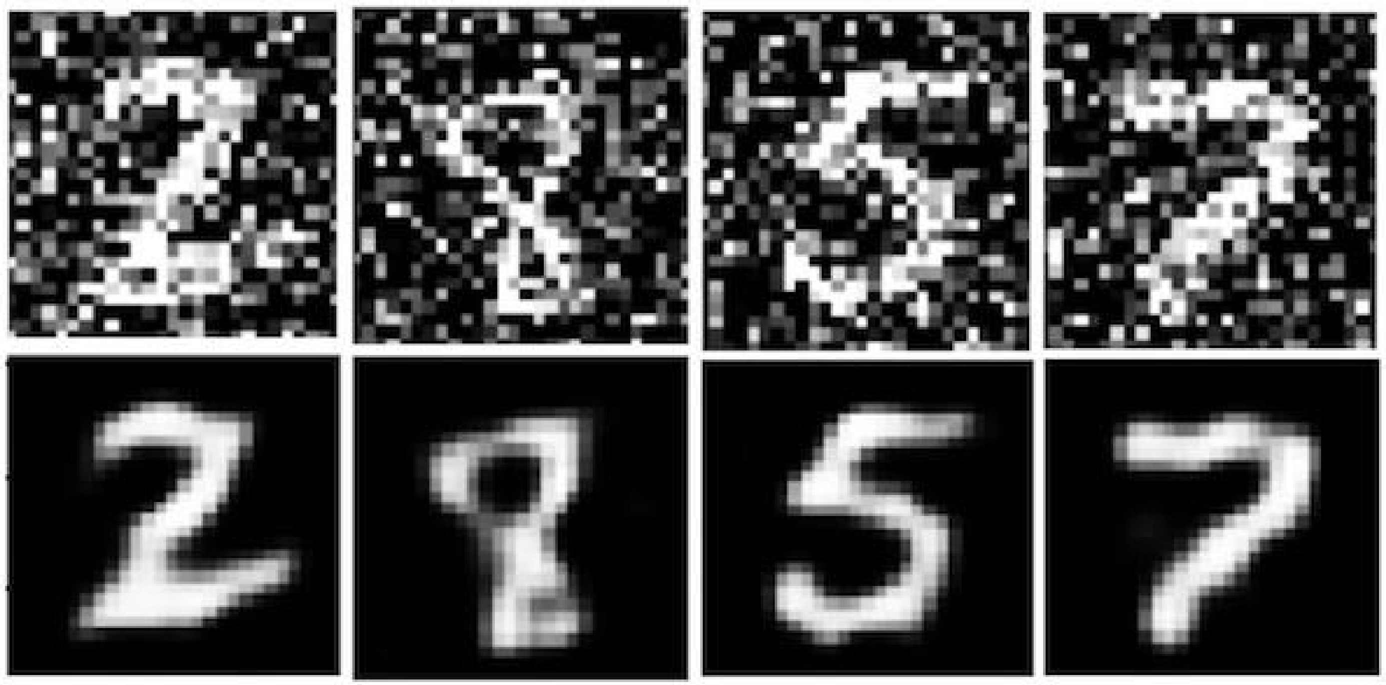
We can use autoencoders for other tasks too

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

Applications

We can use autoencoders for other tasks too

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



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 $\quad \boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

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

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

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

Applications

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

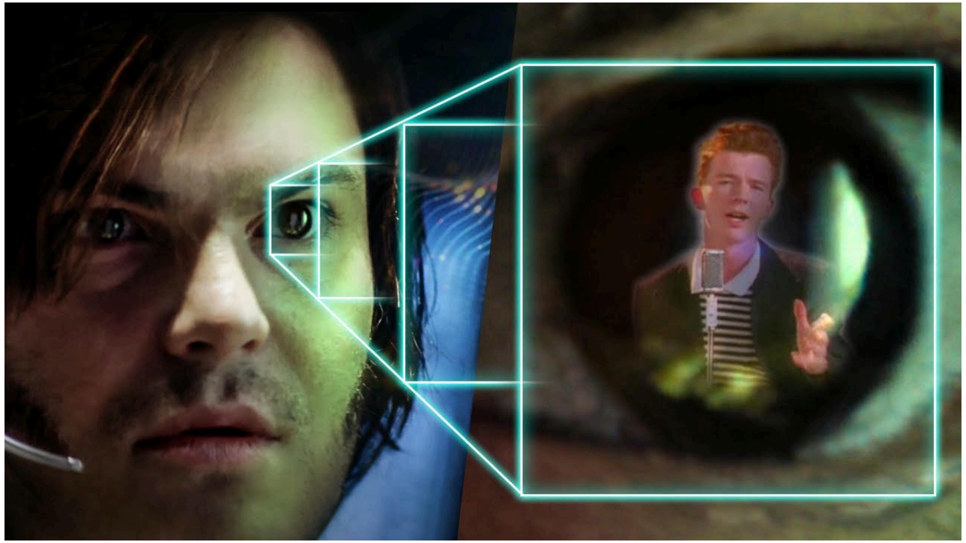
$$\mathcal{L}(\boldsymbol{X}, \boldsymbol{\theta}) = \sum_{i=1}^{n} \sum_{j=1}^{d_x} \left(x_{[i],j} - f^{-1}\left(f\left(\text{blur}\left(\boldsymbol{x}_{[i]} + \boldsymbol{\varepsilon}\right), \boldsymbol{\theta}_e\right), \boldsymbol{\theta}_d\right)_j \right)^2$$

Applications

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

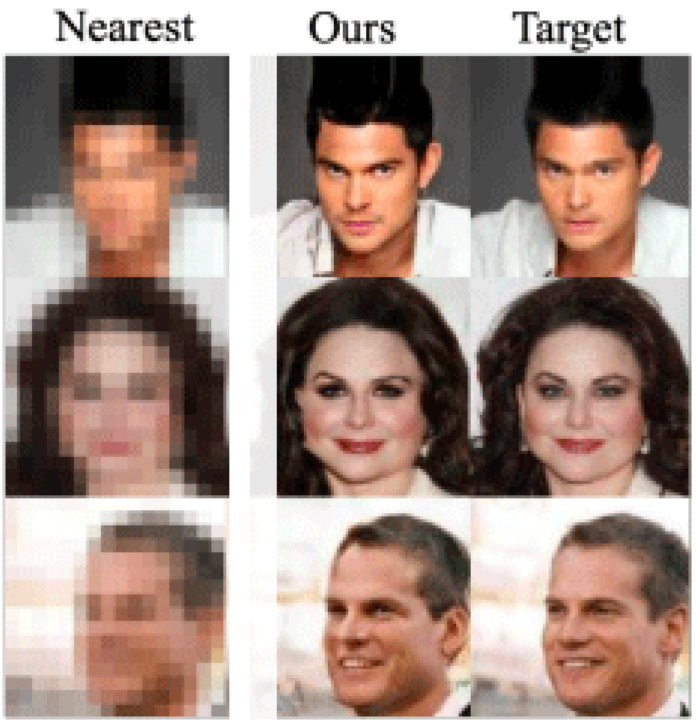
Applications

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

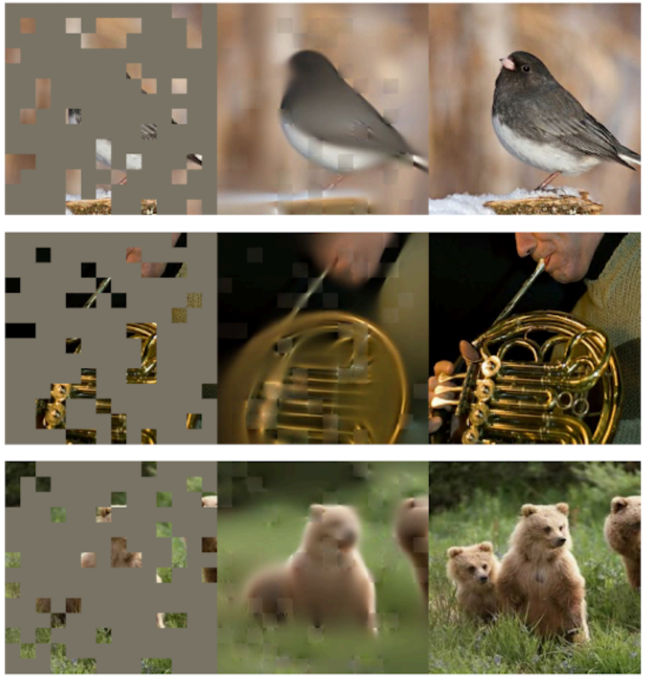
We can even hide parts of the image

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

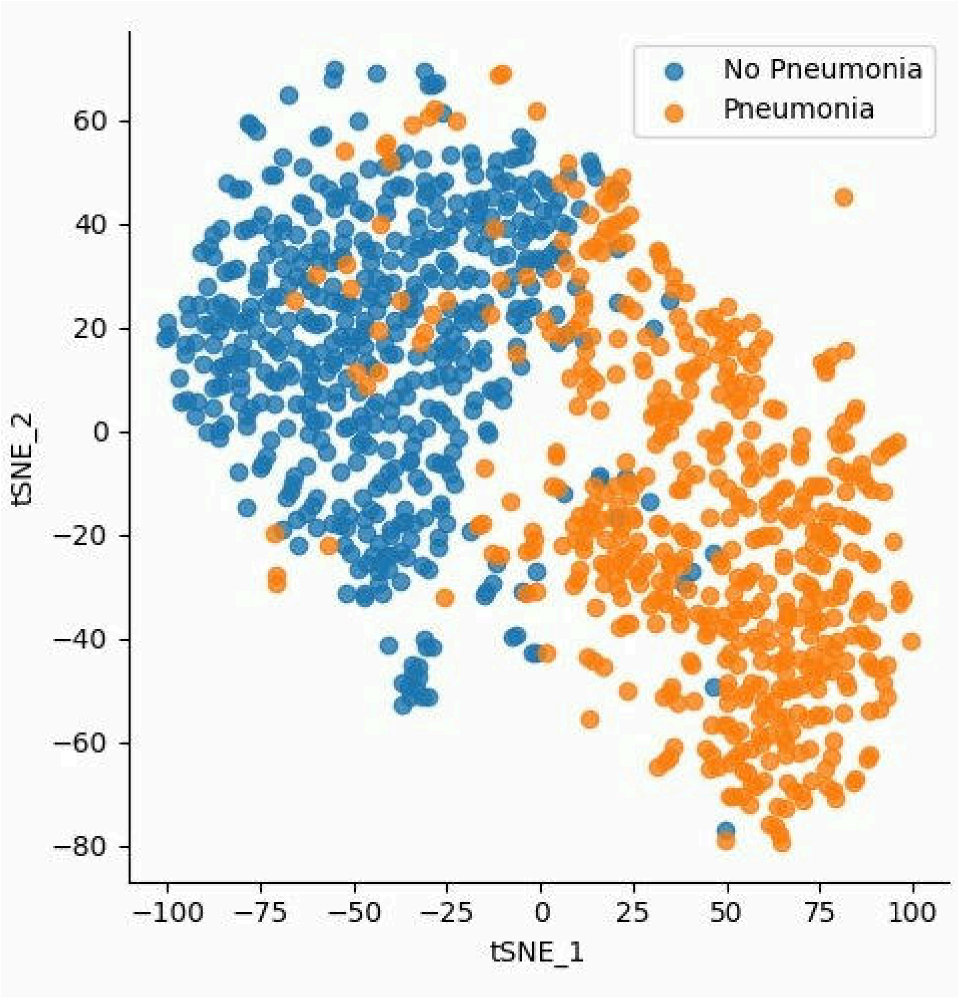
What is happening here? How can these models do this?

Applications

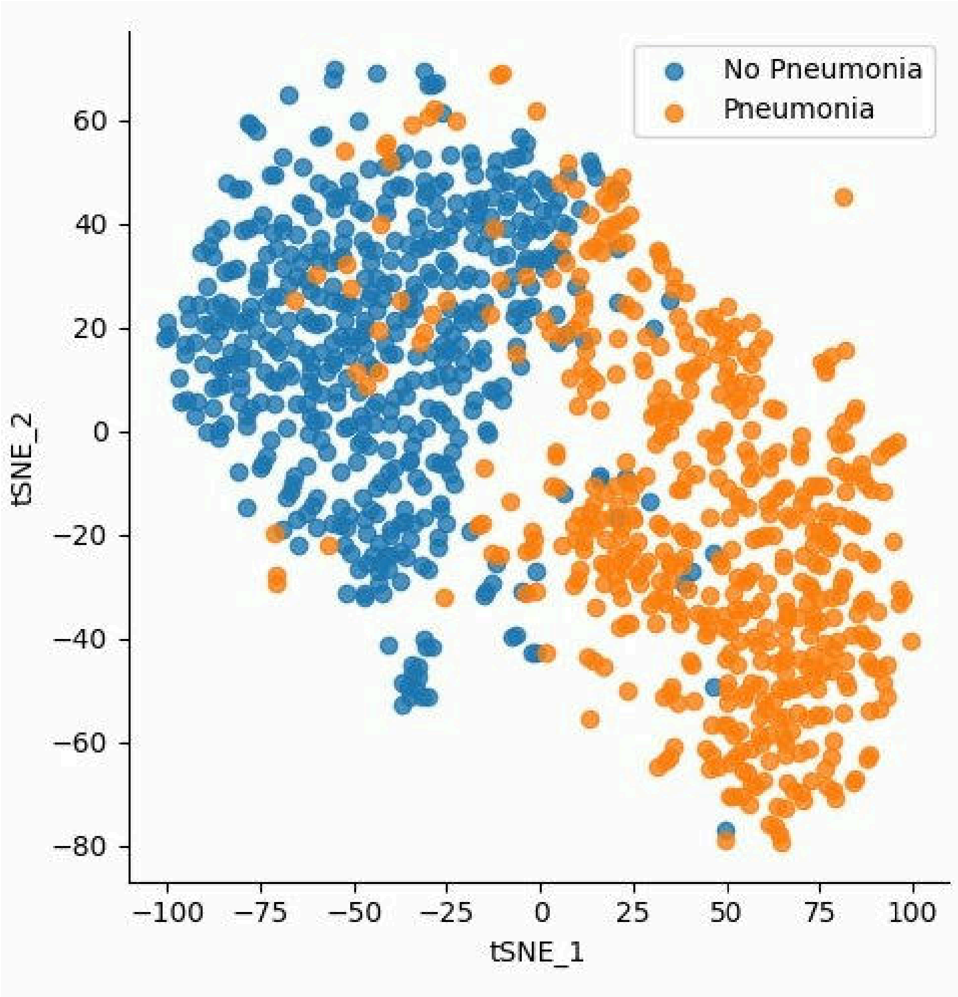
What is happening here? How can these models do this?

Autoencoders learn the structure of the dataset

Applications

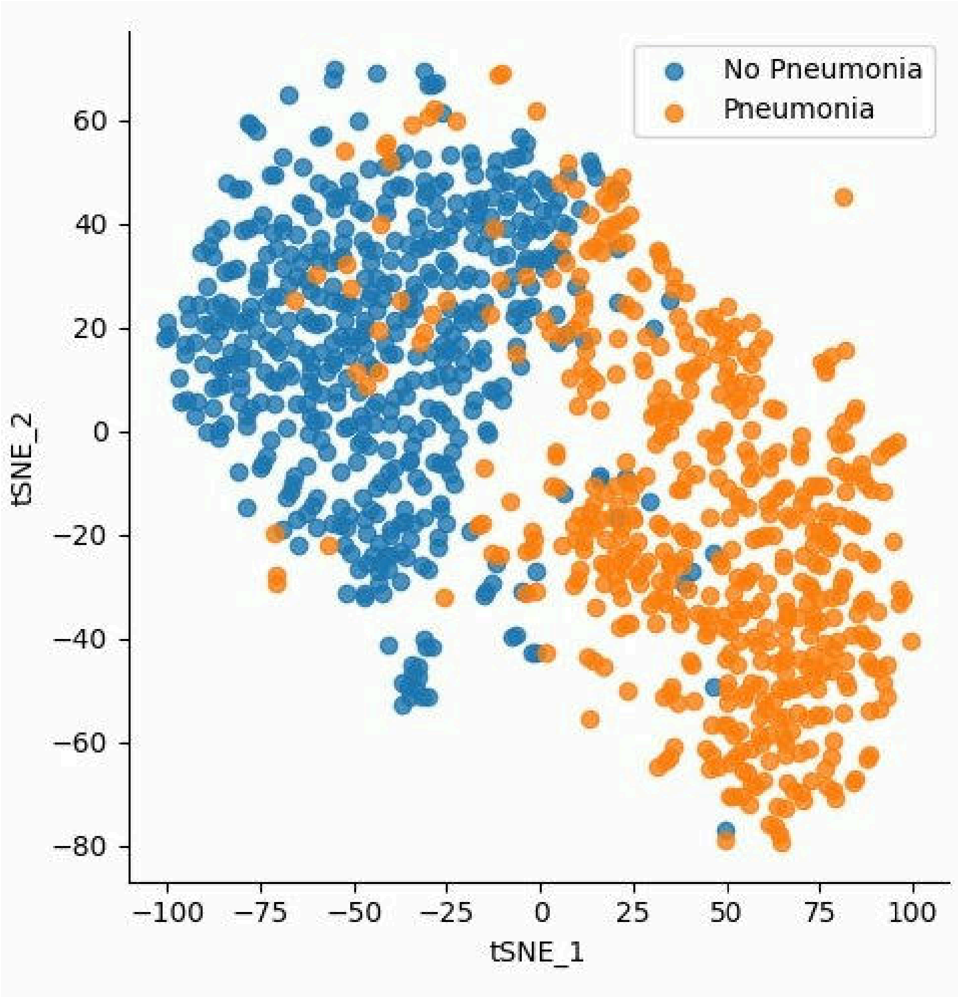


Applications



X : Pictures of human lungs

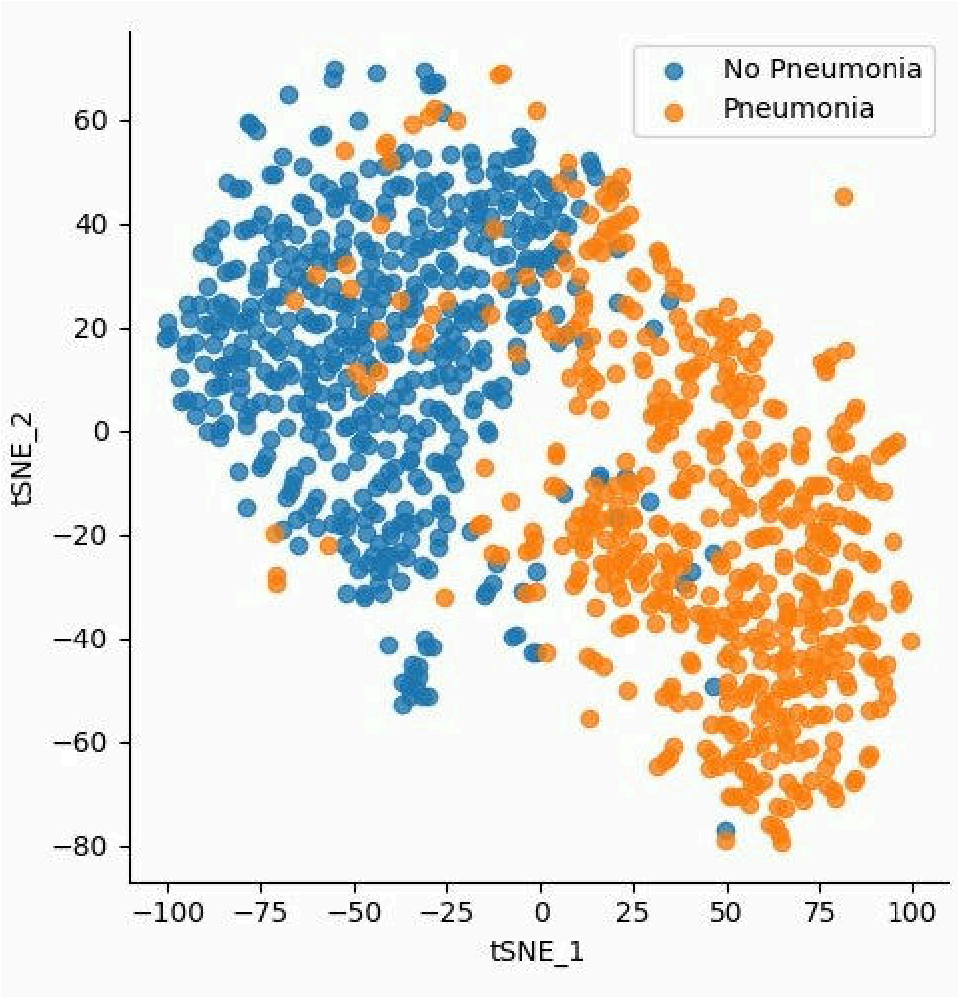
Applications



X : Pictures of human lungs

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

Applications

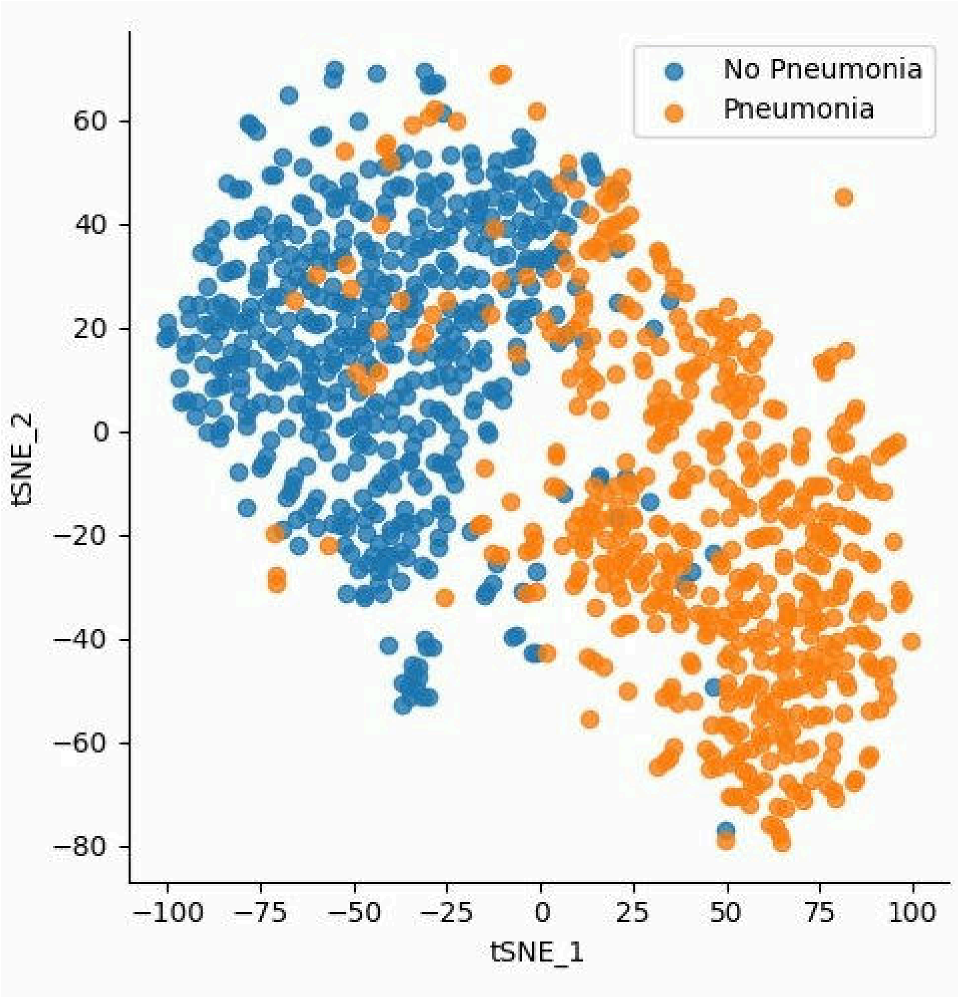


X : Pictures of human lungs

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

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 pictures from our world, then the autoencoders learn the structure of the world

Applications

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

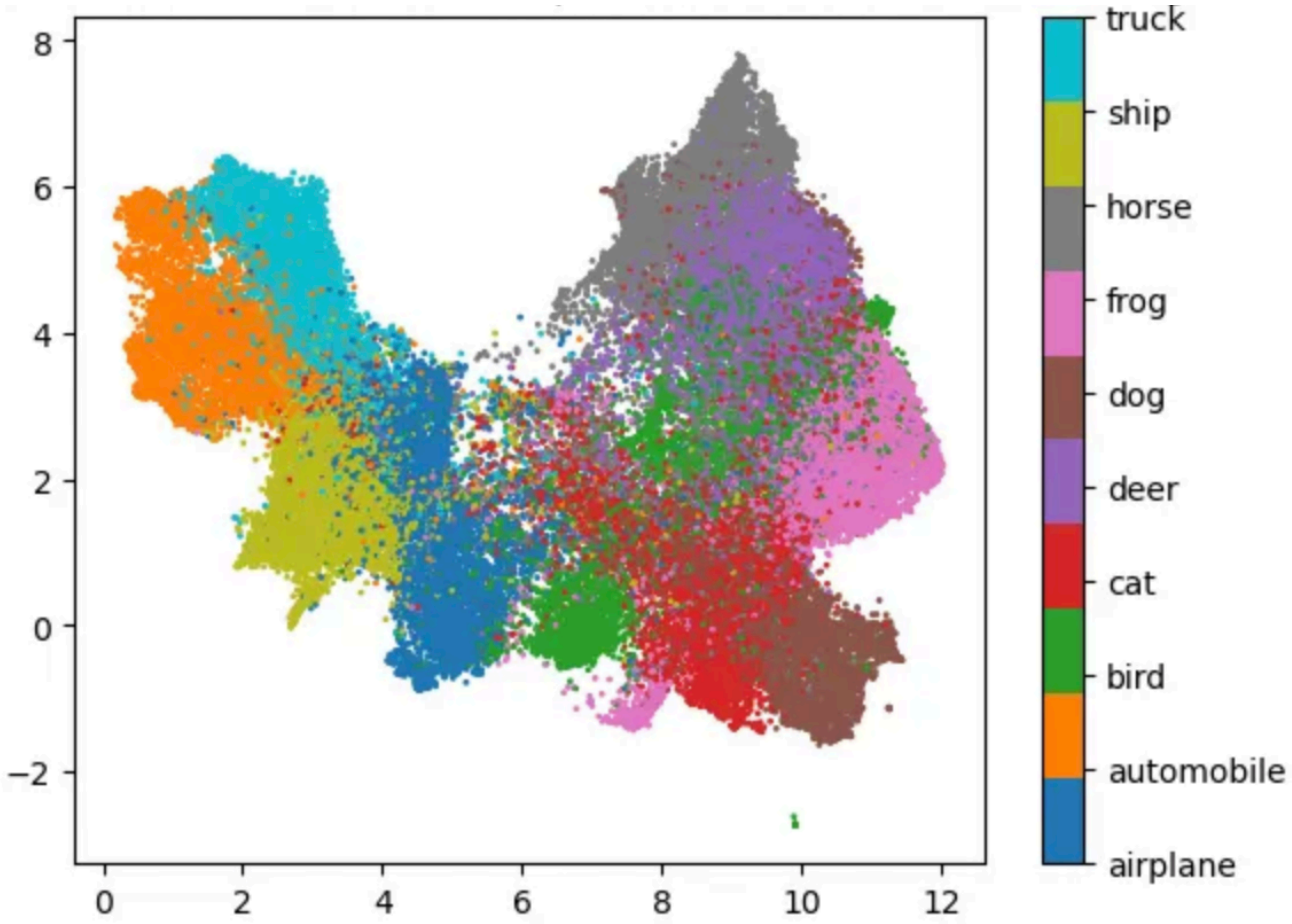
Applications

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

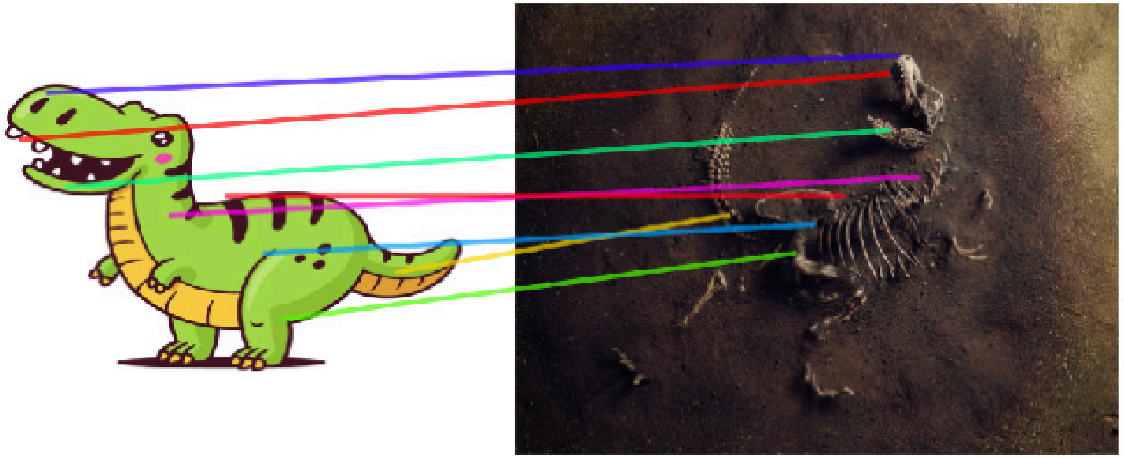
Nobody tells them what a dog or cat is

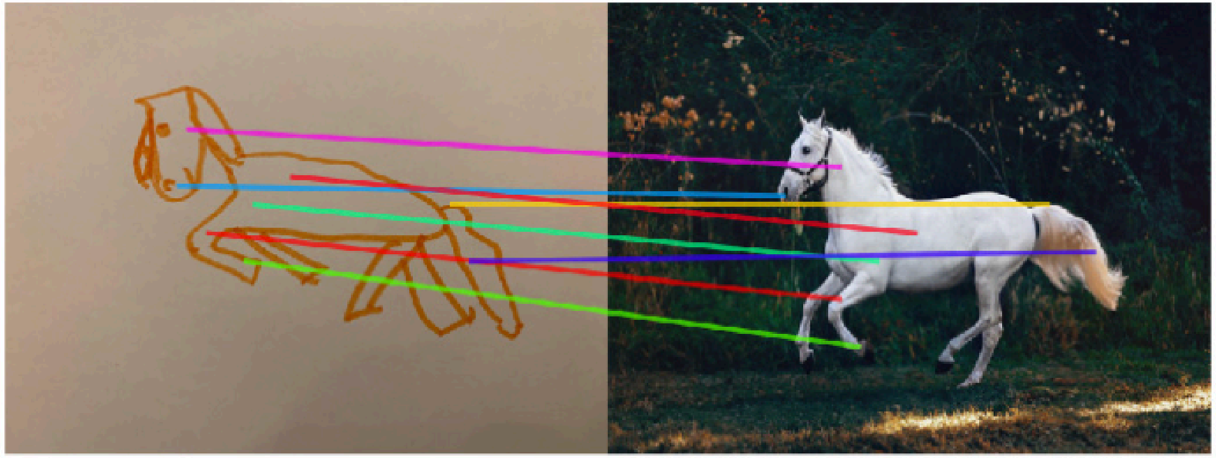
They learn that on their own

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

Clearly, these networks can understand our world

Agenda

1. Review
2. Compression
3. Autoencoders
4. **Applications**
5. Variational Models
6. Coding

Agenda

1. Review
2. Compression
3. Autoencoders
4. Applications
5. **Variational Models**
6. Coding

Variational Models

Variational Models

These pictures were created by an autoencoder

Variational Models

These pictures were created by an autoencoder

But these people do not exist!

Variational Models

Autoencoders are useful for compression and denoising

Variational Models

Autoencoders are useful for compression and denoising

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

Variational Models

Autoencoders are useful for compression and denoising

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

A generative model is a "creative" model that creates new data

Variational Models

Autoencoders are useful for compression and denoising

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

A generative model is a "creative" model that creates new data

- Generate pictures of people that do not exist

Variational Models

Autoencoders are useful for compression and denoising

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

A generative model is a "creative" model that creates new data

- Generate pictures of people that do not exist
- Write a new book

Variational Models

Autoencoders are useful for compression and denoising

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

A generative model is a "creative" model that creates new data

- Generate pictures of people that do not exist
- Write a new book
- Create new proteins

Variational Models

Autoencoders are useful for compression and denoising

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

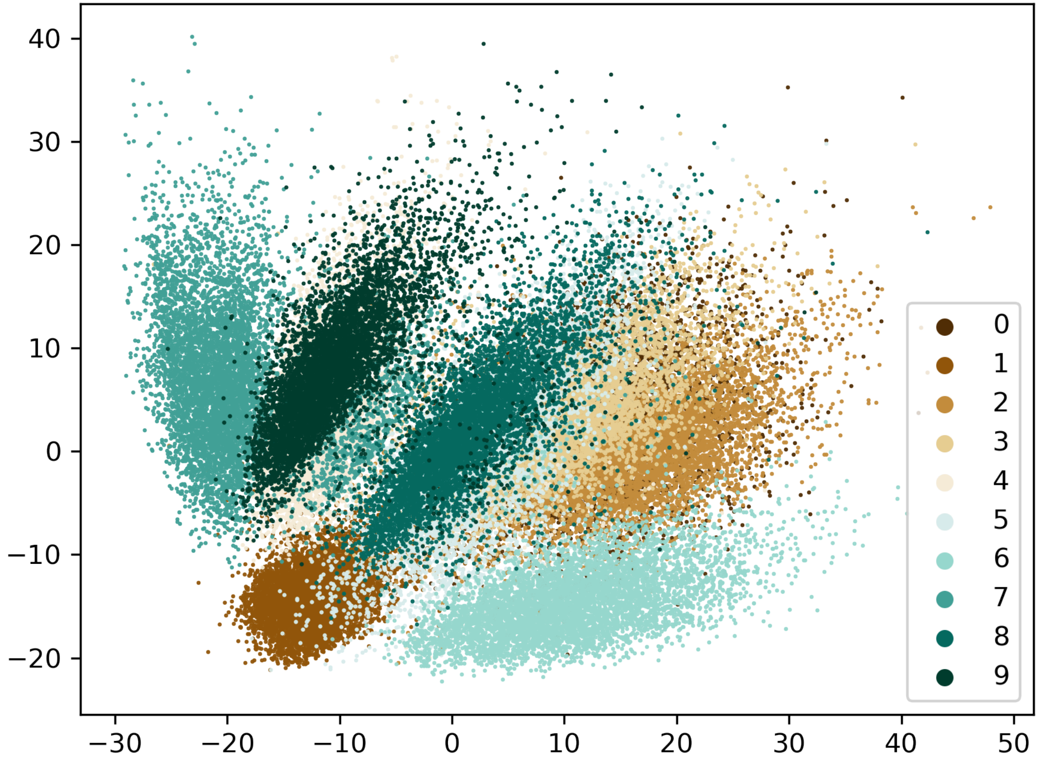
A generative model is a “creative” model that creates new data

- Generate pictures of people that do not exist
- Write a new book
- Create new proteins

How does this work?

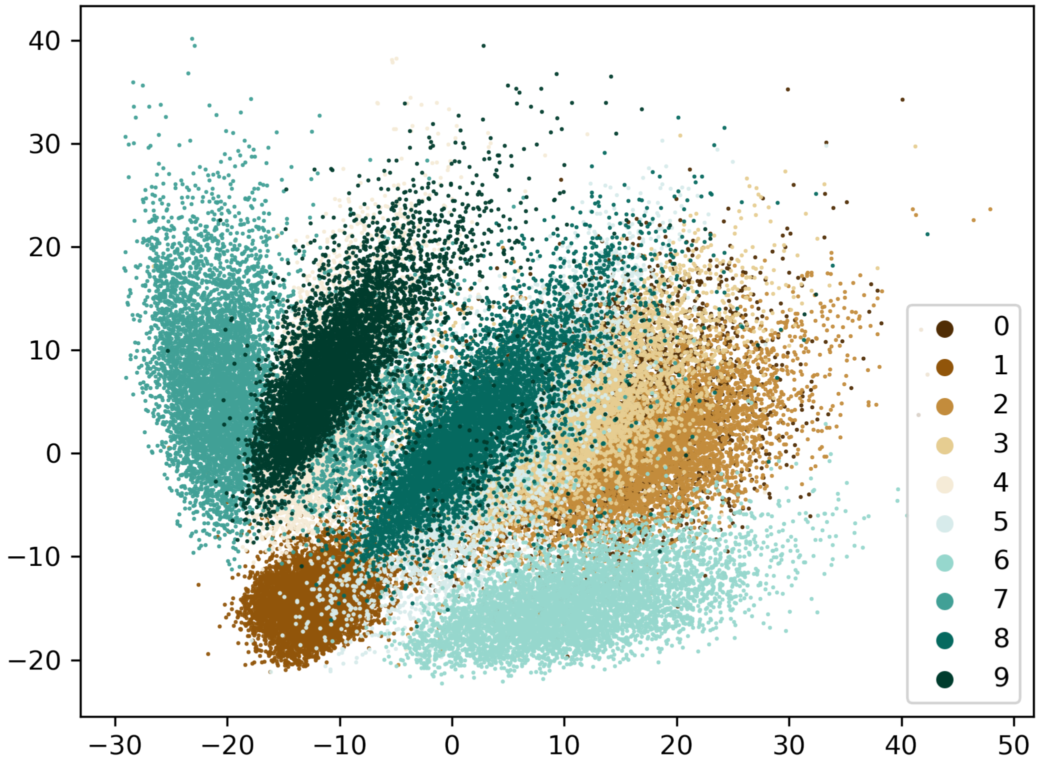
Variational Models

Latent space Z after training on the clothes dataset with $d_z = 2$



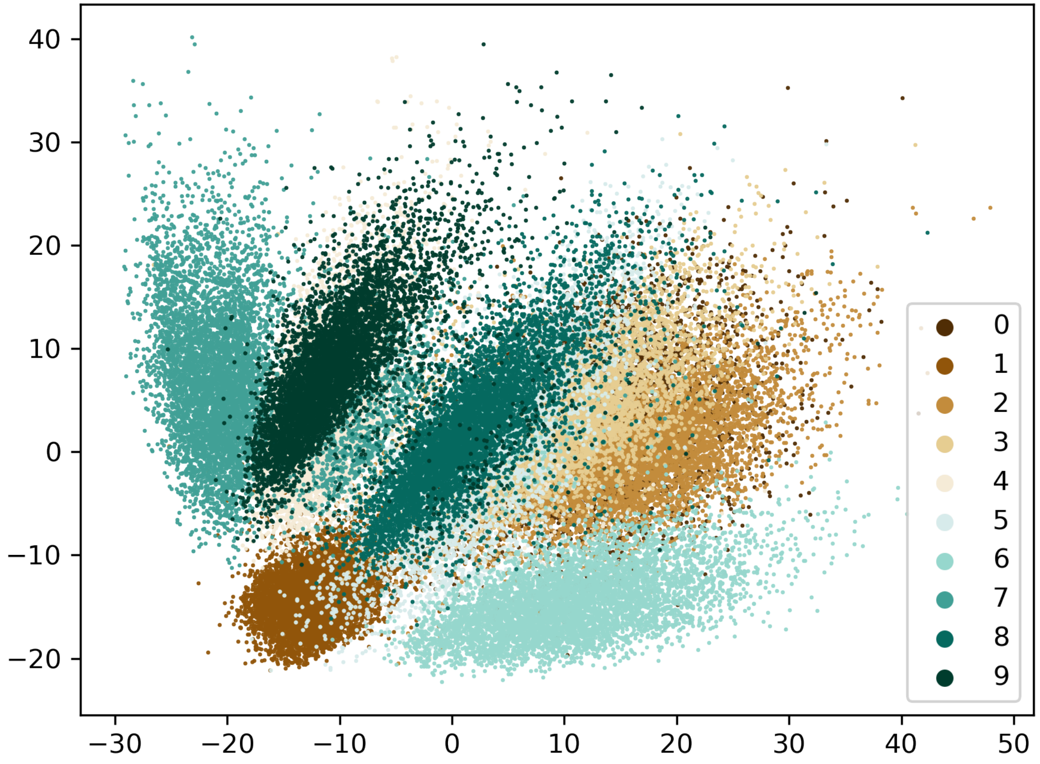
Variational Models

Latent space Z after training on the clothes dataset with $d_z = 2$

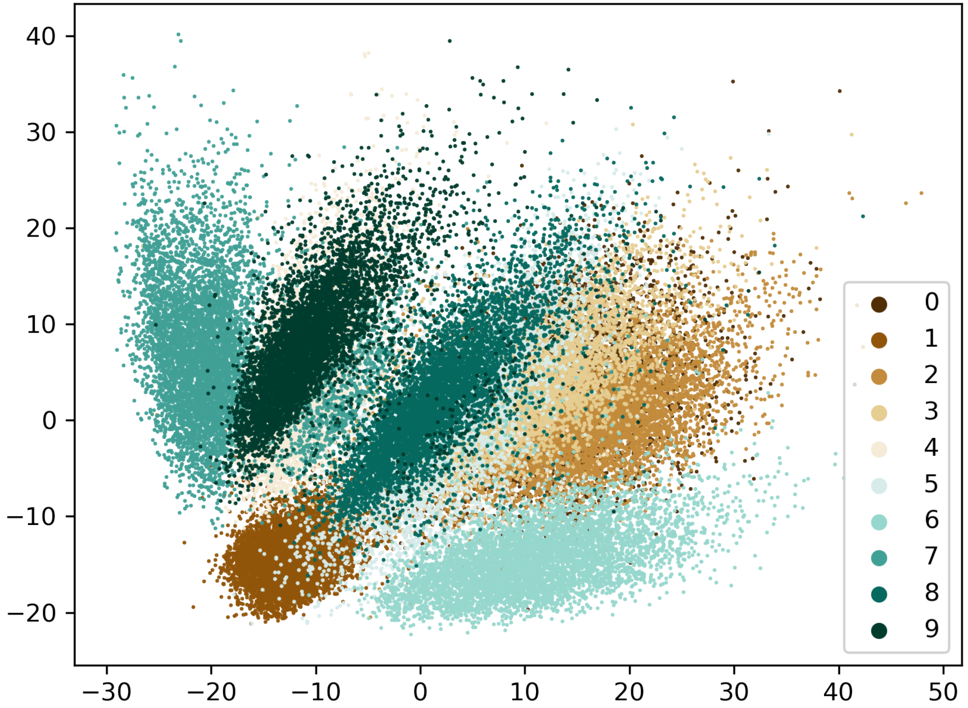


Variational Models

What happens if we decode a new point?

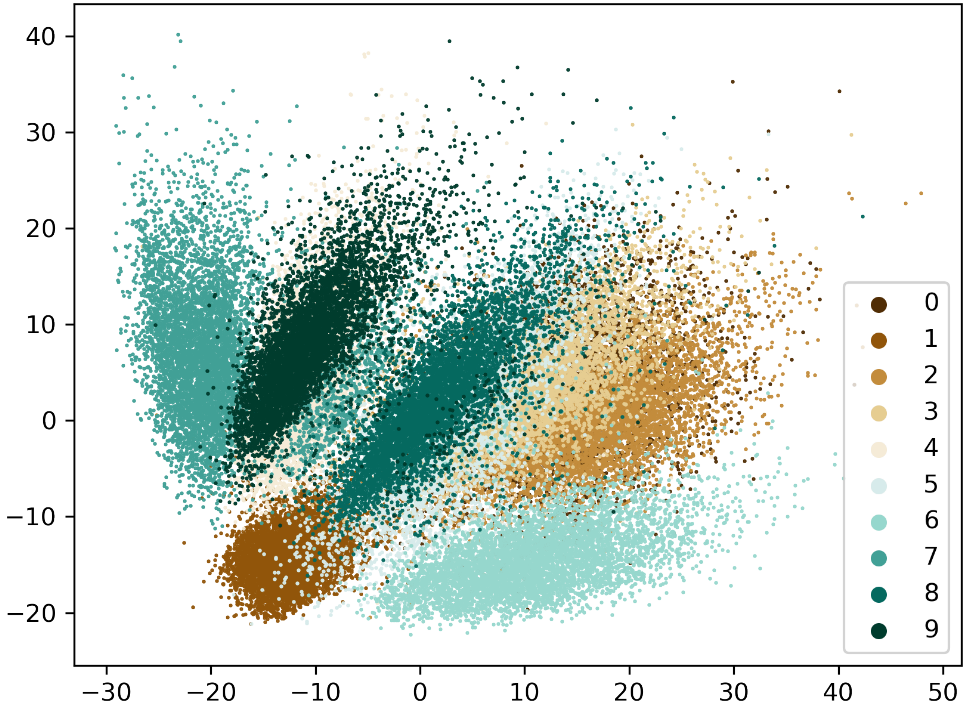


Variational Models



Autoencoder generative model:

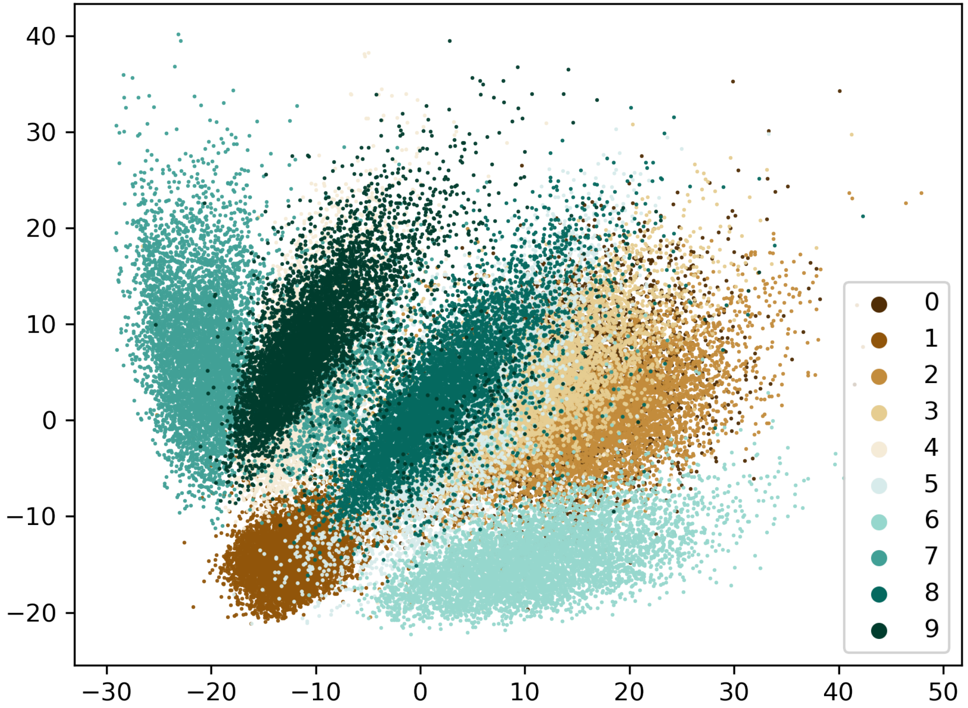
Variational Models



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 Models

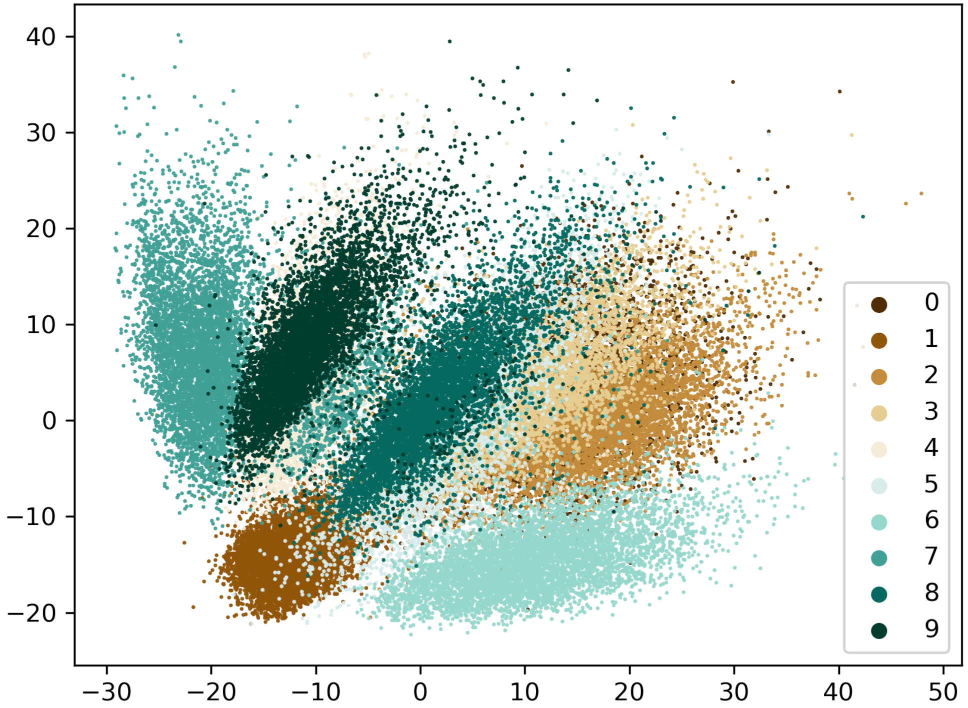


Autoencoder generative model:

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

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

Variational Models



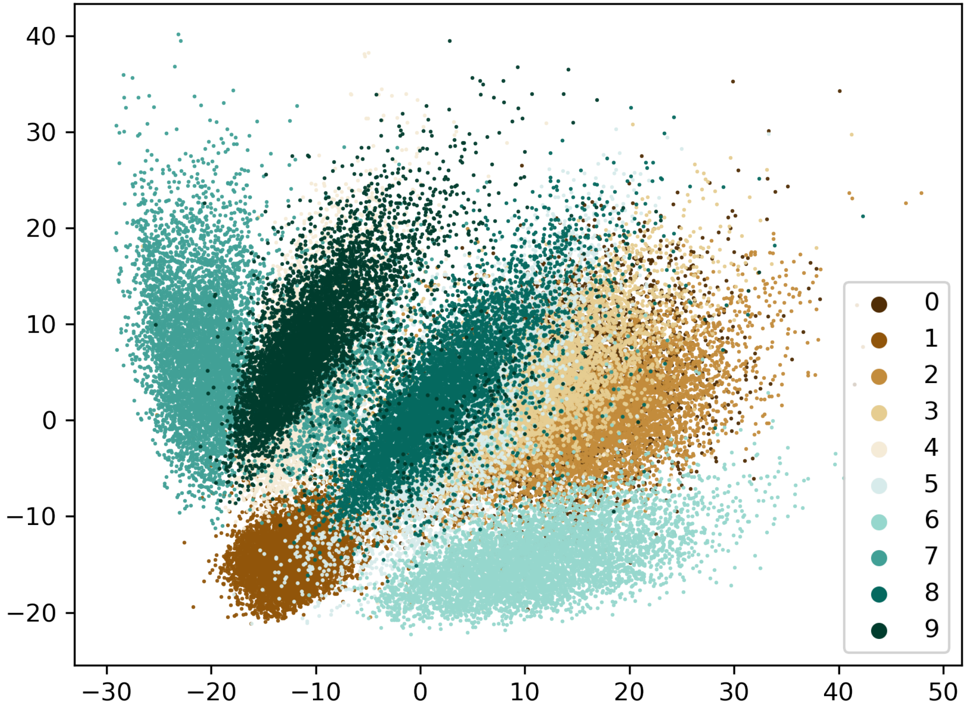
Autoencoder generative model:

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 Models



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 $\boldsymbol{z}_{[k]}$

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

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

Variational Models

Variational Models

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

Variational Models

But there is a problem

Variational Models

But there is a problem

As d_z increases, it becomes difficult to find useful parts of latent space

Variational Models

But there is a problem

As d_z increases, it becomes difficult to find useful parts of latent space

For large d_z, similar inputs map to $\boldsymbol{z}$ that are very far from each other

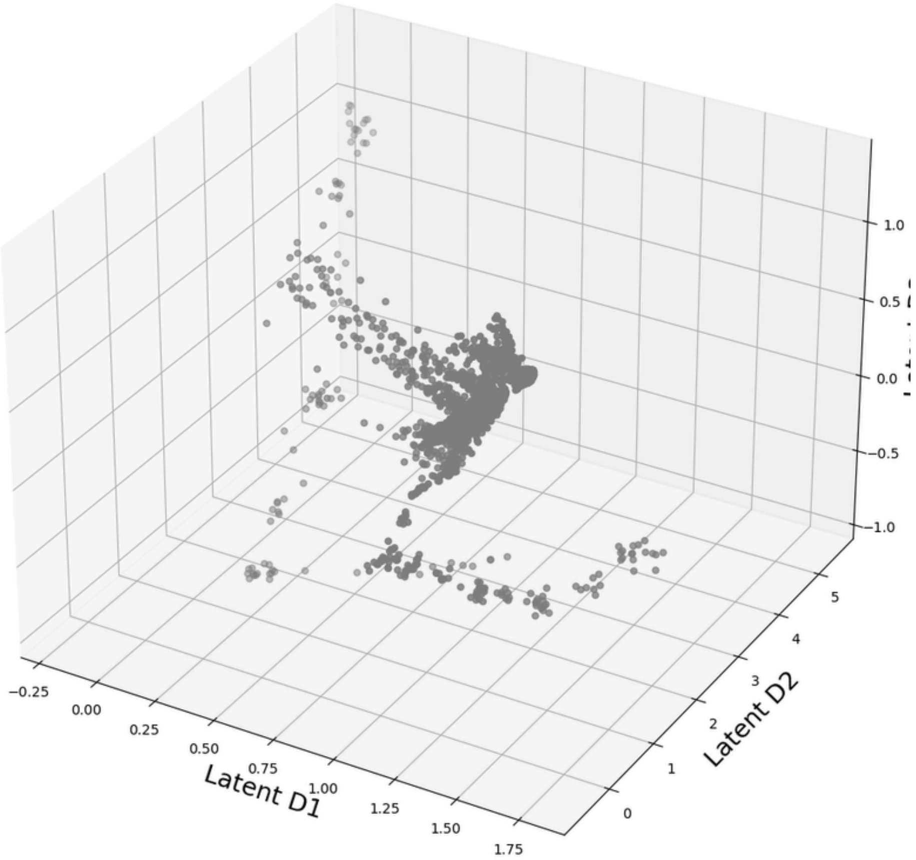
Variational Models

But there is a problem

As d_z increases, it becomes difficult to find useful parts of latent space

For large d_z, similar inputs map to $\boldsymbol{z}$ that are very far from each other

Points $\boldsymbol{z} + \boldsymbol{\varepsilon}$ decode into garbage



Variational Models

Question: What can we do?

Variational Models

Question: What can we do?

Answer: Force the points to be close together!

Variational Models

Question: What can we do?

Answer: Force the points to be close together!

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

Variational Models

VAE discovered by Diederik Kingma (also adam optimizer)

Variational Models

VAE discovered by Diederik Kingma (also adam optimizer)

Variational Models

Variational autoencoders (VAEs) do three things:

Variational Models

Variational autoencoders (VAEs) do three things:

1. Make it easy to sample random z

Variational Models

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 Models

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 Models

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 Models

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

Variational Models

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

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

Variational Models

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

Variational Models

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

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

Variational Models

If $\boldsymbol{z}_{[i]}, ..., \boldsymbol{z}_{[n]}$ 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 Models

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

Variational Models

So how do we ensure that $\boldsymbol{z}_{[i]}, ..., \boldsymbol{z}_{[n]}$ 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 Models

So how do we ensure that $\boldsymbol{z}_{[i]}, ..., \boldsymbol{z}_{[n]}$ 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 Models

Introduce one more Bayesian concept called **marginalization**

Variational Models

Introduce one more Bayesian concept called **marginalization**

Assume we have the probability of two events

$$P(A \cap B) = P(A, B)$$

Variational Models

Introduce one more Bayesian concept called **marginalization**

Assume we have the probability of two events

$$P(A \cap B) = P(A, B)$$

From Bayes rule

$$P(A \cap B) = P(A \mid B) P(B)$$

Variational Models

Introduce one more Bayesian concept called **marginalization**

Assume we have the probability of two events

$$P(A \cap B) = P(A, B)$$

From Bayes rule

$$P(A \cap B) = P(A \mid B) P(B)$$

We can find $P(A)$ by marginalizing out B

Variational Models

Introduce one more Bayesian concept called **marginalization**

Assume we have the probability of two events

$$P(A \cap B) = P(A, B)$$

From Bayes rule

$$P(A \cap B) = P(A \mid B)P(B)$$

We can find $P(A)$ by marginalizing out B

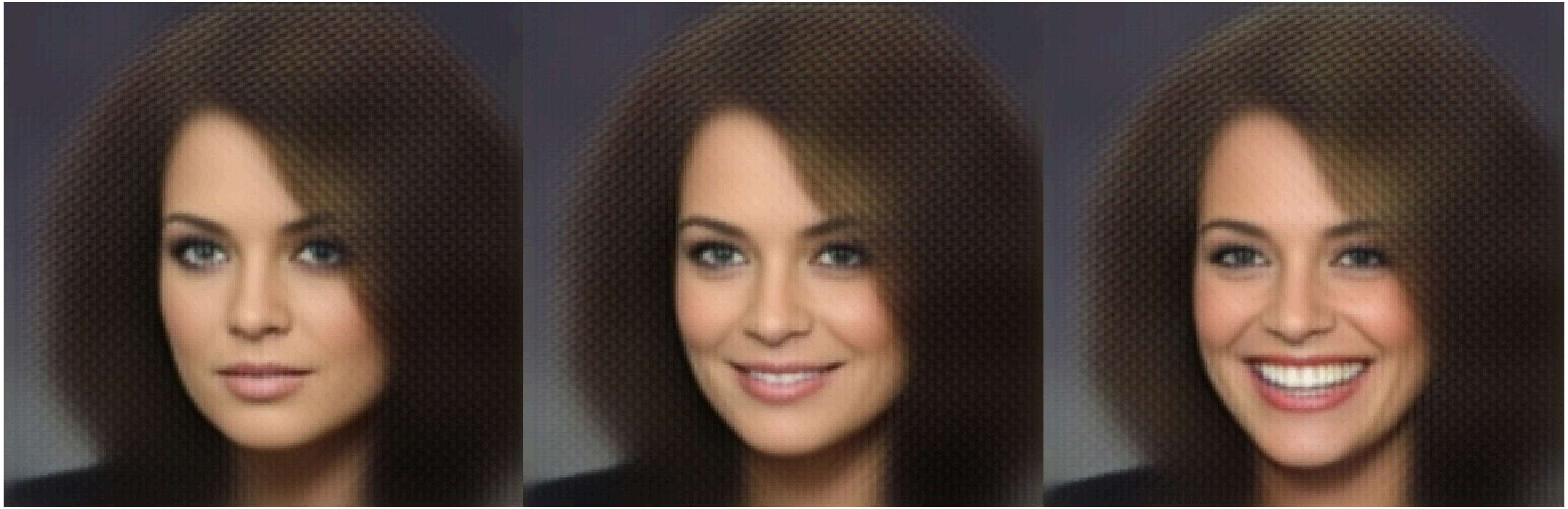
$$P(A) = \int_B P(A \mid B)P(B) \, \mathrm{d}B$$

Variational Models

Key idea: There is some latent variable z which generates data x

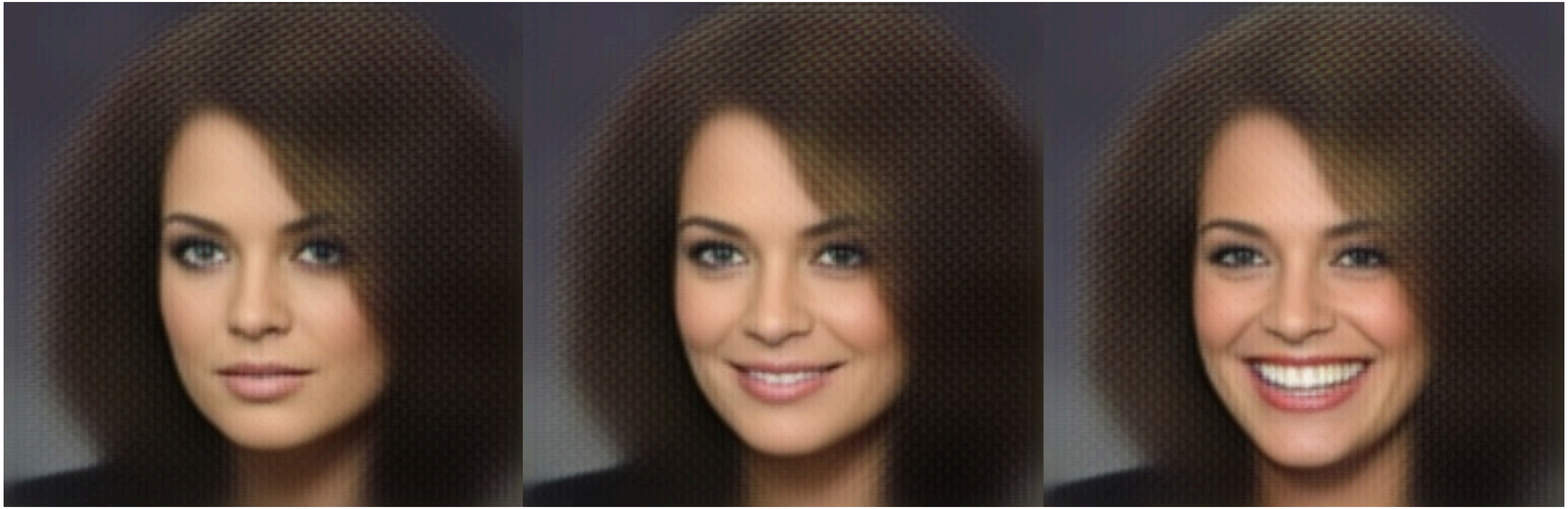
Variational Models

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

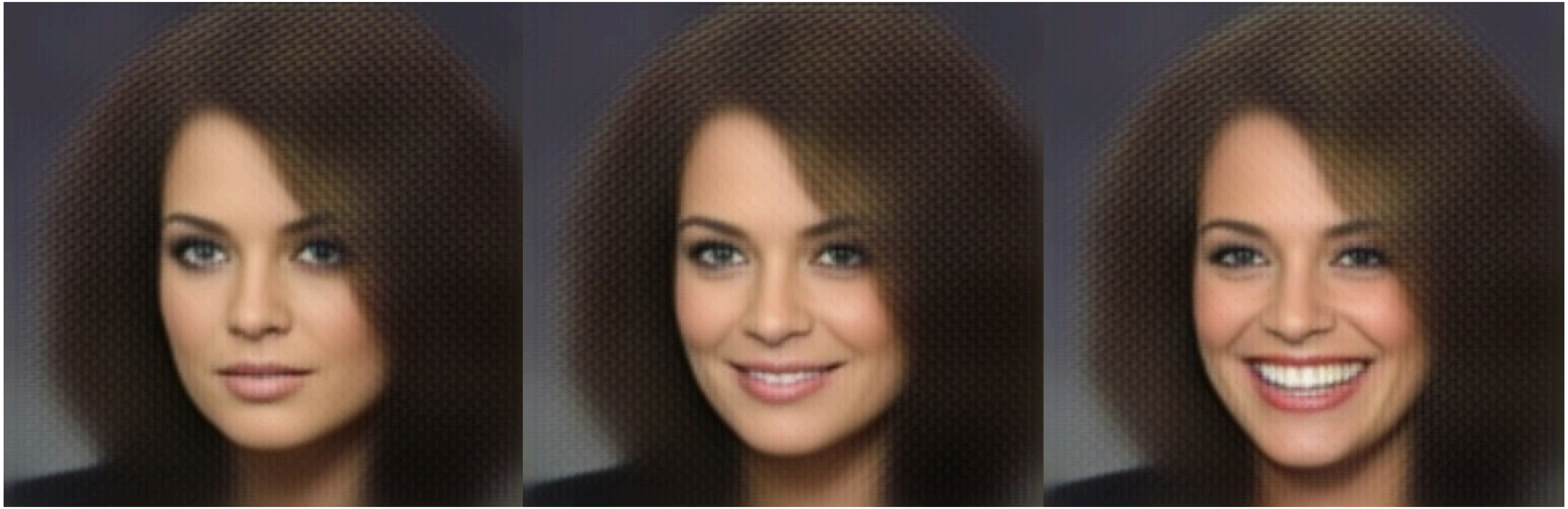
x : 

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

Variational Models

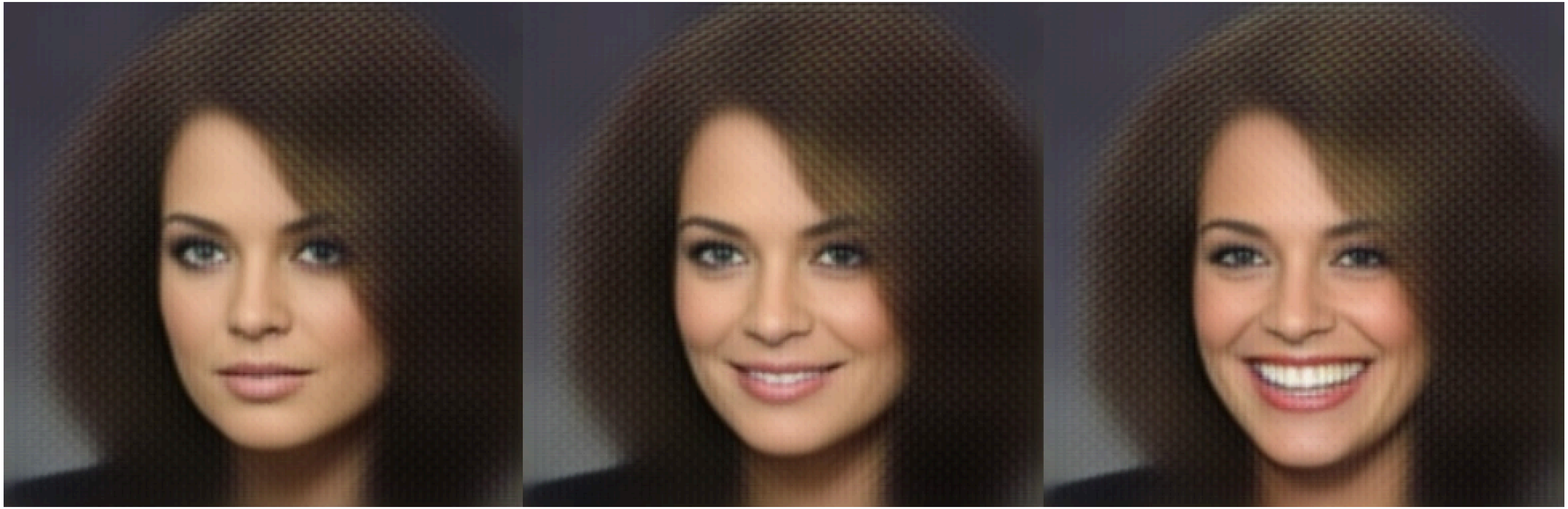


Variational Models



We can only see $\boldsymbol{x}$, we cannot directly observe $\boldsymbol{z}$

Variational Models



We can only see $\boldsymbol{x}$, we cannot directly observe $\boldsymbol{z}$

Given the image $\boldsymbol{x}$, what is the probability that the person is smiling?
$P(\boldsymbol{z} \mid \boldsymbol{x})$

Variational Models

How can we find $P(\boldsymbol{z} \mid \boldsymbol{x})$?

Variational Models

How can we find $P(\boldsymbol{z} \mid \boldsymbol{x})$?

Use Bayes rule

$$P(\boldsymbol{z} \mid \boldsymbol{x}) = \frac{P(\boldsymbol{x} \mid \boldsymbol{z}) P(\boldsymbol{z})}{P(\boldsymbol{x})}$$

Variational Models

How can we find $P(\boldsymbol{z} \mid \boldsymbol{x})$?

Use Bayes rule

$$P(\boldsymbol{z} \mid \boldsymbol{x}) = \frac{P(\boldsymbol{x} \mid \boldsymbol{z})P(\boldsymbol{z})}{P(\boldsymbol{x})}$$

How do we find $P(\boldsymbol{x})$? Marginalize

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

$$P(\boldsymbol{x}) = \int_{\boldsymbol{z}} P(\boldsymbol{x} \mid \boldsymbol{z})P(\boldsymbol{z})$$

Variational Models

$$P(\boldsymbol{z} \mid \boldsymbol{x}) = \frac{P(\boldsymbol{x} \mid \boldsymbol{z}) P(\boldsymbol{z})}{P(\boldsymbol{x})}$$

$$P(\boldsymbol{x}) = \int_{\boldsymbol{z}} P(\boldsymbol{x} \mid \boldsymbol{z}) P(\boldsymbol{z})$$

Variational Models

$$P(\boldsymbol{z} \mid \boldsymbol{x}) = \frac{P(\boldsymbol{x} \mid \boldsymbol{z})P(\boldsymbol{z})}{P(\boldsymbol{x})}$$

$$P(\boldsymbol{x}) = \int_{\boldsymbol{z}} P(\boldsymbol{x} \mid \boldsymbol{z})P(\boldsymbol{z})$$

$$P(\boldsymbol{z} \mid \boldsymbol{x}) = \frac{P(\boldsymbol{x} \mid \boldsymbol{z})P(\boldsymbol{z})}{\int_{\boldsymbol{z}} P(\boldsymbol{x} \mid \boldsymbol{z})P(\boldsymbol{z})}$$

Variational Models

$$P(\boldsymbol{z} \mid \boldsymbol{x}) = \frac{P(\boldsymbol{x} \mid \boldsymbol{z})P(\boldsymbol{z})}{P(\boldsymbol{x})} \qquad P(\boldsymbol{x}) = \int_{\boldsymbol{z}} P(\boldsymbol{x} \mid \boldsymbol{z})P(\boldsymbol{z})$$

$$P(\boldsymbol{z} \mid \boldsymbol{x}) = \frac{P(\boldsymbol{x} \mid \boldsymbol{z})P(\boldsymbol{z})}{\int_{\boldsymbol{z}} P(\boldsymbol{x} \mid \boldsymbol{z})P(\boldsymbol{z})}$$

This integral is intractable to compute!

Variational Models

$$P(\boldsymbol{z} \mid \boldsymbol{x}) = \frac{P(\boldsymbol{x} \mid \boldsymbol{z})P(\boldsymbol{z})}{P(\boldsymbol{x})} \qquad P(\boldsymbol{x}) = \int_{\boldsymbol{z}} P(\boldsymbol{x} \mid \boldsymbol{z})P(\boldsymbol{z})$$

$$P(\boldsymbol{z} \mid \boldsymbol{x}) = \frac{P(\boldsymbol{x} \mid \boldsymbol{z})P(\boldsymbol{z})}{\int_{\boldsymbol{z}} P(\boldsymbol{x} \mid \boldsymbol{z})P(\boldsymbol{z})}$$

This integral is intractable to compute!

Instead, we approximate this distribution using our encoder

$$f(\boldsymbol{x}, \boldsymbol{\theta}_e) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\theta}) \approx P(\boldsymbol{z} \mid \boldsymbol{x})$$

Remember, f outputs a **distribution**

Variational Models

We want to make $f(\boldsymbol{x}, \boldsymbol{\theta}_e)$ close to $P(\boldsymbol{z} \mid \boldsymbol{x})$

Variational Models

We want to make $f(\boldsymbol{x}, \boldsymbol{\theta}_e)$ close to $P(\boldsymbol{z} \mid \boldsymbol{x})$

Remember, it is intractable to find $P(\boldsymbol{z} \mid \boldsymbol{x})$

Variational Models

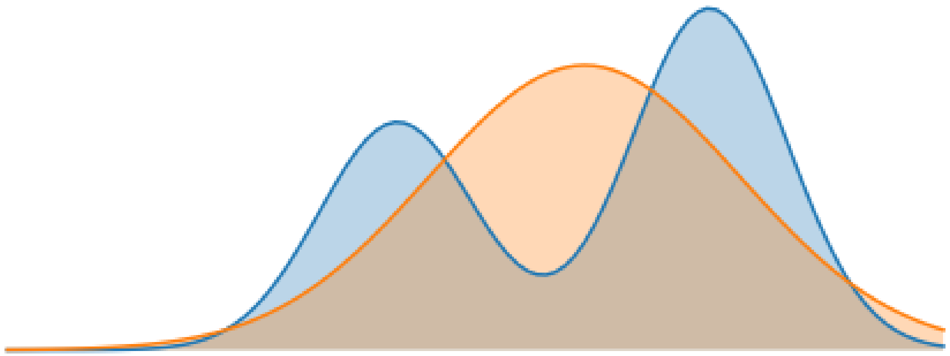
We want to make $f(\boldsymbol{x}, \boldsymbol{\theta}_e)$ close to $P(\boldsymbol{z} \mid \boldsymbol{x})$

Remember, it is intractable to find $P(\boldsymbol{z} \mid \boldsymbol{x})$

But let us continue and see if $P(\boldsymbol{z} \mid \boldsymbol{x})$ goes away

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

Answer: KL divergence



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

Variational Models

$$\mathrm{KL}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), P(\boldsymbol{z} \mid \boldsymbol{x}))$$

Variational Models

$$\mathrm{KL}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), P(\boldsymbol{z} \mid \boldsymbol{x}))$$

Through **magic**, we can write the KL divergence as

$$\mathrm{KL}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), P(\boldsymbol{z} \mid \boldsymbol{x})) = \underbrace{\log P(\boldsymbol{x} \mid \boldsymbol{z})}_{\text{Reconstruction error}} - \underbrace{\mathrm{KL}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), P(\boldsymbol{z}))}_{\text{Encoder and prior}}$$

Variational Models

$$\mathrm{KL}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), P(\boldsymbol{z} \mid \boldsymbol{x}))$$

Through **magic**, we can write the KL divergence as

$$\mathrm{KL}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), P(\boldsymbol{z} \mid \boldsymbol{x})) = \underbrace{\log P(\boldsymbol{x} \mid \boldsymbol{z})}_{\text{Reconstruction error}} - \underbrace{\mathrm{KL}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), P(\boldsymbol{z}))}_{\text{Encoder and prior}}$$

We can pick our prior distribution $P(\boldsymbol{z})$, let's pick

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

Variational Models

$$\mathrm{KL}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), P(\boldsymbol{z} \mid \boldsymbol{x}))$$

Through **magic**, we can write the KL divergence as

$$\mathrm{KL}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), P(\boldsymbol{z} \mid \boldsymbol{x})) = \underbrace{\log P(\boldsymbol{x} \mid \boldsymbol{z})}_{\text{Reconstruction error}} - \underbrace{\mathrm{KL}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), P(\boldsymbol{z}))}_{\text{Encoder and prior}}$$

We can pick our prior distribution $P(\boldsymbol{z})$, let's pick

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

We will reconstruct the input, and the latent space will be distributed according to $\mathcal{N}(\boldsymbol{0}, \boldsymbol{1})$

Variational Models

How do we implement f?

Variational Models

How do we implement f?

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

We represent a normal distribution with 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

Variational Models

How do we implement f?

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

We represent a normal distribution with 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

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

Variational Models

```
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))
```

Variational Models

We covered the encoder

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

We can use the same decoder as a standard autoencoder

Variational Models

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?

Variational Models

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

Variational Models

We can sample from the distribution

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

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

Variational Models

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

Variational Models

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 that matter?

Answer: Must be differentiable for gradient descent

Variational Models

VAE paper proposes the **reparameterization trick**

Variational Models

VAE paper proposes the **reparameterization trick**

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

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

Variational Models

VAE paper proposes the **reparameterization trick**

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

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

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

Variational Models

VAE paper proposes the **reparameterization trick**

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

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

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

We can sample and use gradient descent

Variational Models

VAE paper proposes the **reparameterization trick**

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

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

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

We can sample and use gradient descent

This trick only works with certain distributions

Variational Models

Put it all together

Variational Models

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

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

Variational Models

One last thing, the loss function

Variational Models

One last thing, the loss function

We have this error function

Variational Models

One last thing, the loss function

We have this error function

$$\mathrm{KL}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), P(\boldsymbol{z} \mid \boldsymbol{x})) = \underbrace{\log P(\boldsymbol{x} \mid \boldsymbol{z})}_{\text{Reconstruction error}} - \underbrace{\mathrm{KL}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), P(\boldsymbol{z}))}_{\text{Encoder and prior}}$$

Variational Models

One last thing, the loss function

We have this error function

$$\mathrm{KL}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), P(\boldsymbol{z} \mid \boldsymbol{x})) = \underbrace{\log P(\boldsymbol{x} \mid \boldsymbol{z})}_{\text{Reconstruction error}} - \underbrace{\mathrm{KL}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), P(\boldsymbol{z}))}_{\text{Encoder and prior}}$$

Turn it into a loss function

Variational Models

$$\mathrm{KL}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), P(\boldsymbol{z} \mid \boldsymbol{x})) = \underbrace{\log P(\boldsymbol{x} \mid \boldsymbol{z})}_{\text{Reconstruction error}} - \underbrace{\mathrm{KL}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), P(\boldsymbol{z}))}_{\text{Encoder and prior}}$$

Variational Models

$$\mathrm{KL}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), P(\boldsymbol{z} \mid \boldsymbol{x})) = \underbrace{\log P(\boldsymbol{x} \mid \boldsymbol{z})}_{\text{Reconstruction error}} - \underbrace{\mathrm{KL}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), P(\boldsymbol{z}))}_{\text{Encoder and prior}}$$

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

Variational Models

$$\text{KL}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), P(\boldsymbol{z} \mid \boldsymbol{x})) = \underbrace{\log P(\boldsymbol{x} \mid \boldsymbol{z})}_{\text{Reconstruction error}} - \underbrace{\text{KL}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), P(\boldsymbol{z}))}_{\text{Encoder and prior}}$$

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

$$\text{KL}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), P(\boldsymbol{z} \mid \boldsymbol{x})) = \left(\sum_{j=1}^{d_z} \mu_j^2 + \sigma_j^2 - \log(\sigma^2) - 1 \right)$$

Variational Models

$$\mathrm{KL}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), P(\boldsymbol{z} \mid \boldsymbol{x})) = \underbrace{\log P(\boldsymbol{x} \mid \boldsymbol{z})}_{\text{Reconstruction error}} - \underbrace{\mathrm{KL}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), P(\boldsymbol{z}))}_{\text{Encoder and prior}}$$

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

$$\mathrm{KL}(f(\boldsymbol{x}, \boldsymbol{\theta}_e), P(\boldsymbol{z} \mid \boldsymbol{x})) = \left(\sum_{j=1}^{d_z} \mu_j^2 + \sigma_j^2 - \log(\sigma^2) - 1 \right)$$

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

Variational Models

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

Plug in square error

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

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

Variational Models

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

Define over the entire dataset

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

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

Variational Models

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

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

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

Variational Models

Paper suggests using minibatch size m and dataset size n

Variational Models

Paper suggests using minibatch size m and dataset size n

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

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

Variational Models

Another paper finds hyperparameter β also helps

Variational Models

Another paper finds hyperparameter β also helps

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

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

Variational Models

```
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
```

Variational Models

Coding VAE

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

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