



# Transformers

CISC 7026 - Introduction to Deep Learning

Steven Morad

University of Macau

Review .....	2
Going Deeper .....	11
Transformers .....	29
Positional Encoding .....	33
Applications .....	54
Text Transformers .....	56
Image Transformers .....	61
Unsupervised Training .....	64
World Models .....	80
Closing Remarks .....	83
Course Evaluation .....	90

# Review

---

# Review

Last time, we derived various forms of **attention**

We started with composite memory

# Review

Last time, we derived various forms of **attention**

We started with composite memory

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

# Review

Last time, we derived various forms of **attention**

We started with composite memory

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

Given large enough  $T$ , we will eventually run out of storage space

# Review

Last time, we derived various forms of **attention**

We started with composite memory

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

Given large enough  $T$ , we will eventually run out of storage space

The sum is a **lossy** operation that can store a limited amount of information

# Review

Last time, we derived various forms of **attention**

We started with composite memory

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

Given large enough  $T$ , we will eventually run out of storage space

The sum is a **lossy** operation that can store a limited amount of information

# Review

So we introduced a forgetting term  $\gamma$

# Review

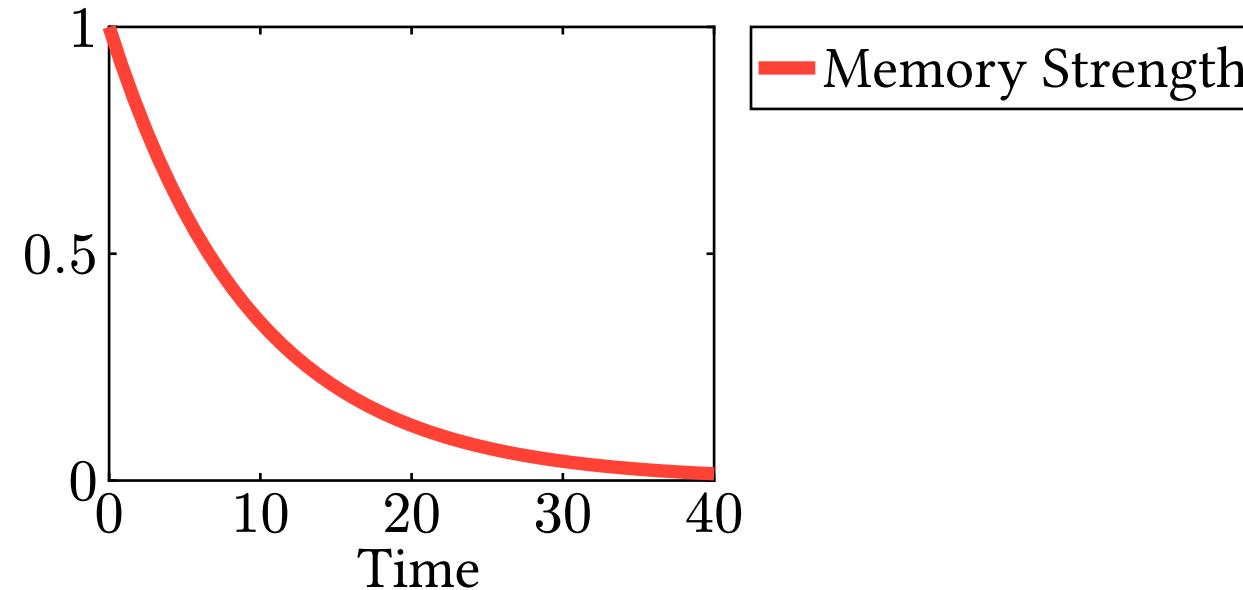
So we introduced a forgetting term  $\gamma$

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

# Review

So we introduced a forgetting term  $\gamma$

$$f(\mathbf{x}, \boldsymbol{\theta}) = \sum_{i=1}^T \gamma^{T-i} \cdot \boldsymbol{\theta}^\top \bar{\mathbf{x}}_i$$



# Review

We went to a party and the forgetting seemed ok

# Review

We went to a party and the forgetting seemed ok



# Review

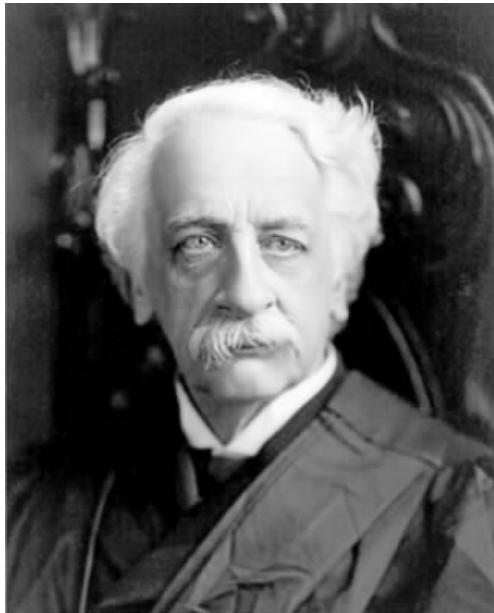
We went to a party and the forgetting seemed ok



10 PM

# Review

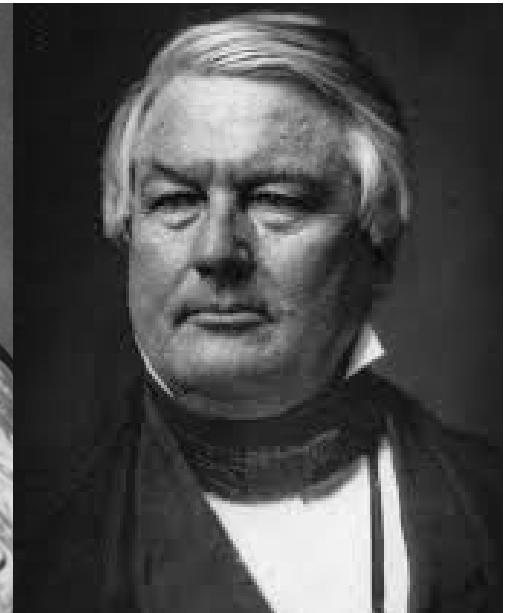
We went to a party and the forgetting seemed ok



10 PM



11 PM



# Review

We went to a party and the forgetting seemed ok



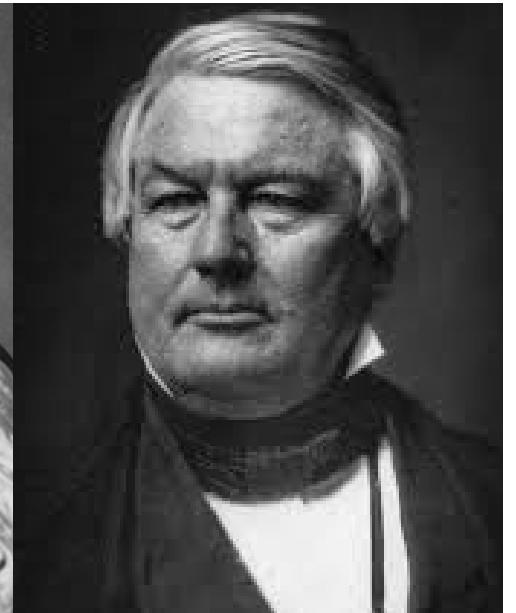
10 PM



11 PM

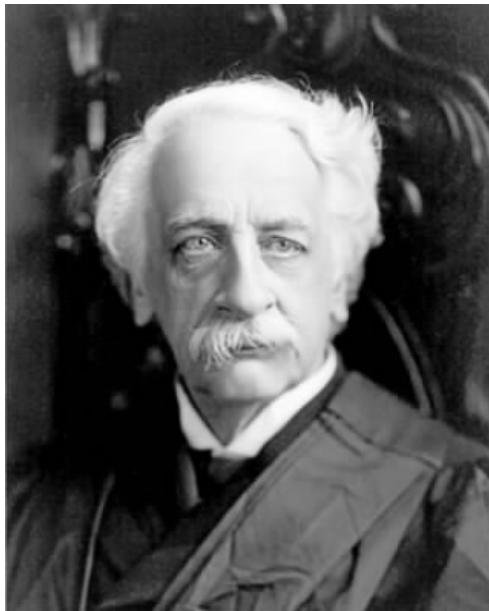


12 AM



# Review

We went to a party and the forgetting seemed ok



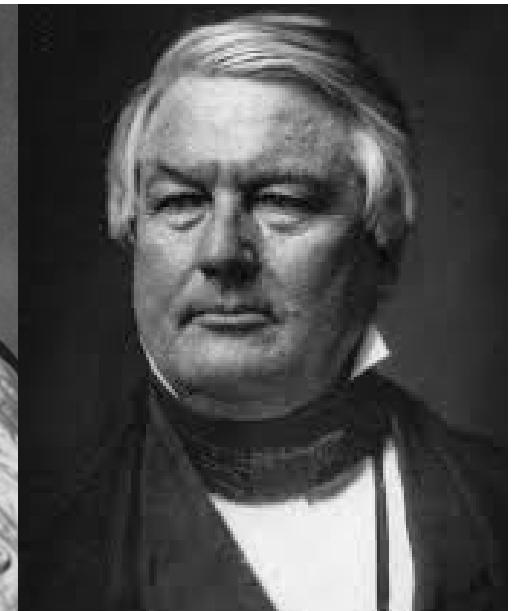
10 PM



11 PM



12 AM



1 AM

# Review



# Review



$$\gamma^3 \theta^\top \bar{x}_1$$

# Review



$$\gamma^3 \boldsymbol{\theta}^\top \bar{\boldsymbol{x}}_1$$

$$\gamma^2 \boldsymbol{\theta}^\top \bar{\boldsymbol{x}}_2$$

# Review



$$\gamma^3 \theta^\top \bar{x}_1$$

$$\gamma^2 \theta^\top \bar{x}_2$$

$$\gamma^1 \theta^\top \bar{x}_3$$

# Review



$$\gamma^3 \boldsymbol{\theta}^\top \bar{\boldsymbol{x}}_1$$

$$\gamma^2 \boldsymbol{\theta}^\top \bar{\boldsymbol{x}}_2$$

$$\gamma^1 \boldsymbol{\theta}^\top \bar{\boldsymbol{x}}_3$$

$$\gamma^0 \boldsymbol{\theta}^\top \bar{\boldsymbol{x}}_4$$

# Review



$$\gamma^3 \boldsymbol{\theta}^\top \bar{\boldsymbol{x}}_1$$

$$\gamma^2 \boldsymbol{\theta}^\top \bar{\boldsymbol{x}}_2$$

$$\gamma^1 \boldsymbol{\theta}^\top \bar{\boldsymbol{x}}_3$$

$$\gamma^0 \boldsymbol{\theta}^\top \bar{\boldsymbol{x}}_4$$

# Review

But we encountered problems when Taylor Swift arrived at the party

# Review

But we encountered problems when Taylor Swift arrived at the party



# Review

But we encountered problems when Taylor Swift arrived at the party



$$\gamma^4 \theta^\top \bar{x}_1$$

$$\gamma^3 \theta^\top \bar{x}_2$$

$$\gamma^2 \theta^\top \bar{x}_3$$

$$\gamma^1 \theta^\top \bar{x}_4$$

$$\gamma^0 \theta^\top \bar{x}_5$$

# Review

But we encountered problems when Taylor Swift arrived at the party



$$\gamma^4 \theta^\top \bar{x}_1$$

$$\gamma^3 \theta^\top \bar{x}_2$$

$$\gamma^2 \theta^\top \bar{x}_3$$

$$\gamma^1 \theta^\top \bar{x}_4$$

$$\gamma^0 \theta^\top \bar{x}_5$$

# Review



$$\gamma^4 \theta^\top \bar{x}_1$$

$$\gamma^3 \theta^\top \bar{x}_2$$

$$\gamma^2 \theta^\top \bar{x}_3$$

$$\gamma^1 \theta^\top \bar{x}_4$$

$$\gamma^0 \theta^\top \bar{x}_5$$

# Review



$$\gamma^4 \boldsymbol{\theta}^\top \bar{\boldsymbol{x}}_1$$

$$\gamma^3 \boldsymbol{\theta}^\top \bar{\boldsymbol{x}}_2$$

$$\gamma^2 \boldsymbol{\theta}^\top \bar{\boldsymbol{x}}_3$$

$$\gamma^1 \boldsymbol{\theta}^\top \bar{\boldsymbol{x}}_4$$

$$\gamma^0 \boldsymbol{\theta}^\top \bar{\boldsymbol{x}}_5$$

With our current model, we forget Taylor Swift!

# Review



$$\gamma^4 \boldsymbol{\theta}^\top \bar{\boldsymbol{x}}_1$$

$$\gamma^3 \boldsymbol{\theta}^\top \bar{\boldsymbol{x}}_2$$

$$\gamma^2 \boldsymbol{\theta}^\top \bar{\boldsymbol{x}}_3$$

$$\gamma^1 \boldsymbol{\theta}^\top \bar{\boldsymbol{x}}_4$$

$$\gamma^0 \boldsymbol{\theta}^\top \bar{\boldsymbol{x}}_5$$

With our current model, we forget Taylor Swift!

Our model of human memory is incomplete

# Review

# Review

Last time we studied attention

# Review

Last time we studied attention

Overview of transformer application and domains

# Going Deeper

---

# Going Deeper

We previously reviewed training tricks

# Going Deeper

We previously reviewed training tricks

- Deeper networks

# Going Deeper

We previously reviewed training tricks

- Deeper networks
- Parameter initialization

# Going Deeper

We previously reviewed training tricks

- Deeper networks
- Parameter initialization
- Stochastic gradient descent

# Going Deeper

We previously reviewed training tricks

- Deeper networks
- Parameter initialization
- Stochastic gradient descent
- Adaptive optimization

# Going Deeper

We previously reviewed training tricks

- Deeper networks
- Parameter initialization
- Stochastic gradient descent
- Adaptive optimization
- Weight decay

# Going Deeper

We previously reviewed training tricks

- Deeper networks
- Parameter initialization
- Stochastic gradient descent
- Adaptive optimization
- Weight decay

These methods empirically improve performance, but we do not always understand why

# Going Deeper

Modern transformers can be very deep

# Going Deeper

Modern transformers can be very deep

For this reason, they use two new training tricks to enable very deep models

# Going Deeper

Modern transformers can be very deep

For this reason, they use two new training tricks to enable very deep models

- Residual connections

# Going Deeper

Modern transformers can be very deep

For this reason, they use two new training tricks to enable very deep models

- Residual connections
- Layer normalization

# Going Deeper

Modern transformers can be very deep

For this reason, they use two new training tricks to enable very deep models

- Residual connections
- Layer normalization

Let us introduce these tricks

# Going Deeper

Modern transformers can be very deep

For this reason, they use two new training tricks to enable very deep models

- Residual connections
- Layer normalization

Let us introduce these tricks

We will start with the **residual connection**

# Going Deeper

Remember that a two-layer MLP is a universal function approximator

# Going Deeper

Remember that a two-layer MLP is a universal function approximator

$$| f(\mathbf{x}, \boldsymbol{\theta}) - g(\mathbf{x}) | < \varepsilon$$

# Going Deeper

Remember that a two-layer MLP is a universal function approximator

$$| f(\mathbf{x}, \boldsymbol{\theta}) - g(\mathbf{x}) | < \varepsilon$$

This is only as the width of the network goes to infinity

# Going Deeper

Remember that a two-layer MLP is a universal function approximator

$$| f(\mathbf{x}, \boldsymbol{\theta}) - g(\mathbf{x}) | < \varepsilon$$

This is only as the width of the network goes to infinity

For certain problems, we need deeper networks

# Going Deeper

Remember that a two-layer MLP is a universal function approximator

$$| f(\mathbf{x}, \boldsymbol{\theta}) - g(\mathbf{x}) | < \varepsilon$$

This is only as the width of the network goes to infinity

For certain problems, we need deeper networks

# Going Deeper

But there is a limit!

# Going Deeper

But there is a limit!

Making the network too deep can hurt performance

# Going Deeper

But there is a limit!

Making the network too deep can hurt performance

The theory is that the input information is **lost** somewhere in the network

# Going Deeper

But there is a limit!

Making the network too deep can hurt performance

The theory is that the input information is **lost** somewhere in the network

$$\mathbf{y} = f_k(\dots f_2(f_1(\mathbf{x}, \boldsymbol{\theta}_1), \boldsymbol{\theta}_2), \dots, \boldsymbol{\theta}_k)$$

# Going Deeper

$$\mathbf{y} = f_k(\dots f_2(f_1(\mathbf{x}, \boldsymbol{\theta}_1), \boldsymbol{\theta}_2), \dots, \boldsymbol{\theta}_k)$$

**Claim:** If the input information is present throughout the network, then we should be able to learn the identity function  $f(x) = x$

# Going Deeper

$$\mathbf{y} = f_k(\dots f_2(f_1(\mathbf{x}, \boldsymbol{\theta}_1), \boldsymbol{\theta}_2), \dots, \boldsymbol{\theta}_k)$$

**Claim:** If the input information is present throughout the network, then we should be able to learn the identity function  $f(x) = x$

$$\mathbf{x} = f_k(\dots f_2(f_1(\mathbf{x}, \boldsymbol{\theta}_1), \boldsymbol{\theta}_2), \dots, \boldsymbol{\theta}_k)$$

**Question:** We have seen a similar model, what was it?

# Going Deeper

$$y = f_k(\dots f_2(f_1(x, \theta_1), \theta_2), \dots, \theta_k)$$

**Claim:** If the input information is present throughout the network, then we should be able to learn the identity function  $f(x) = x$

$$x = f_k(\dots f_2(f_1(x, \theta_1), \theta_2), \dots, \theta_k)$$

**Question:** We have seen a similar model, what was it?

**Question:** Do you agree or disagree with the claim?

# Going Deeper

$$y = f_k(\dots f_2(f_1(x, \theta_1), \theta_2), \dots, \theta_k)$$

**Claim:** If the input information is present throughout the network, then we should be able to learn the identity function  $f(x) = x$

$$x = f_k(\dots f_2(f_1(x, \theta_1), \theta_2), \dots, \theta_k)$$

**Question:** We have seen a similar model, what was it?

**Question:** Do you agree or disagree with the claim?

<https://colab.research.google.com/drive/1qVlbQKpTuBYIa7FvC4IH-kJq-E0jmc0d#scrollTo=bg74S-AvbmJz>

# Going Deeper

$$x = f_k(\dots f_2(f_1(x, \theta_1), \theta_2), \dots, \theta_k)$$

# Going Deeper

$$x = f_k(\dots f_2(f_1(x, \theta_1), \theta_2), \dots, \theta_k)$$

Very deep networks struggle to learn the identity function

# Going Deeper

$$x = f_k(\dots f_2(f_1(x, \theta_1), \theta_2), \dots, \theta_k)$$

Very deep networks struggle to learn the identity function

If the input information is available, then learning the identity function should be very easy!

**Question:** How can we prevent the input from getting lost?

# Going Deeper

We can feed the input to each layer

# Going Deeper

We can feed the input to each layer

The first approach is called the **DenseNet** approach

# Going Deeper

We can feed the input to each layer

The first approach is called the **DenseNet** approach

$$z_1 = f_1(x, \theta_1)$$

$$z_2 = f_2\left(\begin{bmatrix} x \\ z_1 \end{bmatrix}, \theta_2\right)$$

⋮

$$z_k = f_k\left(\begin{bmatrix} x \\ z_1 \\ \vdots \\ z_{k-1} \end{bmatrix}, \theta_k\right)$$

# Going Deeper

$$z_1 = f_1(x, \theta_1)$$

$$z_2 = f_2\left(\begin{bmatrix} x \\ z_1 \end{bmatrix}, \theta_2\right)$$

⋮

$$z_k = f_k\left(\begin{bmatrix} x \\ z_1 \\ \vdots \\ z_{k-1} \end{bmatrix}, \theta_k\right)$$

# Going Deeper

$$z_1 = f_1(x, \theta_1)$$

$$z_2 = f_2\left(\begin{bmatrix} x \\ z_1 \end{bmatrix}, \theta_2\right)$$

⋮

$$z_k = f_k\left(\begin{bmatrix} x \\ z_1 \\ \vdots \\ z_{k-1} \end{bmatrix}, \theta_k\right)$$

**Question:** Any issues with the DenseNet approach?

# Going Deeper

$$z_1 = f_1(x, \theta_1)$$

$$z_2 = f_2\left(\begin{bmatrix} x \\ z_1 \end{bmatrix}, \theta_2\right)$$

⋮

$$z_k = f_k\left(\begin{bmatrix} x \\ z_1 \\ \vdots \\ z_{k-1} \end{bmatrix}, \theta_k\right)$$

**Question:** Any issues with the DenseNet approach?

**Answer:** Very deep networks require too many parameters!

# Going Deeper

The next method is called the **ResNet** approach

$$z_1 = f_1(x, \theta_1)$$

$$z_2 = f_2(x, \theta_2) + z_1$$

$$z_3 = f_2(x, \theta_3) + z_2$$

⋮

$$z_k = f_k(x, \theta_k) + z_{k-1}$$

# Going Deeper

The next method is called the **ResNet** approach

$$z_1 = f_1(x, \theta_1)$$

$$z_2 = f_2(x, \theta_2) + z_1$$

$$z_3 = f_2(x, \theta_3) + z_2$$

⋮

$$z_k = f_k(x, \theta_k) + z_{k-1}$$

This allows information to flow around the layers

# Going Deeper

The next method is called the **ResNet** approach

$$z_1 = f_1(x, \theta_1)$$

$$z_2 = f_2(x, \theta_2) + z_1$$

$$z_3 = f_2(x, \theta_3) + z_2$$

⋮

$$z_k = f_k(x, \theta_k) + z_{k-1}$$

This allows information to flow around the layers

It requires much fewer parameters than a dense net, but also does not work as well

# Going Deeper

The next method is called the **ResNet** approach

$$z_1 = f_1(x, \theta_1)$$

$$z_2 = f_2(x, \theta_2) + z_1$$

$$z_3 = f_2(x, \theta_3) + z_2$$

⋮

$$z_k = f_k(x, \theta_k) + z_{k-1}$$

This allows information to flow around the layers

It requires much fewer parameters than a dense net, but also does not work as well

# Going Deeper

We call  $f(x) + x$  a **residual connection**

# Going Deeper

We call  $f(x) + x$  a **residual connection**

Instead of learning how to change  $x$ ,  $f$  learns what to add to  $x$

# Going Deeper

We call  $f(\mathbf{x}) + \mathbf{x}$  a **residual connection**

Instead of learning how to change  $\mathbf{x}$ ,  $f$  learns what to add to  $\mathbf{x}$

For example, for an identity function we can easily learn

$$f(\mathbf{x}, \theta) = 0; \quad f(\mathbf{x}, \theta) + \mathbf{x} = \mathbf{x}$$

# Going Deeper

We call  $f(\mathbf{x}) + \mathbf{x}$  a **residual connection**

Instead of learning how to change  $\mathbf{x}$ ,  $f$  learns what to add to  $\mathbf{x}$

For example, for an identity function we can easily learn

$$f(\mathbf{x}, \theta) = 0; \quad f(\mathbf{x}, \theta) + \mathbf{x} = \mathbf{x}$$

This helps prevent information from getting lost in very deep networks

# Going Deeper

The second trick is called **layer normalization**

# Going Deeper

The second trick is called **layer normalization**

Recall that with parameter initialization and weight decay, we ensure the parameters are fairly small

# Going Deeper

The second trick is called **layer normalization**

Recall that with parameter initialization and weight decay, we ensure the parameters are fairly small

But we can still have very small or very large outputs from each layer

# Going Deeper

The second trick is called **layer normalization**

Recall that with parameter initialization and weight decay, we ensure the parameters are fairly small

But we can still have very small or very large outputs from each layer

$$f_1(\mathbf{x}, \boldsymbol{\theta}_1) = \sum_{i=1}^{d_x} \theta_{1,i} x_i$$

# Going Deeper

The second trick is called **layer normalization**

Recall that with parameter initialization and weight decay, we ensure the parameters are fairly small

But we can still have very small or very large outputs from each layer

$$f_1(\mathbf{x}, \boldsymbol{\theta}_1) = \sum_{i=1}^{d_x} \theta_{1,i} x_i$$

**Question:** If all  $x_i = 1$ ,  $\theta_{1,i} = 0.01$  and  $d_x = 1000$ , what is the output?

# Going Deeper

$$f_1(\boldsymbol{x}, \boldsymbol{\theta}_1) = \sum_{i=1}^{d_x} \theta_{1,i} x_i$$

# Going Deeper

$$f_1(\mathbf{x}, \boldsymbol{\theta}_1) = \sum_{i=1}^{d_x} \theta_{1,i} x_i$$

$$f_1(\mathbf{x}, \boldsymbol{\theta}_1) = \sum_{i=1}^{1000} 0.01 \cdot 1 = 10$$

# Going Deeper

$$f_1(\mathbf{x}, \boldsymbol{\theta}_1) = \sum_{i=1}^{d_x} \theta_{1,i} x_i$$

$$f_1(\mathbf{x}, \boldsymbol{\theta}_1) = \sum_{i=1}^{1000} 0.01 \cdot 1 = 10$$

What if we add another layer with the same  $d_x$  and  $\theta$ ?

$$f_2(\mathbf{z}, \boldsymbol{\theta}_2) = \sum_{i=1}^{1000} 0.01 \cdot 10 = 100$$

# Going Deeper

$$f_1(\mathbf{x}, \boldsymbol{\theta}_1) = \sum_{i=1}^{d_x} \theta_{1,i} x_i$$

$$f_1(\mathbf{x}, \boldsymbol{\theta}_1) = \sum_{i=1}^{1000} 0.01 \cdot 1 = 10$$

What if we add another layer with the same  $d_x$  and  $\theta$ ?

$$f_2(\mathbf{z}, \boldsymbol{\theta}_2) = \sum_{i=1}^{1000} 0.01 \cdot 10 = 100$$

**Question:** What is the problem?

# Going Deeper

Let us look at the gradient

# Going Deeper

Let us look at the gradient

$$\nabla_{\theta_1} f_2(f_1(x, \theta_1), \theta_2) =$$

# Going Deeper

Let us look at the gradient

$$\nabla_{\theta_1} f_2(f_1(x, \theta_1), \theta_2) = \nabla_{\theta_1}[f_2](f_1(x, \theta_1)) \cdot \nabla_{\theta_1}[f_1](x, \theta_1)$$

# Going Deeper

Let us look at the gradient

$$\begin{aligned}\nabla_{\theta_1} f_2(f_1(\mathbf{x}, \theta_1), \theta_2) &= \nabla_{\theta_1}[f_2](f_1(\mathbf{x}, \theta_1)) \cdot \nabla_{\theta_1}[f_1](\mathbf{x}, \theta_1) \\ &\approx 100 \cdot 10\end{aligned}$$

Can cause exploding or vanishing gradient

# Going Deeper

Let us look at the gradient

$$\begin{aligned}\nabla_{\theta_1} f_2(f_1(x, \theta_1), \theta_2) &= \nabla_{\theta_1}[f_2](f_1(x, \theta_1)) \cdot \nabla_{\theta_1}[f_1](x, \theta_1) \\ &\approx 100 \cdot 10\end{aligned}$$

Can cause exploding or vanishing gradient

Deeper network  $\Rightarrow$  worse exploding/vanishing issues

**Question:** What can we do?

# Going Deeper

Let us look at the gradient

$$\begin{aligned}\nabla_{\theta_1} f_2(f_1(x, \theta_1), \theta_2) &= \nabla_{\theta_1}[f_2](f_1(x, \theta_1)) \cdot \nabla_{\theta_1}[f_1](x, \theta_1) \\ &\approx 100 \cdot 10\end{aligned}$$

Can cause exploding or vanishing gradient

Deeper network  $\Rightarrow$  worse exploding/vanishing issues

**Question:** What can we do?

# Going Deeper

We can use **layer normalization**

# Going Deeper

We can use **layer normalization**

First, layer normalization **centers** the output of the layer

# Going Deeper

We can use **layer normalization**

First, layer normalization **centers** the output of the layer

$$\mu = d_y \sum_{i=1}^{d_y} f(x, \theta)_i$$

$$f(x, \theta) - \mu$$

**Question:** What does this do?

# Going Deeper

We can use **layer normalization**

First, layer normalization **centers** the output of the layer

$$\mu = d_y \sum_{i=1}^{d_y} f(x, \theta)_i$$

$$f(x, \theta) - \mu$$

**Question:** What does this do?

**Answer:** Makes output have zero mean (both positive and negative values)

# Going Deeper

We can use **layer normalization**

First, layer normalization **centers** the output of the layer

$$\mu = d_y \sum_{i=1}^{d_y} f(x, \theta)_i$$

$$f(x, \theta) - \mu$$

**Question:** What does this do?

**Answer:** Makes output have zero mean (both positive and negative values)

# Going Deeper

$$\mu = d_y \sum_{i=1}^{d_y} f(\mathbf{x}, \boldsymbol{\theta})_i \quad f(\mathbf{x}, \boldsymbol{\theta}) - \mu$$

Then, layer normalization **rescales** the outputs

$$\sigma = \frac{\sqrt{\sum_{i=1}^{d_y} f(\mathbf{x}, \boldsymbol{\theta}_i - \mu)^2}}{d_y}$$

$$\text{LN}(f(\mathbf{x}, \boldsymbol{\theta})) = \frac{f(\mathbf{x}, \boldsymbol{\theta}) - \mu}{\sigma}$$

Now, the output of the layer is normally distributed

# Going Deeper

If the output is normally distributed:

# Going Deeper

If the output is normally distributed:

- 99.7% of outputs  $\in [-3, 3]$

# Going Deeper

If the output is normally distributed:

- 99.7% of outputs  $\in [-3, 3]$
- 99.99% of outputs  $\in [-4, 4]$

# Going Deeper

If the output is normally distributed:

- 99.7% of outputs  $\in [-3, 3]$
- 99.99% of outputs  $\in [-4, 4]$
- 99.9999% of outputs  $\in [-5, 5]$

# Going Deeper

If the output is normally distributed:

- 99.7% of outputs  $\in [-3, 3]$
- 99.99% of outputs  $\in [-4, 4]$
- 99.9999% of outputs  $\in [-5, 5]$

This helps prevent vanishing and exploding gradients

# Going Deeper

Now, let's combine residual connections and layer norm and try our very deep network again

TODO COLAB

# Transformers

---

# Transformers

Now we have everything we need to implement a transformer

# Transformers

Now we have everything we need to implement a transformer

- Attention

# Transformers

Now we have everything we need to implement a transformer

- Attention
- MLP

# Transformers

Now we have everything we need to implement a transformer

- Attention
- MLP
- Residual connections

# Transformers

Now we have everything we need to implement a transformer

- Attention
- MLP
- Residual connections
- Layer normalization

A deep neural network consists of many layers

# Transformers

Now we have everything we need to implement a transformer

- Attention
- MLP
- Residual connections
- Layer normalization

A deep neural network consists of many layers

A transformer consists of many **transformer blocks**

# Transformers

```
class TransformerBlock(nn.Module):
    def __init__(self):
        self.attn = Attention()
        self.mlp = Sequential(
            Linear(d_h, d_h), LeakyReLU(), Linear(d_h, d_h))
        self.norm1 = nn.LayerNorm(d_h)
        self.norm2 = nn.LayerNorm(d_h)

    def forward(self, x):
        x = self.norm1(self.attn(x) + x)
        x = self.norm2(self.mlp(x) + x)
        return x
```

# Transformers

```
class Transformer(nn.Module):  
    def __init__(self):  
        self.block1 = TransformerBlock()  
        self.block2 = TransformerBlock()  
        self.block3 = TransformerBlock()  
  
    def forward(self, x):  
        x = self.block1(x)  
        x = self.block2(x)  
        x = self.block3(x)  
        return x
```

# Transformers

```
class Transformer(nn.Module):
    def __init__(self):
        self.block1 = TransformerBlock()
        self.block2 = TransformerBlock()
        self.block3 = TransformerBlock()

    def forward(self, x):
        x = self.block1(x)
        x = self.block2(x)
        x = self.block3(x)
        return x
```

**Question:** What are the input/output shapes of the transformer?

# Transformers

```
class Transformer(nn.Module):
    def __init__(self):
        self.block1 = TransformerBlock()
        self.block2 = TransformerBlock()
        self.block3 = TransformerBlock()

    def forward(self, x):
        x = self.block1(x)
        x = self.block2(x)
        x = self.block3(x)
        return x
```

**Question:** What are the input/output shapes of the transformer?

**Answer:**  $f : \mathbb{R}^{T \times d_x} \mapsto \mathbb{R}^{T \times d_h}$

# Positional Encoding

---

# Positional Encoding

Our transformer maps  $T$  inputs to  $T$  outputs

# Positional Encoding

Our transformer maps  $T$  inputs to  $T$  outputs

$$f : \mathbb{R}^{T \times d_x} \times \Theta \mapsto \mathbb{R}^{T \times d_y}$$

# Positional Encoding

Our transformer maps  $T$  inputs to  $T$  outputs

$$f : \mathbb{R}^{T \times d_x} \times \Theta \mapsto \mathbb{R}^{T \times d_y}$$

- $T$  words in a sentence

# Positional Encoding

Our transformer maps  $T$  inputs to  $T$  outputs

$$f : \mathbb{R}^{T \times d_x} \times \Theta \mapsto \mathbb{R}^{T \times d_y}$$

- $T$  words in a sentence
- $T$  pixels in an image

# Positional Encoding

Our transformer maps  $T$  inputs to  $T$  outputs

$$f : \mathbb{R}^{T \times d_x} \times \Theta \mapsto \mathbb{R}^{T \times d_y}$$

- $T$  words in a sentence
- $T$  pixels in an image
- $T$  amino acids in a protein

# Positional Encoding

Our transformer maps  $T$  inputs to  $T$  outputs

$$f : \mathbb{R}^{T \times d_x} \times \Theta \mapsto \mathbb{R}^{T \times d_y}$$

- $T$  words in a sentence
- $T$  pixels in an image
- $T$  amino acids in a protein

**Question:** Do we care about the order of the  $T$  inputs?

# Positional Encoding

Our transformer maps  $T$  inputs to  $T$  outputs

$$f : \mathbb{R}^{T \times d_x} \times \Theta \mapsto \mathbb{R}^{T \times d_y}$$

- $T$  words in a sentence
- $T$  pixels in an image
- $T$  amino acids in a protein

**Question:** Do we care about the order of the  $T$  inputs?

**Answer:** In some tasks, yes! In others, no

# Positional Encoding

**Question:** Detecting birds in an image of  $T$  pixels, does order matter?

# Positional Encoding

**Question:** Detecting birds in an image of  $T$  pixels, does order matter?



# Positional Encoding

**Question:** Detecting birds in an image of  $T$  pixels, does order matter?



**Answer:** Yes!

# Positional Encoding

**Question:**  $T$  robots searching for an object. Does order matter?

# Positional Encoding

**Question:**  $T$  robots searching for an object. Does order matter?



# Positional Encoding

**Question:**  $T$  robots searching for an object. Does order matter?



**Answer:** No!

# Positional Encoding

**Question:** We translate a sentence containing  $T$  words. Does order matter?

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} \text{The} \\ \text{dog} \\ \text{licks} \\ \text{the} \\ \text{owner} \end{bmatrix}$$

# Positional Encoding

**Question:** We translate a sentence containing  $T$  words. Does order matter?

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} \text{The} \\ \text{dog} \\ \text{licks} \\ \text{the} \\ \text{owner} \end{bmatrix}$$

$$\begin{bmatrix} x_1 \\ \textcolor{red}{x_5} \\ x_3 \\ x_4 \\ \textcolor{red}{x_2} \end{bmatrix} = \begin{bmatrix} \text{The} \\ \text{owner} \\ \text{licks} \\ \text{the} \\ \text{dog} \end{bmatrix}$$

# Positional Encoding

**Question:** We translate a sentence containing  $T$  words. Does order matter?

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} \text{The} \\ \text{dog} \\ \text{licks} \\ \text{the} \\ \text{owner} \end{bmatrix}$$

$$\begin{bmatrix} x_1 \\ \textcolor{red}{x_5} \\ x_3 \\ x_4 \\ \textcolor{red}{x_2} \end{bmatrix} = \begin{bmatrix} \text{The} \\ \text{owner} \\ \text{licks} \\ \text{the} \\ \text{dog} \end{bmatrix}$$

**Answer:** Yes!

# Positional Encoding

For some tasks, we must know the order of the inputs

# Positional Encoding

For some tasks, we must know the order of the inputs

**Question:** Does the transformer know the order of the  $T$  inputs?

# Positional Encoding

For some tasks, we must know the order of the inputs

**Question:** Does the transformer know the order of the  $T$  inputs?

Define a **permutation matrix**  $P \in \{0, 1\}^{T \times T}$  that reorders the inputs

# Positional Encoding

For some tasks, we must know the order of the inputs

**Question:** Does the transformer know the order of the  $T$  inputs?

Define a **permutation matrix**  $P \in \{0, 1\}^{T \times T}$  that reorders the inputs

# Positional Encoding

Example 1:

$$P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}; \quad a = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix};$$

# Positional Encoding

Example 1:

$$P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}; \quad a = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix}; \quad Pa = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix}$$

# Positional Encoding

**Example 1:**

$$P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}; \quad a = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix}; \quad Pa = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix}$$

**Example 2:**

$$P = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}; \quad a = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix};$$

# Positional Encoding

**Example 1:**

$$P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}; \quad a = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix}; \quad Pa = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix}$$

**Example 2:**

$$P = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}; \quad a = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix}; \quad Pa = \begin{bmatrix} 4 \\ 3 \\ 5 \end{bmatrix}$$

# Positional Encoding

$$f\left(P \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}\right) \neq Pf\left(\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}\right)$$

# Positional Encoding

$$f\left(P \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}\right) \neq Pf\left(\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}\right)$$

$f$  is equivariant, order **does** matter

# Positional Encoding

$$f\left(P \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}\right) \neq Pf\left(\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}\right)$$

$f$  is equivariant, order **does** matter

$$f\left(P \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}\right) = Pf\left(\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}\right)$$

# Positional Encoding

$$f\left(P \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}\right) \neq Pf\left(\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}\right)$$

$f$  is equivariant, order **does** matter

$$f\left(P \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}\right) = Pf\left(\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}\right)$$

$f$  is equivariant, order **does not** matter

Which is a transformer?

# Positional Encoding

$$f\left(P \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}\right) \neq Pf\left(\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}\right)$$

$f$  is equivariant, order **does** matter

$$f\left(P \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}\right) = Pf\left(\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}\right)$$

$f$  is equivariant, order **does not** matter

Which is a transformer?

MLP is equivariant, but what about attention?

# Positional Encoding

$$f\left(P \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}\right) \neq Pf\left(\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}\right)$$

$f$  is equivariant, order **does** matter

$$f\left(P \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}\right) = Pf\left(\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}\right)$$

$f$  is equivariant, order **does not** matter

Which is a transformer?

MLP is equivariant, but what about attention?

# Positional Encoding

Recall dot product self attention

# Positional Encoding

Recall dot product self attention

$$Q = [q_1 \ q_2 \ \dots \ q_T] = [\theta_Q^\top x_1 \ \theta_Q^\top x_2 \ \dots \ \theta_Q^\top x_T]$$

$$K = [k_1 \ k_2 \ \dots \ k_T] = [\theta_K^\top x_1 \ \theta_K^\top x_2 \ \dots \ \theta_K^\top x_T]$$

$$V = [v_1 \ v_2 \ \dots \ v_T] = [\theta_V^\top x_1 \ \theta_V^\top x_2 \ \dots \ \theta_V^\top x_T]$$

# Positional Encoding

Recall dot product self attention

$$Q = [q_1 \ q_2 \ \dots \ q_T] = [\theta_Q^\top x_1 \ \theta_Q^\top x_2 \ \dots \ \theta_Q^\top x_T]$$

$$K = [k_1 \ k_2 \ \dots \ k_T] = [\theta_K^\top x_1 \ \theta_K^\top x_2 \ \dots \ \theta_K^\top x_T]$$

$$V = [v_1 \ v_2 \ \dots \ v_T] = [\theta_V^\top x_1 \ \theta_V^\top x_2 \ \dots \ \theta_V^\top x_T]$$

$$\text{attn}\left(\begin{bmatrix} x_1 \\ \vdots \\ x_T \end{bmatrix}, \theta\right) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_h}}\right)V$$

# Positional Encoding

Permuting the inputs reorders  $Q, K, V$

# Positional Encoding

Permuting the inputs reorders  $Q, K, V$

$$PQ = [q_T \ q_1 \ \dots \ q_2] = [\theta_Q^\top x_T \ \theta_Q^\top x_1 \ \dots \ \theta_Q^\top x_2]$$

$$PK = [k_T \ k_1 \ \dots \ k_2] = [\theta_K^\top x_T \ \theta_K^\top x_1 \ \dots \ \theta_K^\top x_2]$$

$$PV = [v_T \ v_1 \ \dots \ v_2] = [\theta_V^\top x_T \ \theta_V^\top x_1 \ \dots \ \theta_V^\top x_2]$$

# Positional Encoding

Permuting the inputs reorders  $Q, K, V$

$$PQ = [q_T \ q_1 \ \dots \ q_2] = [\theta_Q^\top x_T \ \theta_Q^\top x_1 \ \dots \ \theta_Q^\top x_2]$$

$$PK = [k_T \ k_1 \ \dots \ k_2] = [\theta_K^\top x_T \ \theta_K^\top x_1 \ \dots \ \theta_K^\top x_2]$$

$$PV = [v_T \ v_1 \ \dots \ v_2] = [\theta_V^\top x_T \ \theta_V^\top x_1 \ \dots \ \theta_V^\top x_2]$$

$$\text{attn}\left(P \begin{bmatrix} x_1 \\ \vdots \\ x_T \end{bmatrix}, \theta\right) = \text{softmax}\left(\frac{(PQ)(PK)^\top}{\sqrt{d_h}}\right)(PV)$$

# Positional Encoding

Permuting the inputs reorders  $\mathbf{Q}, \mathbf{K}, \mathbf{V}$

$$\mathbf{PQ} = [q_T \ q_1 \ \dots \ q_2] = [\theta_Q^\top x_T \ \theta_Q^\top x_1 \ \dots \ \theta_Q^\top x_2]$$

$$\mathbf{PK} = [k_T \ k_1 \ \dots \ k_2] = [\theta_K^\top x_T \ \theta_K^\top x_1 \ \dots \ \theta_K^\top x_2]$$

$$\mathbf{PV} = [v_T \ v_1 \ \dots \ v_2] = [\theta_V^\top x_T \ \theta_V^\top x_1 \ \dots \ \theta_V^\top x_2]$$

$$\text{attn}\left(\mathbf{P} \begin{bmatrix} x_1 \\ \vdots \\ x_T \end{bmatrix}, \theta\right) = \text{softmax}\left(\frac{(\mathbf{PQ})(\mathbf{PK})^\top}{\sqrt{d_h}}\right)(\mathbf{PV})$$

# Positional Encoding

$$\text{attn}\left(\mathbf{P} \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_T \end{bmatrix}, \theta\right) = \text{softmax}\left(\frac{(\mathbf{PQ})(\mathbf{PK})^\top}{\sqrt{d_h}}\right)(\mathbf{PV})$$

# Positional Encoding

$$\text{attn}\left(\mathbf{P} \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_T \end{bmatrix}, \theta\right) = \text{softmax}\left(\frac{(\mathbf{PQ})(\mathbf{PK})^\top}{\sqrt{d_h}}\right)(\mathbf{PV})$$

$$\text{attn}\left(\mathbf{P} \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_T \end{bmatrix}, \theta\right) = \text{softmax}\left(\mathbf{P} \frac{\mathbf{Q}(\mathbf{PK})^\top}{\sqrt{d_h}}\right)(\mathbf{PV})$$

# Positional Encoding

$$\text{attn}\left(\mathbf{P} \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_T \end{bmatrix}, \theta\right) = \text{softmax}\left(\frac{(\mathbf{PQ})(\mathbf{PK})^\top}{\sqrt{d_h}}\right)(\mathbf{PV})$$

$$\text{attn}\left(\mathbf{P} \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_T \end{bmatrix}, \theta\right) = \text{softmax}\left(\mathbf{P} \frac{\mathbf{Q}(\mathbf{PK})^\top}{\sqrt{d_h}}\right)(\mathbf{PV})$$

$$\text{attn}\left(\mathbf{P} \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_T \end{bmatrix}, \theta\right) = \text{softmax}\left(\mathbf{P} \frac{\mathbf{Q}(\mathbf{K}^\top \mathbf{P}^\top)}{\sqrt{d_h}}\right)(\mathbf{PV})$$

# Positional Encoding

$$\text{attn}\left(\mathbf{P} \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_T \end{bmatrix}, \theta\right) = \text{softmax}\left(\frac{(\mathbf{PQ})(\mathbf{PK})^\top}{\sqrt{d_h}}\right)(\mathbf{PV})$$

$$\text{attn}\left(\mathbf{P} \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_T \end{bmatrix}, \theta\right) = \text{softmax}\left(\mathbf{P} \frac{\mathbf{Q}(\mathbf{PK})^\top}{\sqrt{d_h}}\right)(\mathbf{PV})$$

$$\text{attn}\left(\mathbf{P} \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_T \end{bmatrix}, \theta\right) = \text{softmax}\left(\mathbf{P} \frac{\mathbf{Q}(\mathbf{K}^\top \mathbf{P}^\top)}{\sqrt{d_h}}\right)(\mathbf{PV})$$

# Positional Encoding

$$\text{attn}\left(\mathbf{P} \begin{bmatrix} x_1 \\ \vdots \\ x_T \end{bmatrix}, \theta\right) = \text{softmax}\left(\mathbf{P} \frac{\mathbf{Q}(\mathbf{K}^\top \mathbf{P}^\top)}{\sqrt{d_h}}\right)(\mathbf{P}\mathbf{V})$$

# Positional Encoding

$$\text{attn}\left(\mathbf{P} \begin{bmatrix} x_1 \\ \vdots \\ x_T \end{bmatrix}, \theta\right) = \text{softmax}\left(\mathbf{P} \frac{\mathbf{Q}(\mathbf{K}^\top \mathbf{P}^\top)}{\sqrt{d_h}}\right)(\mathbf{P}\mathbf{V})$$

$$\text{attn}\left(\mathbf{P} \begin{bmatrix} x_1 \\ \vdots \\ x_T \end{bmatrix}, \theta\right) = \text{softmax}\left(\mathbf{P} \frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_h}} \mathbf{P}^\top\right)(\mathbf{P}\mathbf{V})$$

# Positional Encoding

$$\text{attn}\left(\mathbf{P} \begin{bmatrix} x_1 \\ \vdots \\ x_T \end{bmatrix}, \theta\right) = \text{softmax}\left(\mathbf{P} \frac{\mathbf{Q}(\mathbf{K}^\top \mathbf{P}^\top)}{\sqrt{d_h}}\right)(\mathbf{P}\mathbf{V})$$

$$\text{attn}\left(\mathbf{P} \begin{bmatrix} x_1 \\ \vdots \\ x_T \end{bmatrix}, \theta\right) = \text{softmax}\left(\mathbf{P} \frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_h}} \mathbf{P}^\top\right)(\mathbf{P}\mathbf{V})$$

$$\text{attn}\left(\mathbf{P} \begin{bmatrix} x_1 \\ \vdots \\ x_T \end{bmatrix}, \theta\right) = \mathbf{P} \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_h}} \mathbf{P}^\top\right)(\mathbf{P}\mathbf{V})$$

# Positional Encoding

$$\text{attn}\left(\mathbf{P} \begin{bmatrix} x_1 \\ \vdots \\ x_T \end{bmatrix}, \theta\right) = \text{softmax}\left(\mathbf{P} \frac{\mathbf{Q}(\mathbf{K}^\top \mathbf{P}^\top)}{\sqrt{d_h}}\right)(\mathbf{P}\mathbf{V})$$

$$\text{attn}\left(\mathbf{P} \begin{bmatrix} x_1 \\ \vdots \\ x_T \end{bmatrix}, \theta\right) = \text{softmax}\left(\mathbf{P} \frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_h}} \mathbf{P}^\top\right)(\mathbf{P}\mathbf{V})$$

$$\text{attn}\left(\mathbf{P} \begin{bmatrix} x_1 \\ \vdots \\ x_T \end{bmatrix}, \theta\right) = \mathbf{P} \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_h}} \mathbf{P}^\top\right)(\mathbf{P}\mathbf{V})$$

# Positional Encoding

$$\text{attn}\left(\mathbf{P} \begin{bmatrix} x_1 \\ \vdots \\ x_T \end{bmatrix}, \theta\right) = \mathbf{P} \text{softmax}\left(\frac{\mathbf{QK}^\top}{\sqrt{d_h}} \mathbf{P}^\top\right) (\mathbf{PV})$$

# Positional Encoding

$$\text{attn}\left(\mathbf{P} \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_T \end{bmatrix}, \theta\right) = \mathbf{P} \text{softmax}\left(\frac{\mathbf{QK}^\top}{\sqrt{d_h}} \mathbf{P}^\top\right) (\mathbf{PV})$$

$$\text{attn}\left(\mathbf{P} \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_T \end{bmatrix}, \theta\right) = \mathbf{P} \text{softmax}\left(\frac{\mathbf{QK}^\top}{\sqrt{d_h}}\right) \mathbf{P}^\top (\mathbf{PV})$$

# Positional Encoding

$$\text{attn}\left(\mathbf{P} \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_T \end{bmatrix}, \theta\right) = \mathbf{P} \text{softmax}\left(\frac{\mathbf{QK}^\top}{\sqrt{d_h}} \mathbf{P}^\top\right) (\mathbf{PV})$$

$$\text{attn}\left(\mathbf{P} \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_T \end{bmatrix}, \theta\right) = \mathbf{P} \text{softmax}\left(\frac{\mathbf{QK}^\top}{\sqrt{d_h}}\right) \mathbf{P}^\top (\mathbf{PV})$$

$\mathbf{P}$  swaps row  $i$  and  $j$

# Positional Encoding

$$\text{attn}\left(\mathbf{P} \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_T \end{bmatrix}, \theta\right) = \mathbf{P} \text{softmax}\left(\frac{\mathbf{QK}^\top}{\sqrt{d_h}} \mathbf{P}^\top\right) (\mathbf{PV})$$

$$\text{attn}\left(\mathbf{P} \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_T \end{bmatrix}, \theta\right) = \mathbf{P} \text{softmax}\left(\frac{\mathbf{QK}^\top}{\sqrt{d_h}}\right) \mathbf{P}^\top (\mathbf{PV})$$

$\mathbf{P}$  swaps row  $i$  and  $j$      $\mathbf{P}^T$  swaps row  $j$  and  $i$

# Positional Encoding

$$\text{attn}\left(\mathbf{P} \begin{bmatrix} x_1 \\ \vdots \\ x_T \end{bmatrix}, \theta\right) = \mathbf{P} \text{softmax}\left(\frac{\mathbf{QK}^\top}{\sqrt{d_h}} \mathbf{P}^\top\right) (\mathbf{PV})$$

$$\text{attn}\left(\mathbf{P} \begin{bmatrix} x_1 \\ \vdots \\ x_T \end{bmatrix}, \theta\right) = \mathbf{P} \text{softmax}\left(\frac{\mathbf{QK}^\top}{\sqrt{d_h}}\right) \mathbf{P}^\top (\mathbf{PV})$$

$\mathbf{P}$  swaps row  $i$  and  $j$      $\mathbf{P}^T$  swaps row  $j$  and  $i$                    $\mathbf{P}^\top \mathbf{P} = \mathbf{I}$

# Positional Encoding

$$\text{attn}\left(\mathbf{P} \begin{bmatrix} x_1 \\ \vdots \\ x_T \end{bmatrix}, \theta\right) = \mathbf{P} \text{softmax}\left(\frac{\mathbf{QK}^\top}{\sqrt{d_h}} \mathbf{P}^\top\right) (\mathbf{PV})$$

$$\text{attn}\left(\mathbf{P} \begin{bmatrix} x_1 \\ \vdots \\ x_T \end{bmatrix}, \theta\right) = \mathbf{P} \text{softmax}\left(\frac{\mathbf{QK}^\top}{\sqrt{d_h}}\right) \mathbf{P}^\top (\mathbf{PV})$$

$\mathbf{P}$  swaps row  $i$  and  $j$      $\mathbf{P}^T$  swaps row  $j$  and  $i$                    $\mathbf{P}^\top \mathbf{P} = \mathbf{I}$

$$\text{attn}\left(\mathbf{P} \begin{bmatrix} x_1 \\ \vdots \\ x_T \end{bmatrix}, \theta\right) = \mathbf{P} \text{softmax}\left(\frac{\mathbf{QK}^\top}{\sqrt{d_h}}\right) \mathbf{V}$$

# Positional Encoding

$$\text{attn}\left(P \begin{bmatrix} x_1 \\ \vdots \\ x_T \end{bmatrix}, \theta\right) = P \text{softmax}\left(\frac{QK^\top}{\sqrt{d_h}}\right)V$$

# Positional Encoding

$$\text{attn}\left(P \begin{bmatrix} x_1 \\ \vdots \\ x_T \end{bmatrix}, \theta\right) = P \text{softmax}\left(\frac{QK^\top}{\sqrt{d_h}}\right)V$$

$$f\left(P \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}\right) = Pf\begin{pmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \end{pmatrix}$$

# Positional Encoding

$$\text{attn}\left(P \begin{bmatrix} x_1 \\ \vdots \\ x_T \end{bmatrix}, \theta\right) = P \text{softmax}\left(\frac{QK^\top}{\sqrt{d_h}}\right)V$$

$$f\left(P \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}\right) = Pf\begin{pmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \end{pmatrix}$$

**Question:** What does this mean?

# Positional Encoding

$$\text{attn}\left(P \begin{bmatrix} x_1 \\ \vdots \\ x_T \end{bmatrix}, \theta\right) = P \text{softmax}\left(\frac{QK^\top}{\sqrt{d_h}}\right)V$$
$$f\left(P \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}\right) = Pf\left(\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}\right)$$

**Question:** What does this mean?

**Answer:** Attention is equivariant, order **does not** matter

# Positional Encoding

This makes sense, in our party example we never consider the order

# Positional Encoding

This makes sense, in our party example we never consider the order



# Positional Encoding

This makes sense, in our party example we never consider the order



Transformer cannot determine order of inputs! **Equivariant** to ordering

# Positional Encoding

The following sentences are the same to a transformer

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \\ \mathbf{x}_5 \end{bmatrix} = \begin{bmatrix} \text{The} \\ \text{dog} \\ \text{licks} \\ \text{the} \\ \text{owner} \end{bmatrix}$$

# Positional Encoding

The following sentences are the same to a transformer

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \\ \mathbf{x}_5 \end{bmatrix} = \begin{bmatrix} \text{The} \\ \text{dog} \\ \text{licks} \\ \text{the} \\ \text{owner} \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_5 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \text{The} \\ \text{owner} \\ \text{licks} \\ \text{the} \\ \text{dog} \end{bmatrix}$$

# Positional Encoding

The following sentences are the same to a transformer

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \\ \mathbf{x}_5 \end{bmatrix} = \begin{bmatrix} \text{The} \\ \text{dog} \\ \text{licks} \\ \text{the} \\ \text{owner} \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_5 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \text{The} \\ \text{owner} \\ \text{licks} \\ \text{the} \\ \text{dog} \end{bmatrix}$$

This is a problem! For some tasks, we care about input order

# Positional Encoding

**Question:** What are some ways we can introduce ordering?

# Positional Encoding

**Question:** What are some ways we can introduce ordering?

**Answer 1:** We can introduce forgetting

# Positional Encoding

**Question:** What are some ways we can introduce ordering?

**Answer 1:** We can introduce forgetting

**ALiBi:** Press, Ofir, Noah Smith, and Mike Lewis. “Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation.” *International Conference on Learning Representations*.

# Positional Encoding

**Question:** What are some ways we can introduce ordering?

**Answer 1:** We can introduce forgetting

**ALiBi:** Press, Ofir, Noah Smith, and Mike Lewis. “*Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation.*” *International Conference on Learning Representations*.

**RoPE:** Su, Jianlin, et al. “*Roformer: Enhanced transformer with rotary position embedding.*” *Neurocomputing*.

# Positional Encoding

**Question:** What are some ways we can introduce ordering?

**Answer 1:** We can introduce forgetting

**ALiBi:** Press, Ofir, Noah Smith, and Mike Lewis. “*Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation.*” *International Conference on Learning Representations*.

**RoPE:** Su, Jianlin, et al. “*Roformer: Enhanced transformer with rotary position embedding.*” *Neurocomputing*.

**Answer 2:** We can modify the inputs based on their ordering

# Positional Encoding

**Question:** What are some ways we can introduce ordering?

**Answer 1:** We can introduce forgetting

**ALiBi:** Press, Ofir, Noah Smith, and Mike Lewis. “*Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation.*” *International Conference on Learning Representations*.

**RoPE:** Su, Jianlin, et al. “*Roformer: Enhanced transformer with rotary position embedding.*” *Neurocomputing*.

**Answer 2:** We can modify the inputs based on their ordering

We will focus on answer 2 because it is more common

# Positional Encoding

$$\text{attn} \left( \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_T \end{bmatrix}, \theta \right)$$

# Positional Encoding

$$\text{attn} \left( \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_T \end{bmatrix}, \theta \right)$$

$$\text{attn} \left( \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_T \end{bmatrix} + \begin{bmatrix} f_{\text{pos}}(1) \\ f_{\text{pos}}(2) \\ \vdots \\ f_{\text{pos}}(3) \end{bmatrix}, \theta \right)$$

# Positional Encoding

$$\text{attn} \left( \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_T \end{bmatrix}, \theta \right)$$

$$\text{attn} \left( \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_T \end{bmatrix} + \begin{bmatrix} f_{\text{pos}}(1) \\ f_{\text{pos}}(2) \\ \vdots \\ f_{\text{pos}}(3) \end{bmatrix}, \theta \right)$$

Even if we permute the inputs, we still know the order!

# Positional Encoding

$$\text{attn} \left( \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_T \end{bmatrix}, \theta \right)$$

$$\text{attn} \left( \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_T \end{bmatrix} + \begin{bmatrix} f_{\text{pos}}(1) \\ f_{\text{pos}}(2) \\ \vdots \\ f_{\text{pos}}(3) \end{bmatrix}, \theta \right)$$

Even if we permute the inputs, we still know the order!

$$\text{attn} \left( \begin{bmatrix} \mathbf{x}_T \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_2 \end{bmatrix} + \begin{bmatrix} f_{\text{pos}}(T) \\ f_{\text{pos}}(1) \\ \vdots \\ f_{\text{pos}}(2) \end{bmatrix}, \theta \right)$$

# Positional Encoding

$$\text{attn} \left( \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_T \end{bmatrix}, \theta \right)$$

$$\text{attn} \left( \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_T \end{bmatrix} + \begin{bmatrix} f_{\text{pos}}(1) \\ f_{\text{pos}}(2) \\ \vdots \\ f_{\text{pos}}(3) \end{bmatrix}, \theta \right)$$

Even if we permute the inputs, we still know the order!

$$\text{attn} \left( \begin{bmatrix} \mathbf{x}_T \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_2 \end{bmatrix} + \begin{bmatrix} f_{\text{pos}}(T) \\ f_{\text{pos}}(1) \\ \vdots \\ f_{\text{pos}}(2) \end{bmatrix}, \theta \right)$$

# Positional Encoding

What is  $f_{\text{pos}}$ ?

# Positional Encoding

What is  $f_{\text{pos}}$ ?

Easiest solution is just some random parameters

# Positional Encoding

What is  $f_{\text{pos}}$ ?

Easiest solution is just some random parameters

$$f(i) = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_T \end{bmatrix}_i$$

# Positional Encoding

What is  $f_{\text{pos}}$ ?

Easiest solution is just some random parameters

$$f(i) = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_T \end{bmatrix}_i$$

If ordering matters, learn  $\theta$  to be different

# Positional Encoding

What is  $f_{\text{pos}}$ ?

Easiest solution is just some random parameters

$$f(i) = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_T \end{bmatrix}_i$$

If ordering matters, learn  $\theta$  to be different

If order does not matter, learn  $\theta$  to be the same/zero

# Positional Encoding

What is  $f_{\text{pos}}$ ?

Easiest solution is just some random parameters

$$f(i) = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_T \end{bmatrix}_i$$

If ordering matters, learn  $\theta$  to be different

If order does not matter, learn  $\theta$  to be the same/zero

In torch, this is called `nn.Embedding`

# Positional Encoding

So, our final transformer is

```
class Transformer(nn.Module):
    def __init__(self):
        self.position_embedding = nn.Embedding(d_x)
        self.block1 = TransformerBlock()
        self.block2 = TransformerBlock()

    def forward(self, x):
        x = x + embedding(torch.arange(x.shape[0]))
        x = self.block1(x)
        x = self.block2(x)
        return x
```

# Positional Encoding

Let us code up the transformer in colab

<https://colab.research.google.com/drive/1qVlIbQKpTuBYIa7FvC4IH-kJq-E0jmc0d#scrollTo=iQtXjGYiz5CD>

# Applications

---

# Applications

Now that we understand the transformer, how do we use it?

# Applications

Now that we understand the transformer, how do we use it?

We can apply it to many domains, but today we will focus on text and images

# Applications

Now that we understand the transformer, how do we use it?

We can apply it to many domains, but today we will focus on text and images

# Text Transformers

---

# Text Transformers

In text transformers, we create a parameter for each word

# Text Transformers

In text transformers, we create a parameter for each word

We call these parameters **tokens**

# Text Transformers

In text transformers, we create a parameter for each word

We call these parameters **tokens**

The simple way is to create one parameter for each unique word in the dataset

# Text Transformers

In text transformers, we create a parameter for each word

We call these parameters **tokens**

The simple way is to create one parameter for each unique word in the dataset

Find unique words, and assign each one a parameter

# Text Transformers

In text transformers, we create a parameter for each word

We call these parameters **tokens**

The simple way is to create one parameter for each unique word in the dataset

Find unique words, and assign each one a parameter

$$\text{unique} \left( \begin{bmatrix} \text{John likes movies} \\ \text{Mary likes movies} \\ \text{I like dogs} \end{bmatrix} \right) = [\text{John likes movies Mary I dogs}]$$

# Text Transformers

Then assign a parameter to each word

$$\begin{bmatrix} \text{John} \\ \text{likes} \\ \text{movies} \\ \text{Mary} \\ I \\ \text{dogs} \end{bmatrix} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \end{bmatrix}$$

$$[\theta_1 \ \theta_2 \ \theta_3] =$$

# Text Transformers

Then assign a parameter to each word

$$\begin{bmatrix} \text{John} \\ \text{likes} \\ \text{movies} \\ \text{Mary} \\ I \\ \text{dogs} \end{bmatrix} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \end{bmatrix}$$

$$[\theta_1 \ \theta_2 \ \theta_3] = \text{John likes movies}$$

$$[\theta_4 \ \theta_2 \ \theta_1] =$$

# Text Transformers

Then assign a parameter to each word

$$\begin{bmatrix} \text{John} \\ \text{likes} \\ \text{movies} \\ \text{Mary} \\ I \\ \text{dogs} \end{bmatrix} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \end{bmatrix}$$

$$[\theta_1 \ \theta_2 \ \theta_3] = \text{John likes movies}$$

$$[\theta_4 \ \theta_2 \ \theta_1] = \text{Mary likes John}$$

# Text Transformers

```
unique_words = set(sentence.split(" ")) for sentence in xs)
tokens = {word: i for i, word in enumerate(unique_words)}
embeddings = nn.Embedding(len(tokens), d_x)
# Convert from words to parameters
xs = []
for sentence in sentences:
    xs.append([])
    for word in sentence:
        index = embeddings[word]
        token = tokens[index]
        xs.append(token)

print(xs)
>>> [[Tensor(...), Tensor(...), ...]]
```

# Text Transformers

```
model = Transformer()
for tokenized_sentence in xs:
    # Convert list to tensor
    x = torch.stack(tokenized_sentence)
    y = model(x)
```

# Image Transformers

---

# Image Transformers

In image transformers, we treat a **patch** of pixels as an  $x$

# Image Transformers

In image transformers, we treat a **patch** of pixels as an  $x$

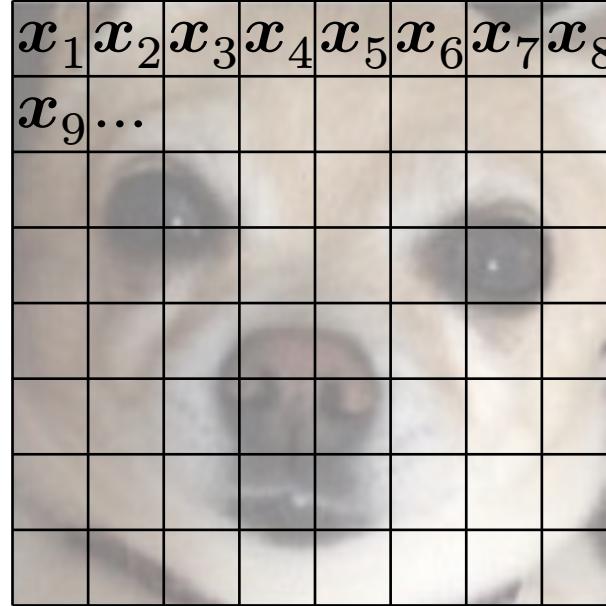
$$X \in [0, 1]^{16 \times 16}$$

# Image Transformers

In image transformers, we treat a **patch** of pixels as an  $x$

$$X \in [0, 1]^{16 \times 16}$$

$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$	$x_8$
$x_9$	...						



# Image Transformers

```
# Convert image into patches
patches = []
for i in range(0, x.shape[0] - k + 1, k):
    for j in range(0, x.shape[1] - k + 1, k):
        patches.append(
            x[i: i + k , j: j + k]
)
patches = stack(patches, axis=0)
print(patches.shape)
>>> (T, k, k)

model = Transformer()
y = model(patches)
```

# Unsupervised Training

---

# Unsupervised Training

**Question:** How do we train transformers?

# Unsupervised Training

**Question:** How do we train transformers?

**Answer:** Can train just like other neural networks

# Unsupervised Training

**Question:** How do we train transformers?

**Answer:** Can train just like other neural networks

- Classification loss

# Unsupervised Training

**Question:** How do we train transformers?

**Answer:** Can train just like other neural networks

- Classification loss
- Regression loss

# Unsupervised Training

**Question:** How do we train transformers?

**Answer:** Can train just like other neural networks

- Classification loss
- Regression loss

In practice, transformers require lots of training data

# Unsupervised Training

**Question:** How do we train transformers?

**Answer:** Can train just like other neural networks

- Classification loss
- Regression loss

In practice, transformers require lots of training data

There is almost infinite empirical scaling – add more data, model becomes stronger

# Unsupervised Training

**Question:** How do we train transformers?

**Answer:** Can train just like other neural networks

- Classification loss
- Regression loss

In practice, transformers require lots of training data

There is almost infinite empirical scaling – add more data, model becomes stronger

Transformers are limited by the size of datasets today

# Unsupervised Training

**Question:** How do we train transformers?

**Answer:** Can train just like other neural networks

- Classification loss
- Regression loss

In practice, transformers require lots of training data

There is almost infinite empirical scaling – add more data, model becomes stronger

Transformers are limited by the size of datasets today

There are not enough students to label training data!

# Unsupervised Training

**Question:** How can we train transformers if we cannot create big enough datasets?

# Unsupervised Training

**Question:** How can we train transformers if we cannot create big enough datasets?

**Answer:** We can use **unsupervised learning**

# Unsupervised Training

**Question:** How can we train transformers if we cannot create big enough datasets?

**Answer:** We can use **unsupervised learning**

The internet contains billions of unlabeled sentences and images

# Unsupervised Training

**Question:** How can we train transformers if we cannot create big enough datasets?

**Answer:** We can use **unsupervised learning**

The internet contains billions of unlabeled sentences and images

We can mask/hide part of the input, and train the model to predict the missing parts

# Unsupervised Training

**Question:** How can we train transformers if we cannot create big enough datasets?

**Answer:** We can use **unsupervised learning**

The internet contains billions of unlabeled sentences and images

We can mask/hide part of the input, and train the model to predict the missing parts

Another name for this is **generative pre-training** (GPT)

# Unsupervised Training

**Question:** How can we train transformers if we cannot create big enough datasets?

**Answer:** We can use **unsupervised learning**

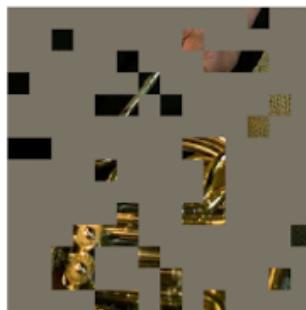
The internet contains billions of unlabeled sentences and images

We can mask/hide part of the input, and train the model to predict the missing parts

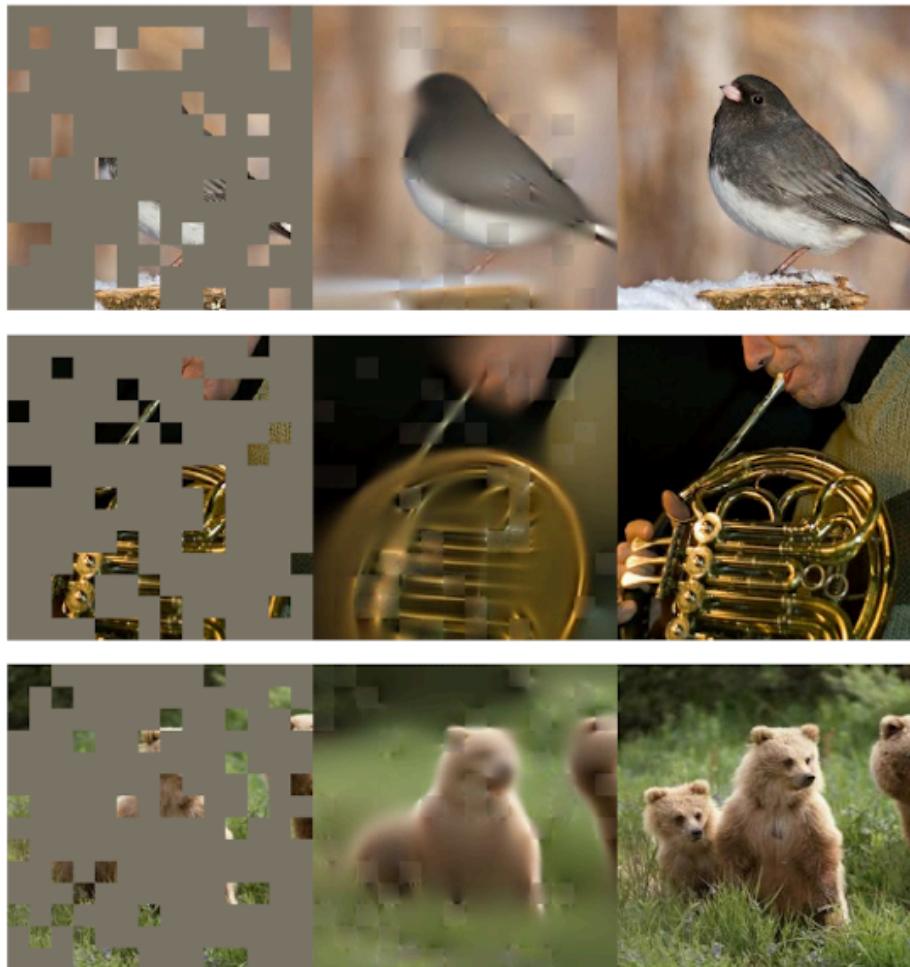
Another name for this is **generative pre-training** (GPT)

This method is **extremely** powerful

# Unsupervised Training

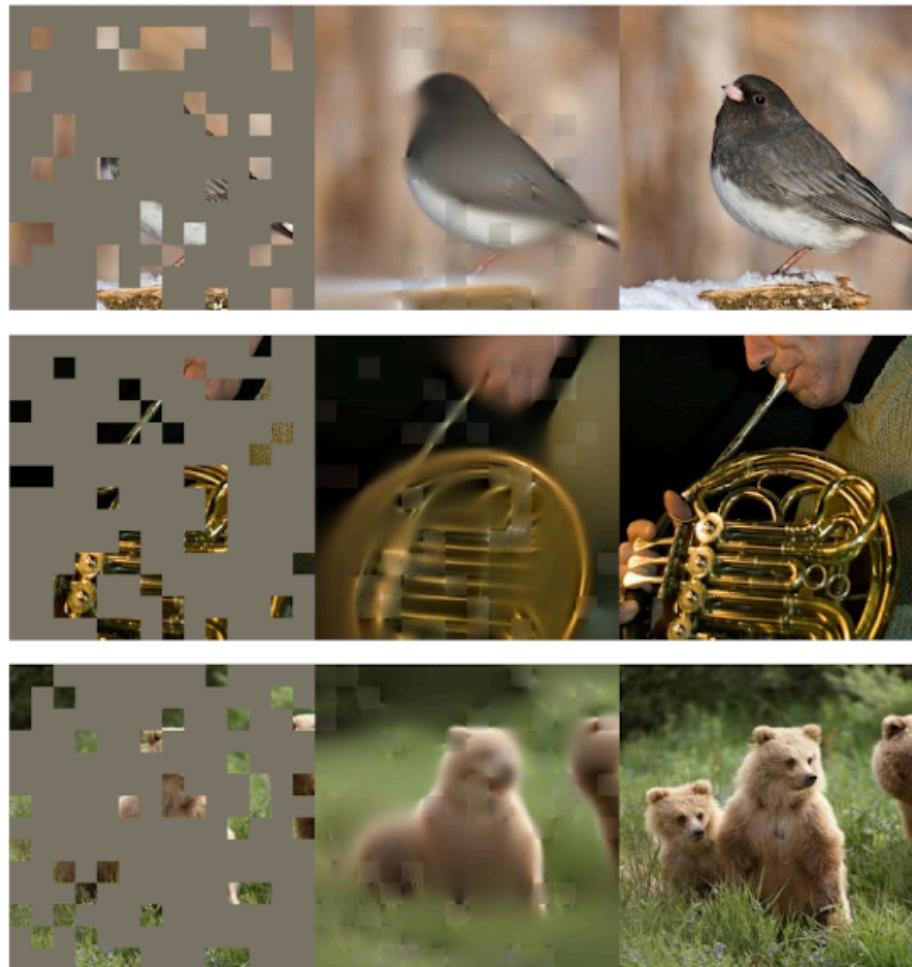


# Unsupervised Training



*He, Kaiming, et al. “Masked autoencoders are scalable vision learners.” Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022.*

# Unsupervised Training



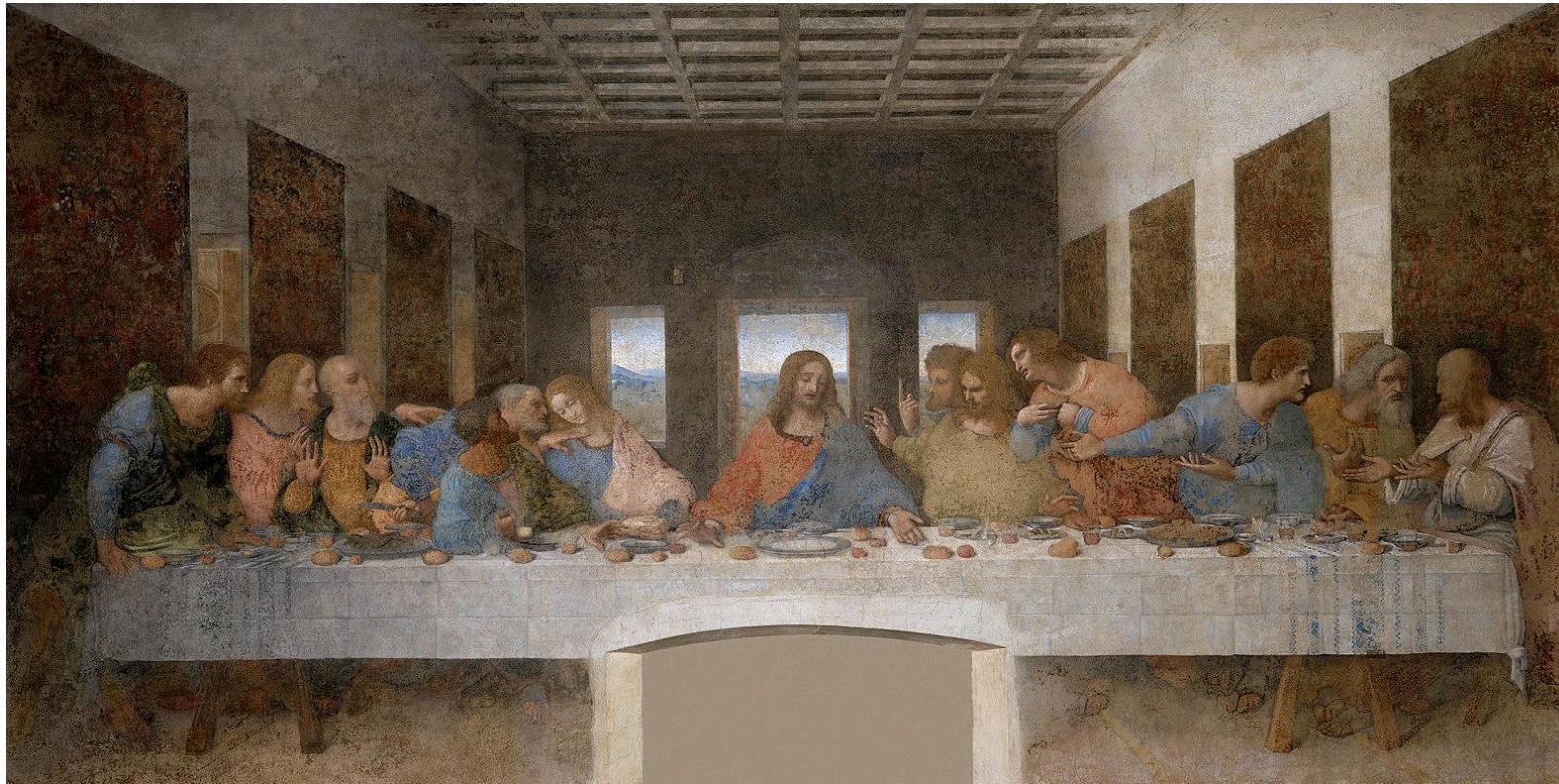
*He, Kaiming, et al. “Masked autoencoders are scalable vision learners.” Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022.*

# Unsupervised Training

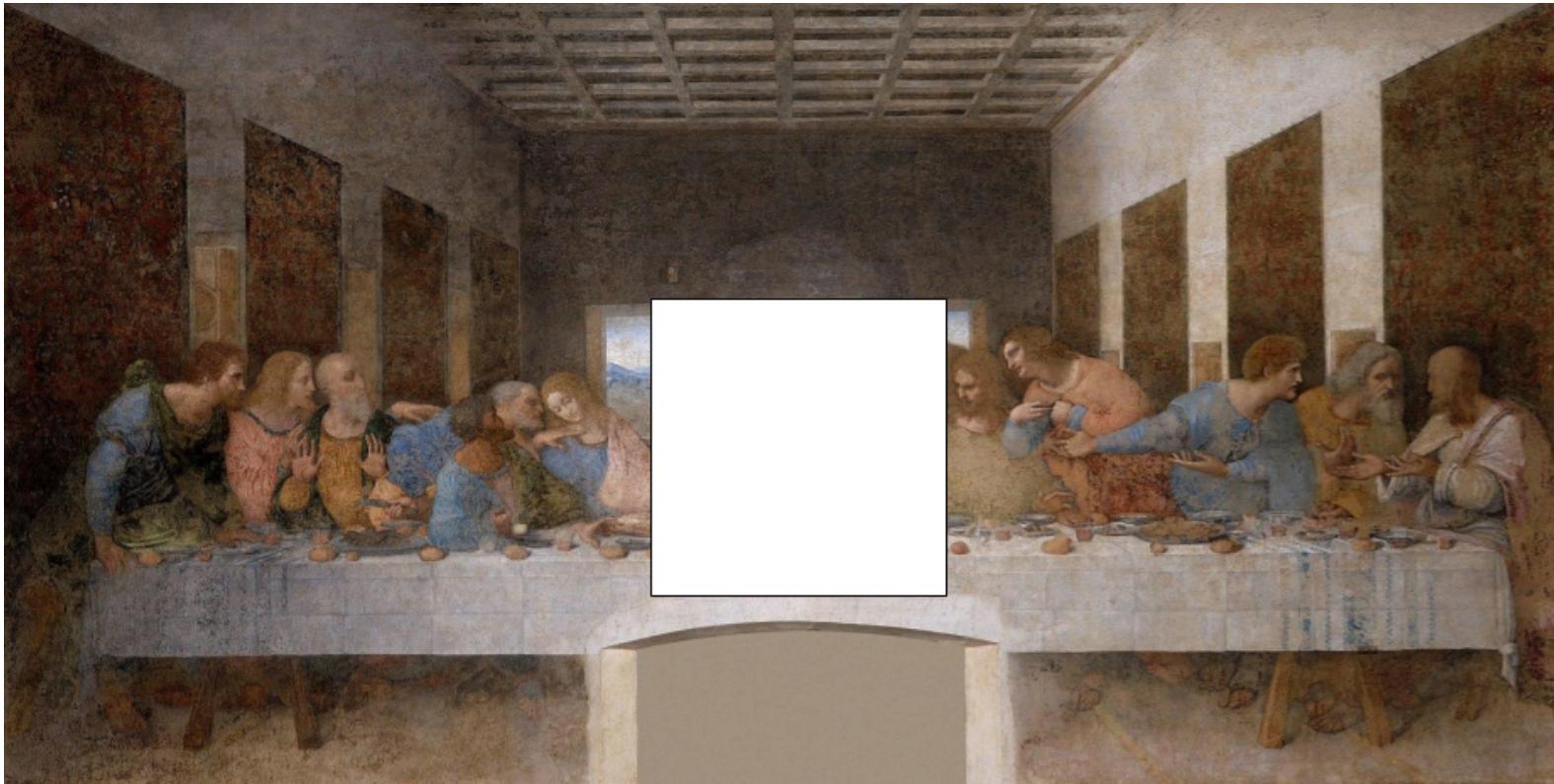
Anyone familiar with Da Vinci's painting *The Last Supper*?

# Unsupervised Training

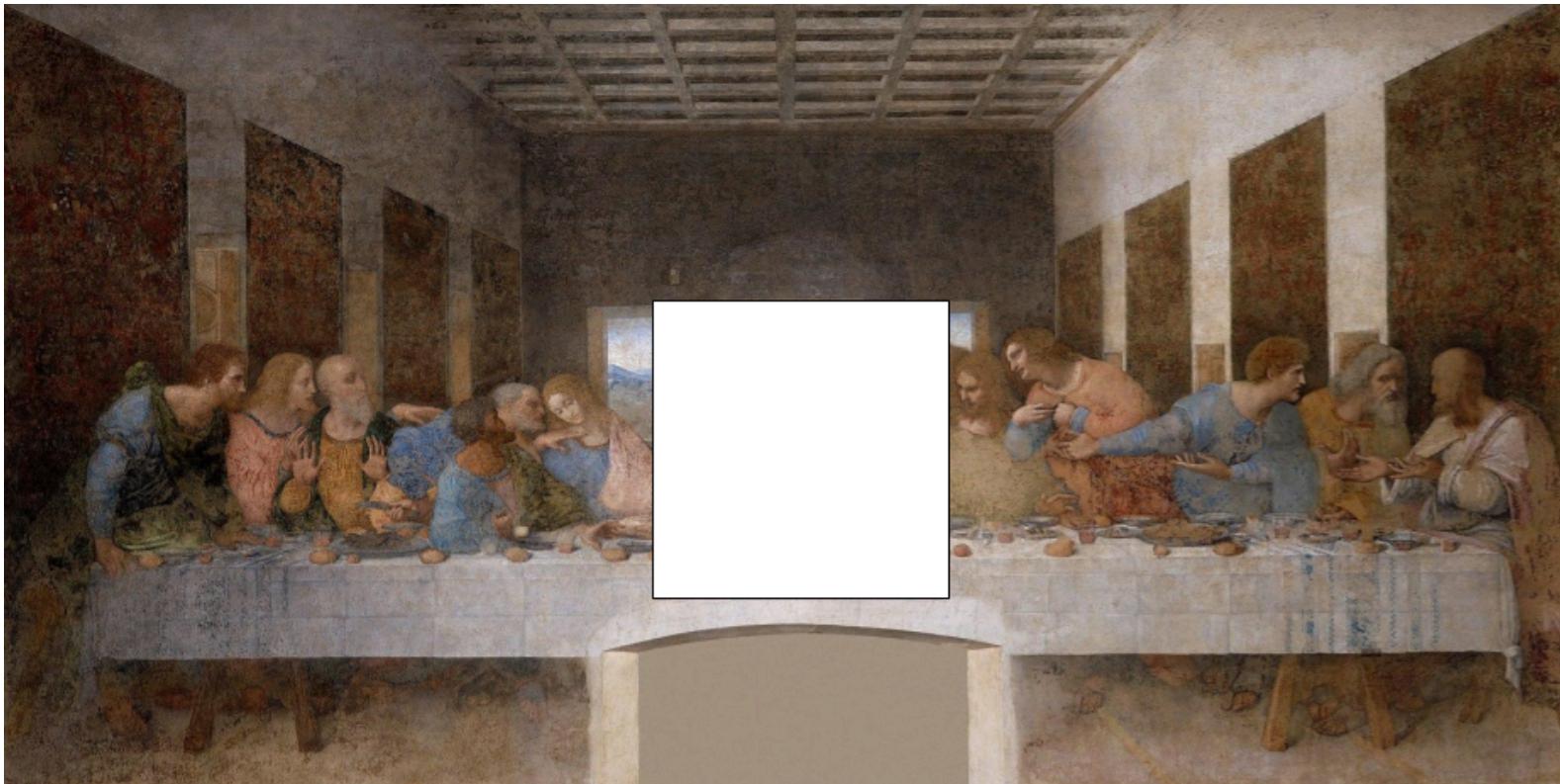
Anyone familiar with Da Vinci's painting *The Last Supper*?



# Unsupervised Training



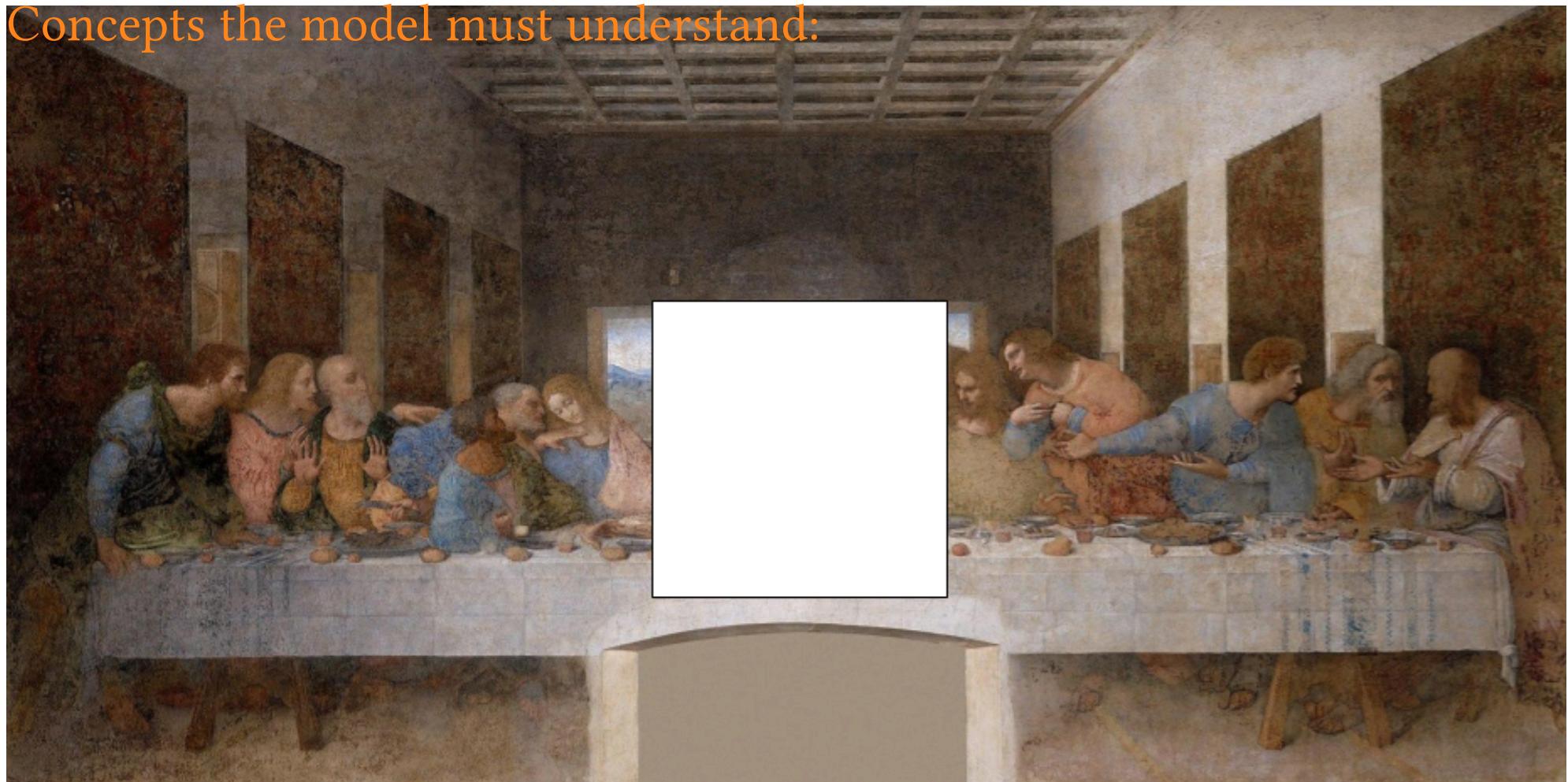
# Unsupervised Training



**Question:** What concepts does the vision transformer need to understand to predict the missing pixels?

# Unsupervised Training

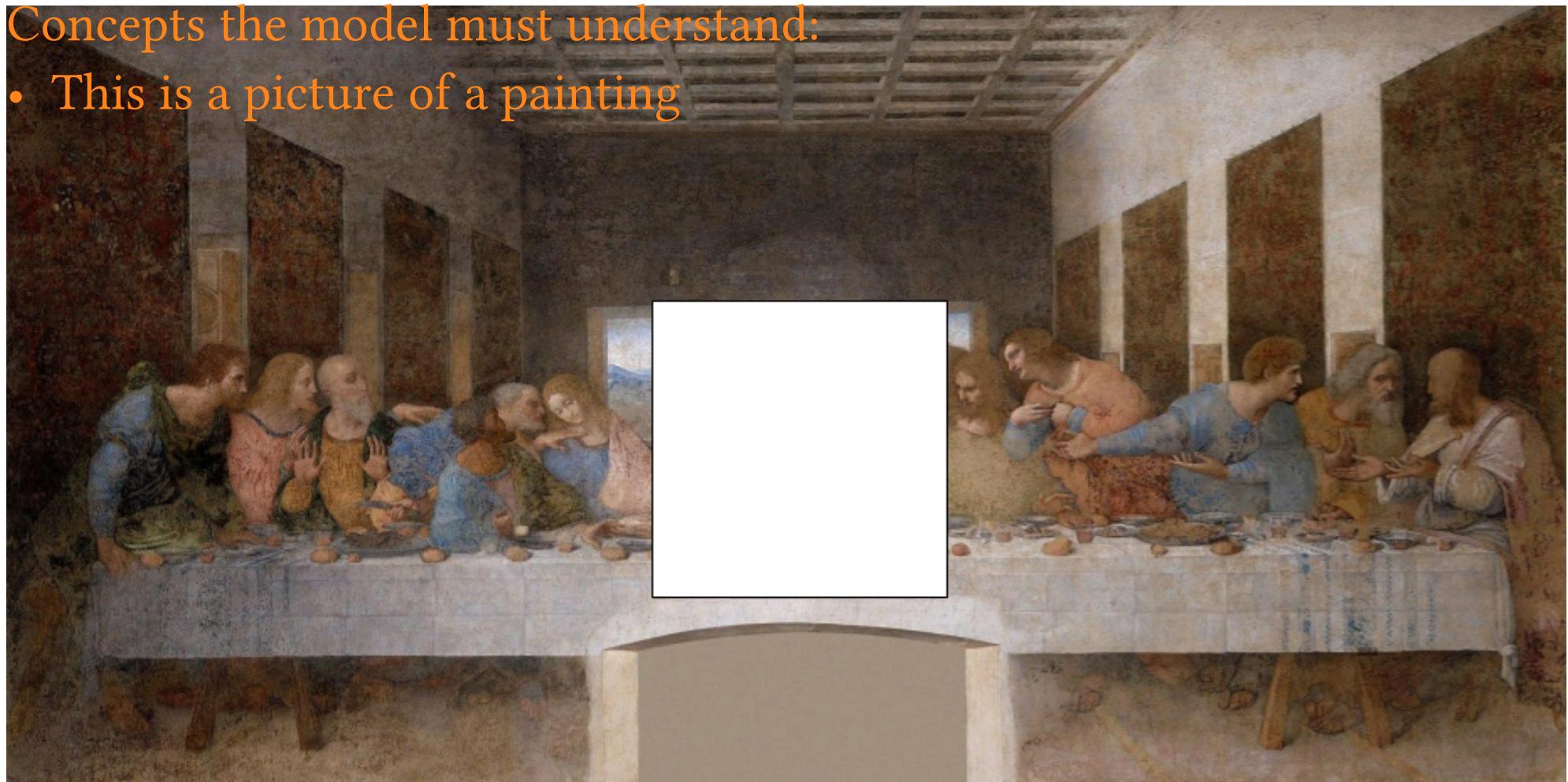
Concepts the model must understand:



# Unsupervised Training

Concepts the model must understand:

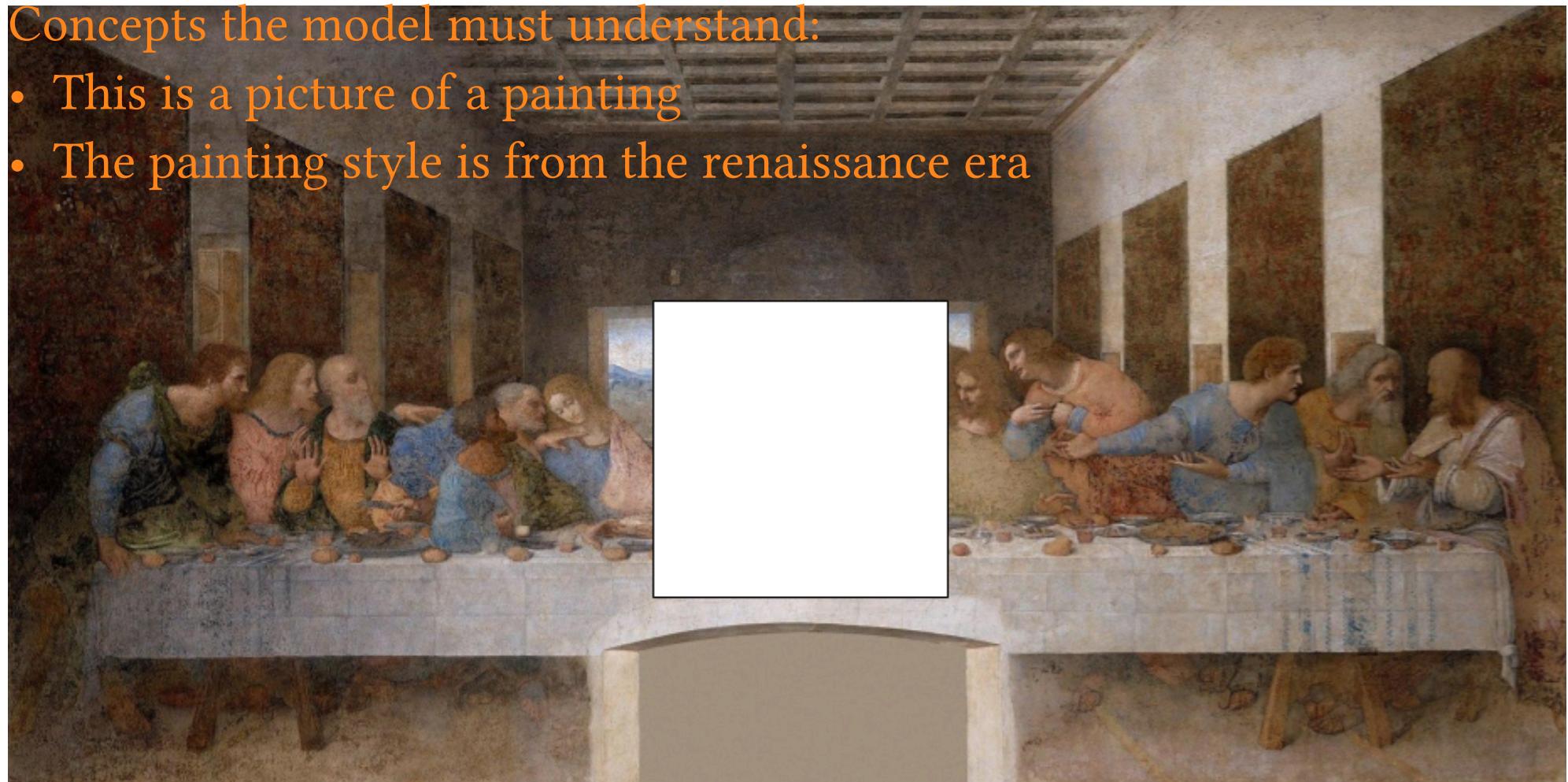
- This is a picture of a painting



# Unsupervised Training

Concepts the model must understand:

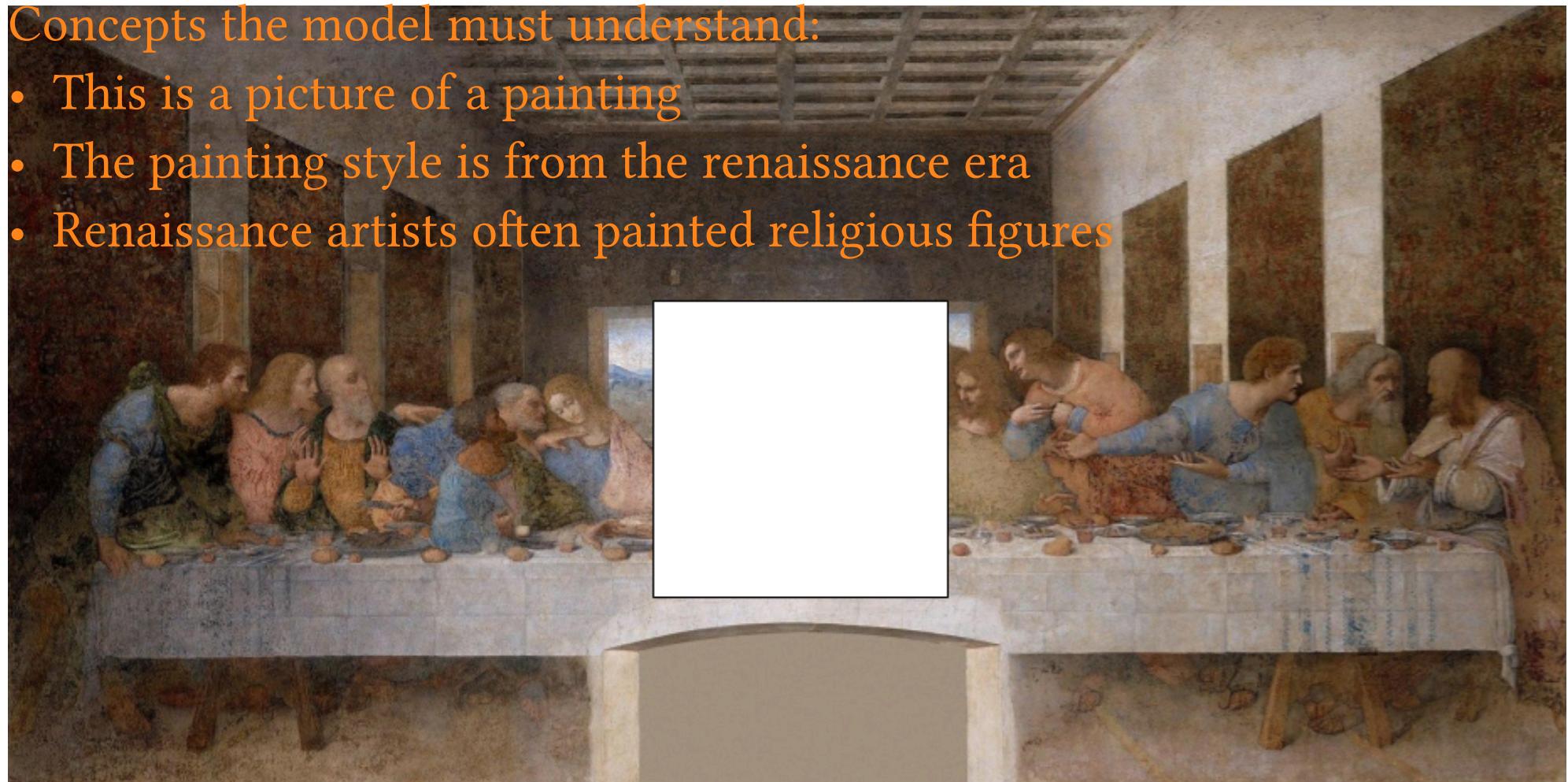
- This is a picture of a painting
- The painting style is from the renaissance era



# Unsupervised Training

Concepts the model must understand:

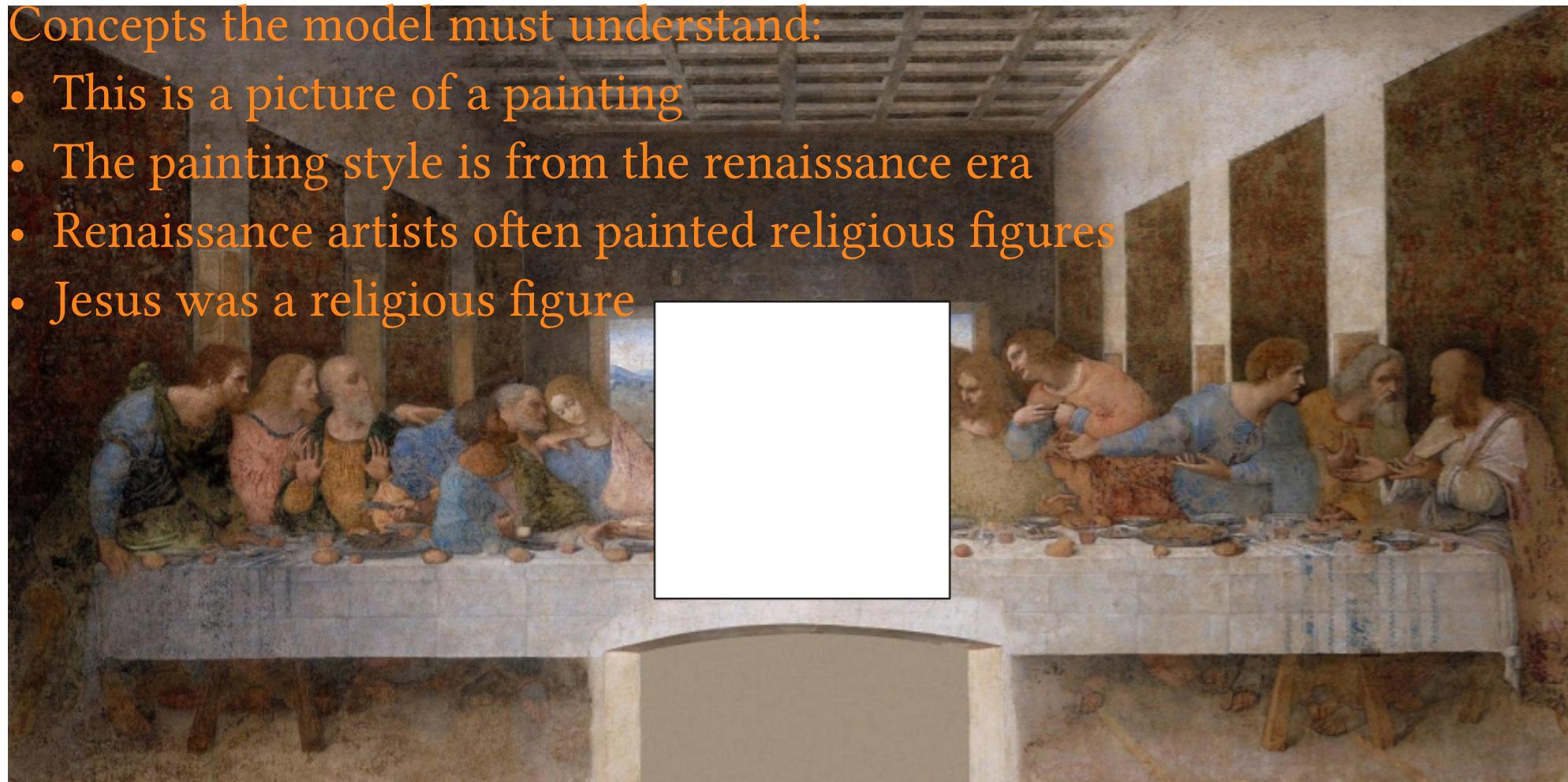
- This is a picture of a painting
- The painting style is from the renaissance era
- Renaissance artists often painted religious figures



# Unsupervised Training

Concepts the model must understand:

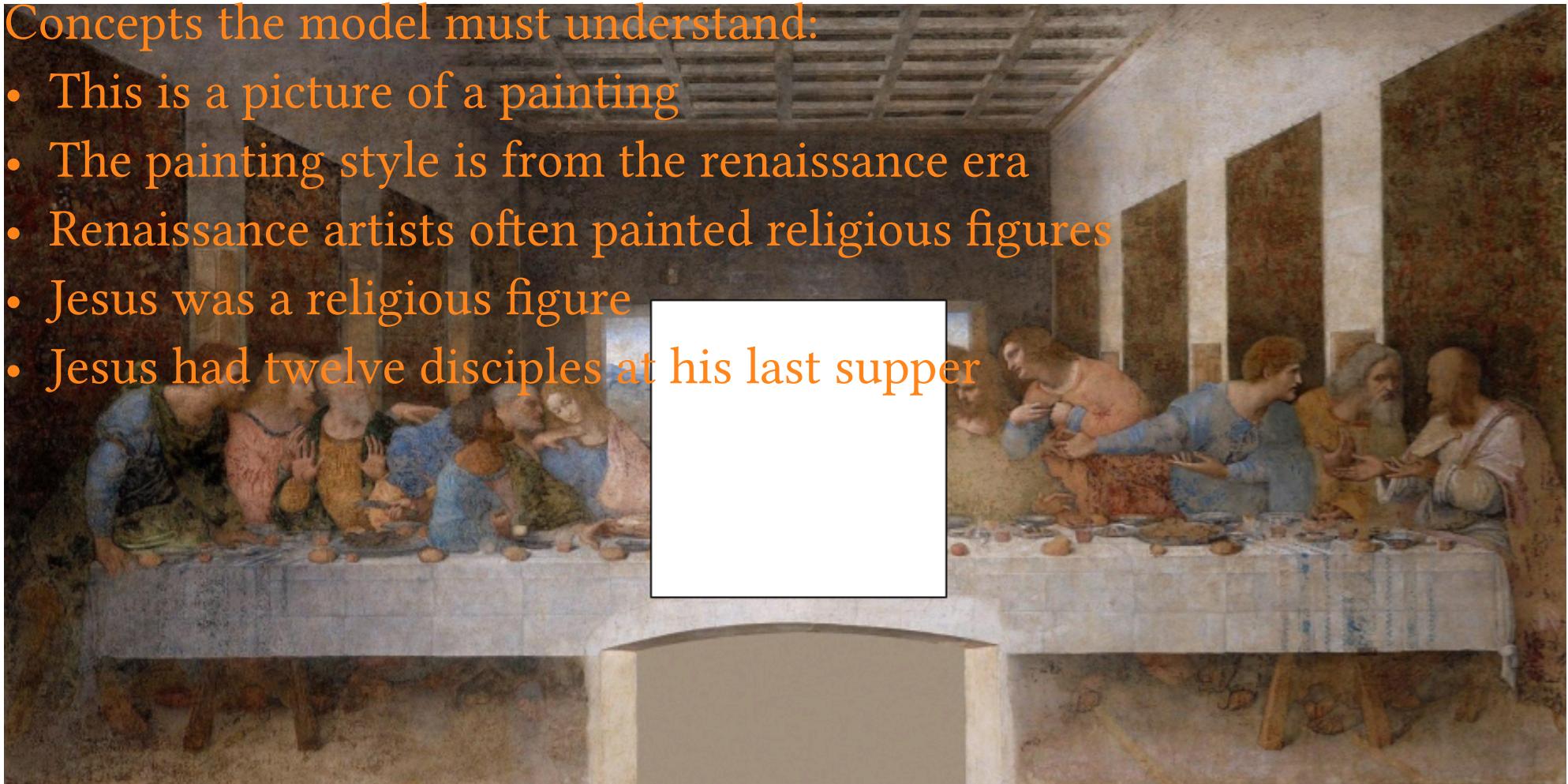
- This is a picture of a painting
- The painting style is from the renaissance era
- Renaissance artists often painted religious figures
- Jesus was a religious figure



# Unsupervised Training

Concepts the model must understand:

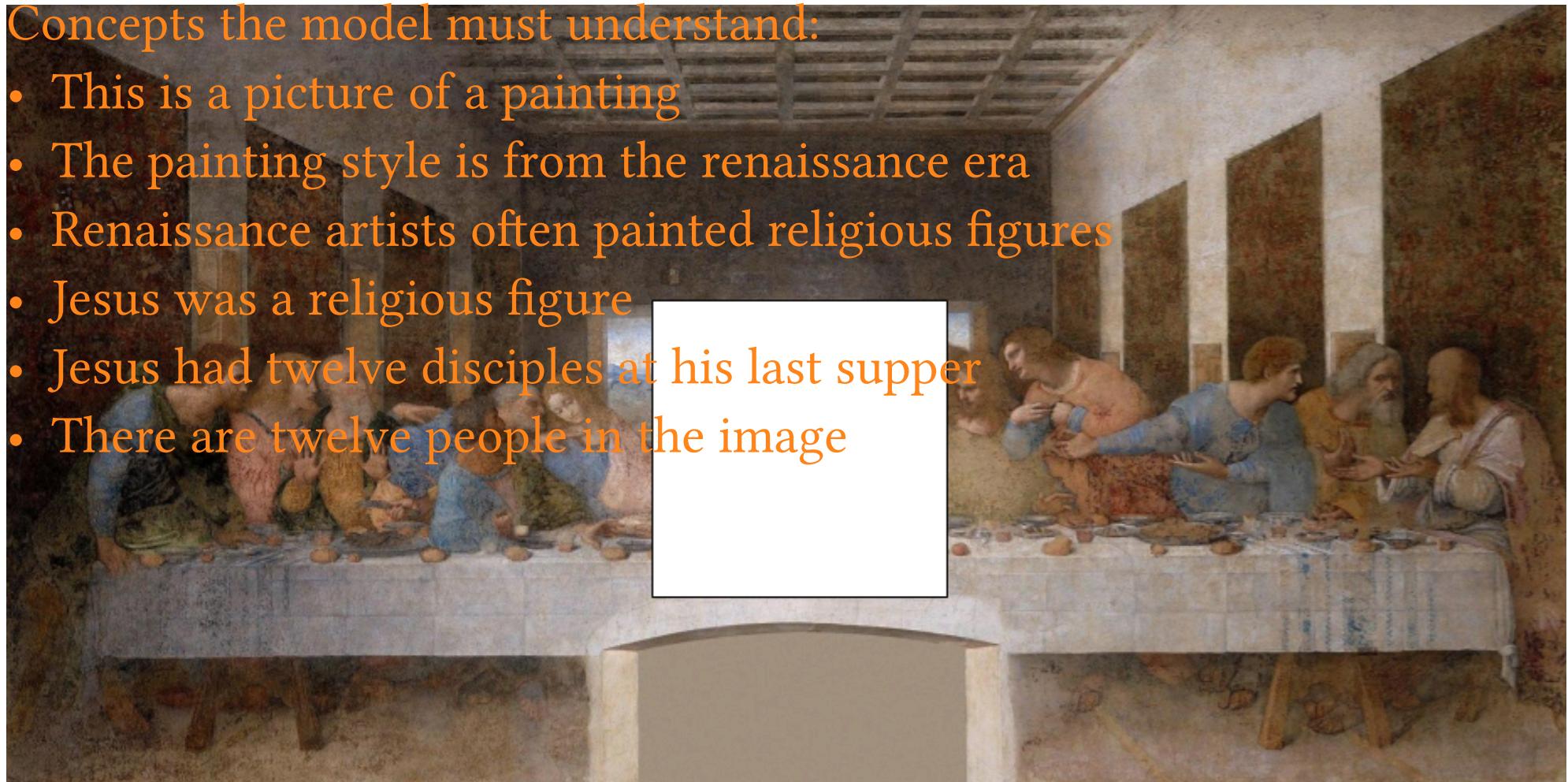
- This is a picture of a painting
- The painting style is from the renaissance era
- Renaissance artists often painted religious figures
- Jesus was a religious figure
- Jesus had twelve disciples at his last supper



# Unsupervised Training

Concepts the model must understand:

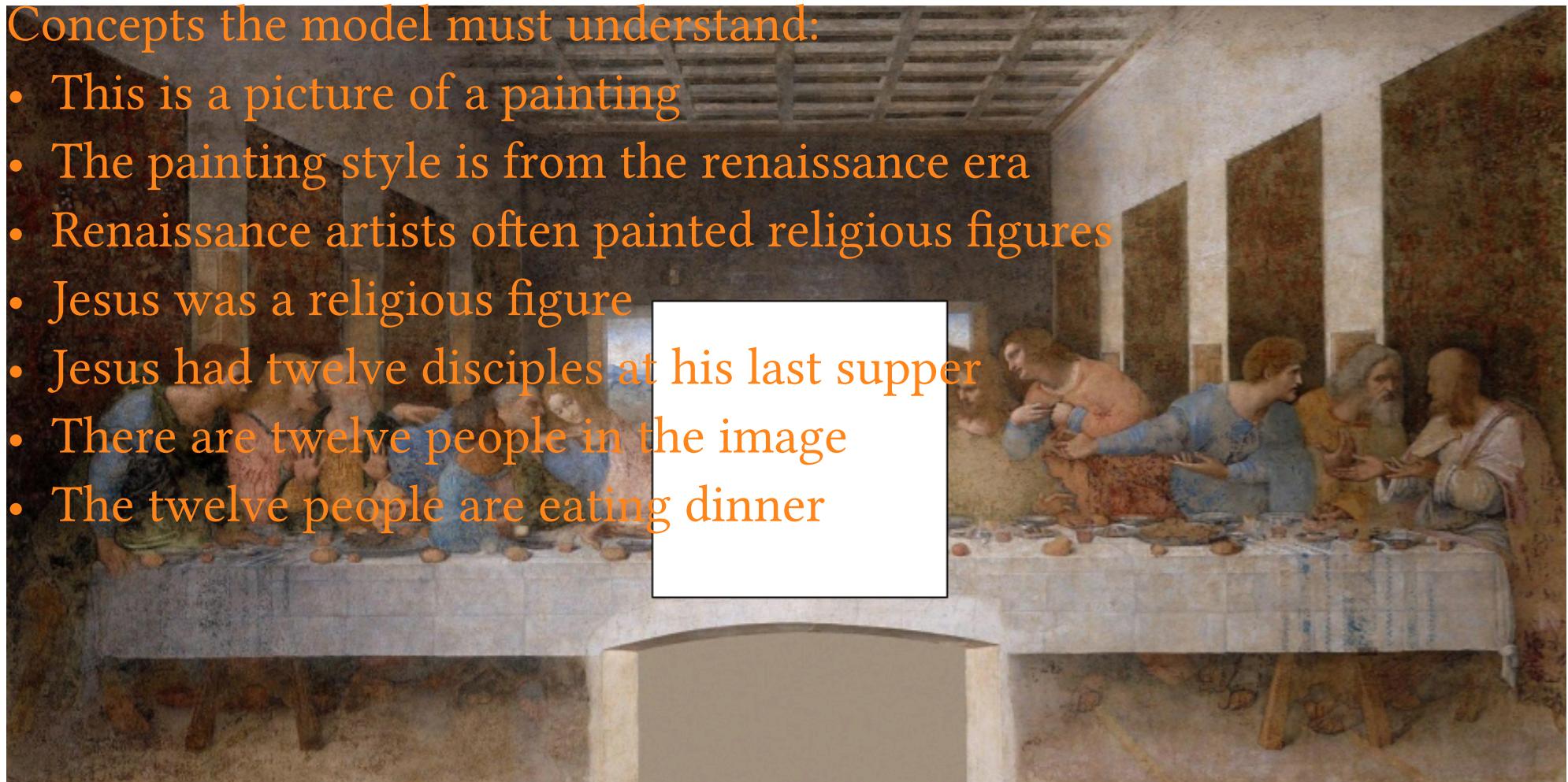
- This is a picture of a painting
- The painting style is from the renaissance era
- Renaissance artists often painted religious figures
- Jesus was a religious figure
- Jesus had twelve disciples at his last supper
- There are twelve people in the image



# Unsupervised Training

Concepts the model must understand:

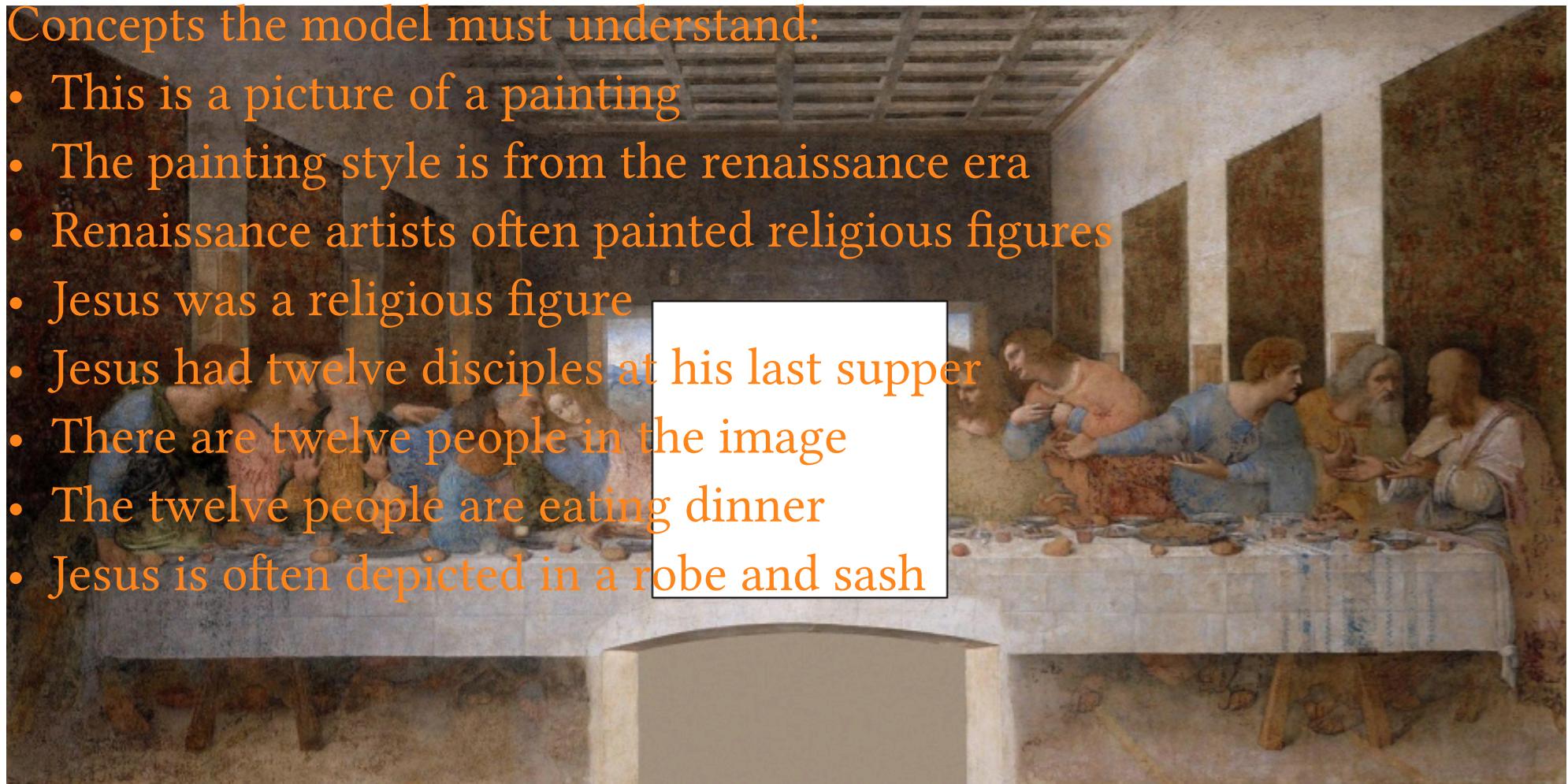
- This is a picture of a painting
- The painting style is from the renaissance era
- Renaissance artists often painted religious figures
- Jesus was a religious figure
- Jesus had twelve disciples at his last supper
- There are twelve people in the image
- The twelve people are eating dinner



# Unsupervised Training

Concepts the model must understand:

- This is a picture of a painting
- The painting style is from the renaissance era
- Renaissance artists often painted religious figures
- Jesus was a religious figure
- Jesus had twelve disciples at his last supper
- There are twelve people in the image
- The twelve people are eating dinner
- Jesus is often depicted in a robe and sash



# Unsupervised Training

We train the model to fix the image

# Unsupervised Training

We train the model to fix the image

To fix the image, the model must understand so much of our world

# Unsupervised Training

We train the model to fix the image

To fix the image, the model must understand so much of our world

This is the power of generative pre-training

# Unsupervised Training

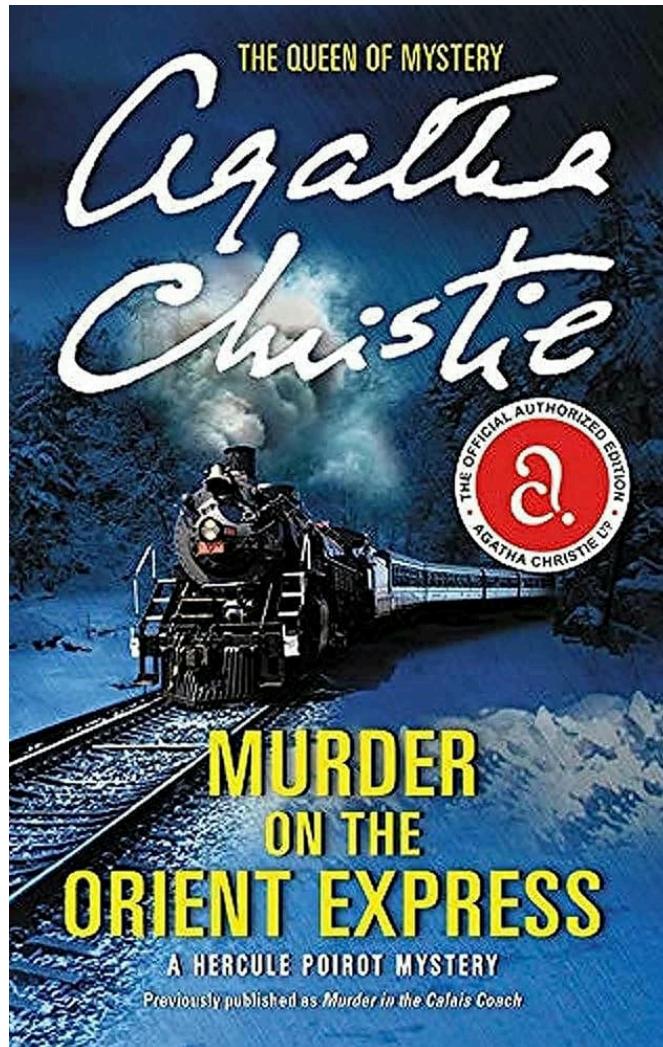
We train the model to fix the image

To fix the image, the model must understand so much of our world

This is the power of generative pre-training

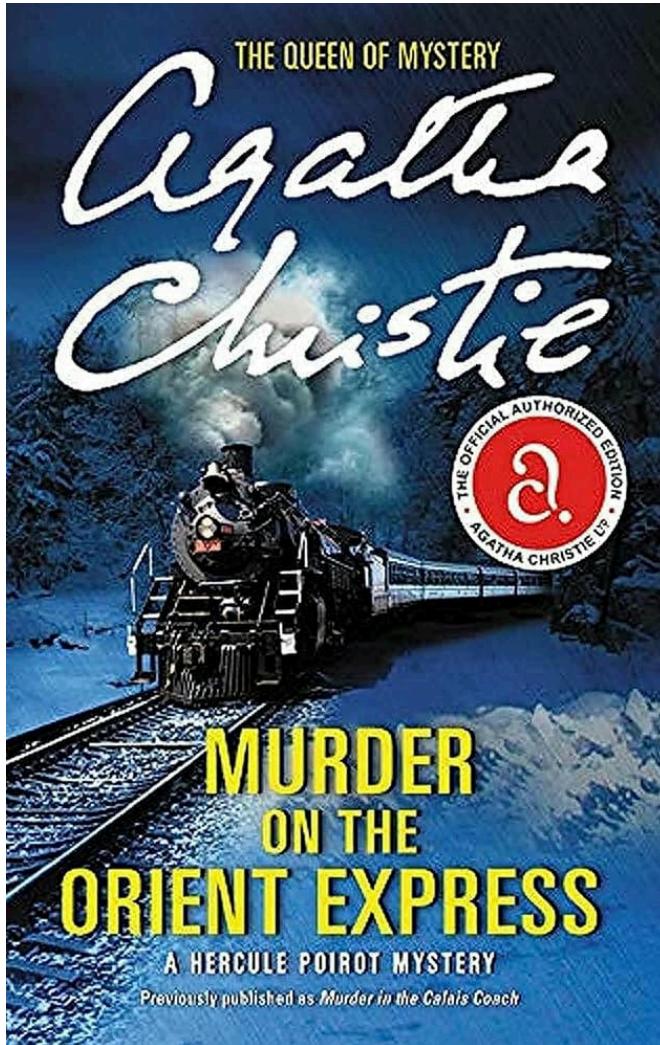
What about text transformers?

# Unsupervised Training



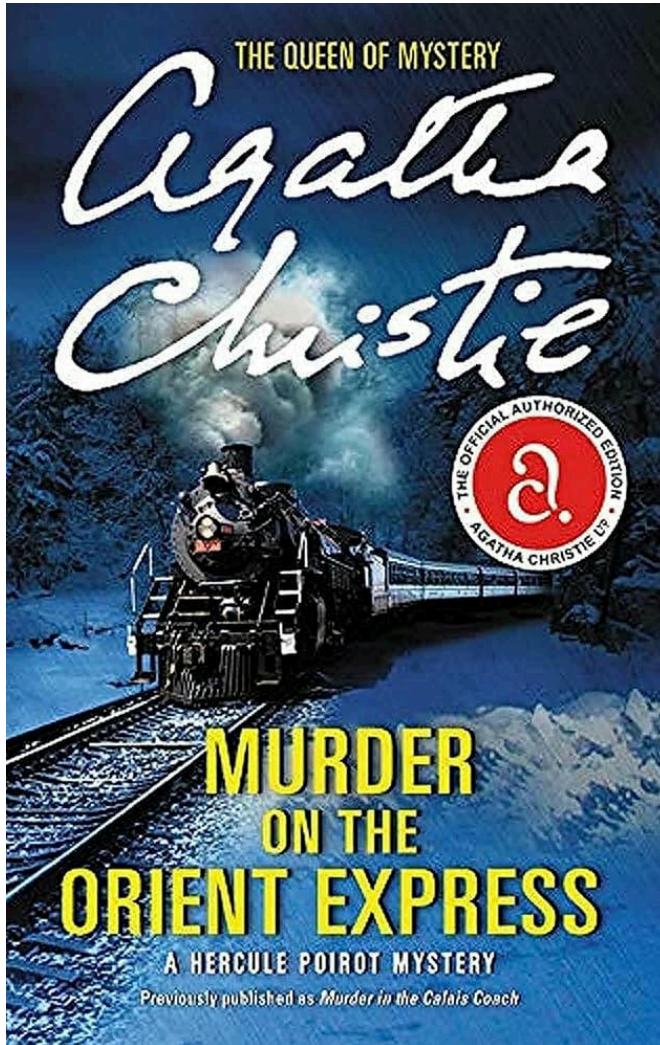
This is a mystery novel

# Unsupervised Training



This is a mystery novel  
Clues, intrigue, murder, etc

# Unsupervised Training

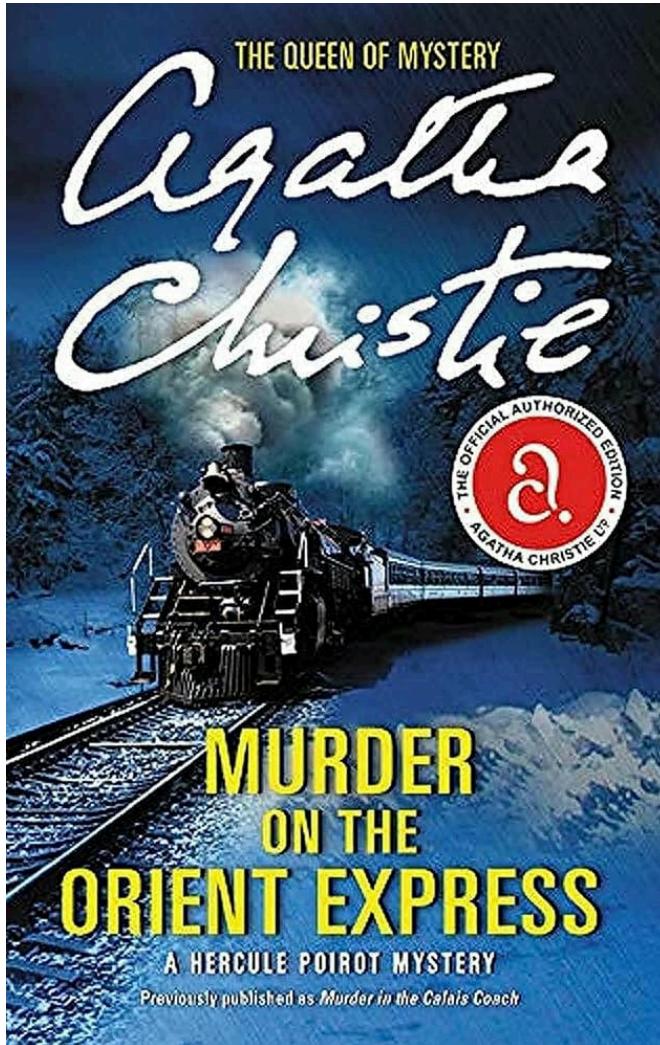


This is a mystery novel

Clues, intrigue, murder, etc

“Ah, said inspector Poirot, the murderer must be \_\_\_\_\_.”

# Unsupervised Training



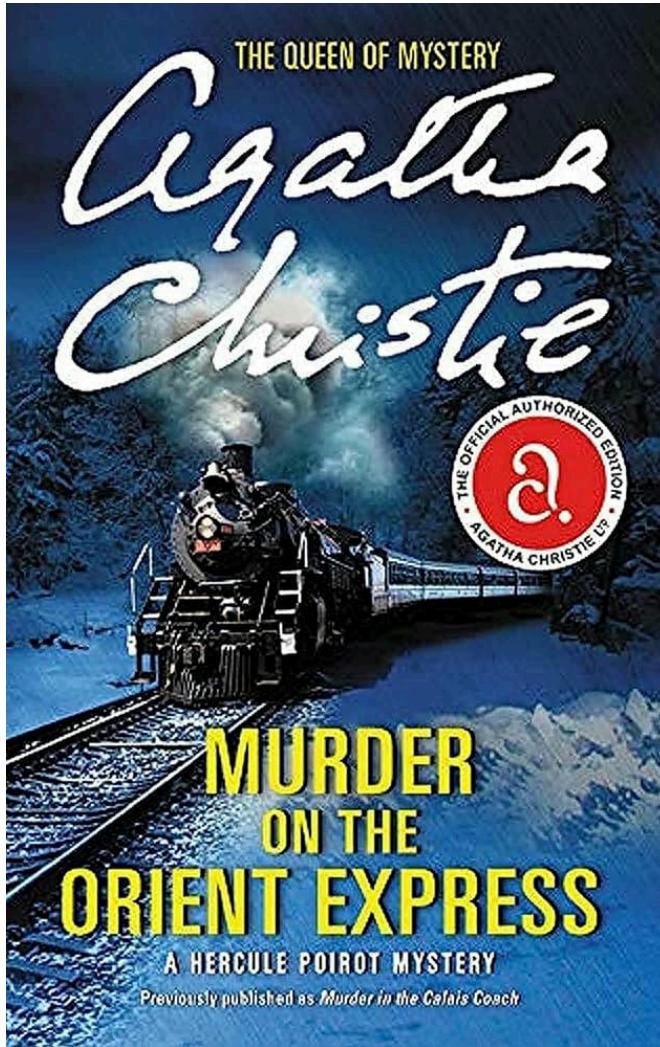
This is a mystery novel

Clues, intrigue, murder, etc

“Ah, said inspector Poirot, the murderer must be \_\_\_\_\_.”

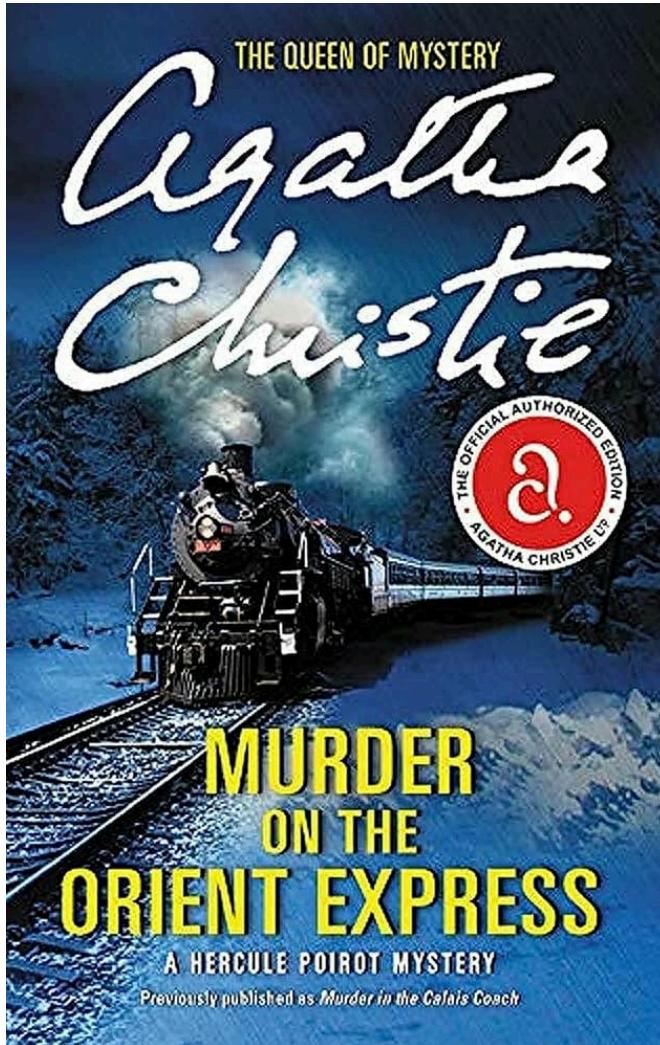
To complete the sentence, the model must understand:

# Unsupervised Training



To complete the sentence, the model must understand:

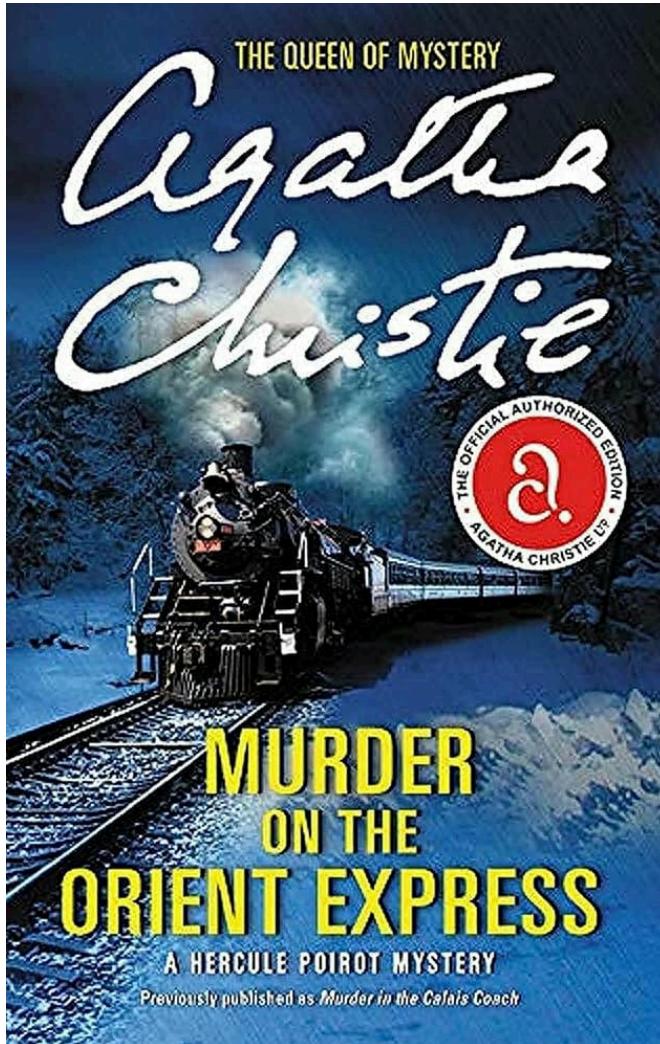
# Unsupervised Training



To complete the sentence, the model must understand:

- What a murder is

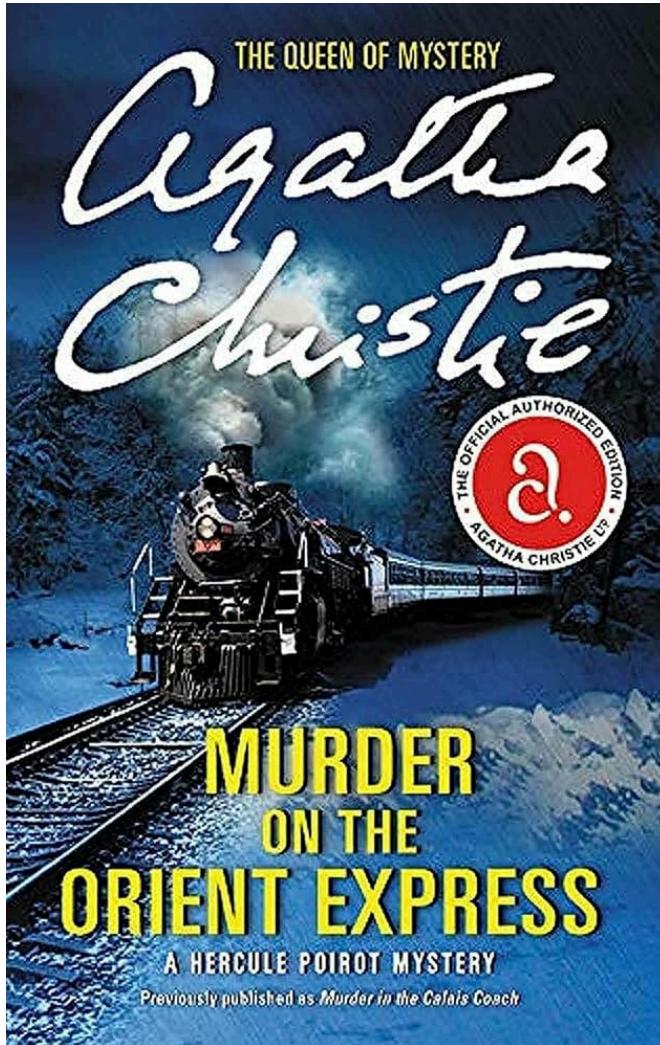
# Unsupervised Training



To complete the sentence, the model must understand:

- What a murder is
- What it means to be alive

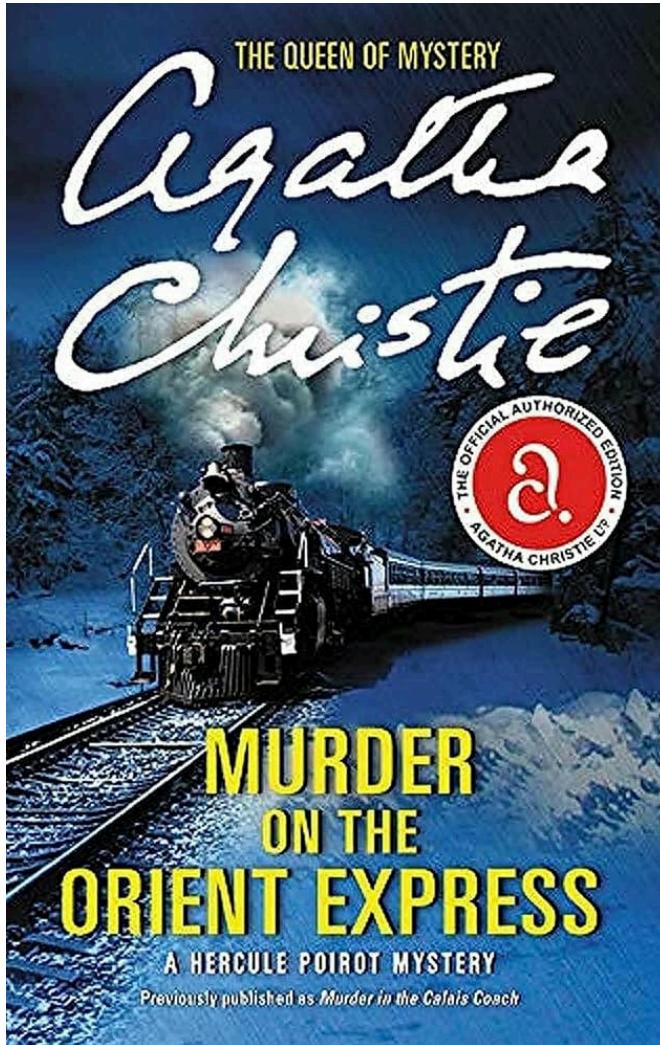
# Unsupervised Training



To complete the sentence, the model must understand:

- What a murder is
- What it means to be alive
- Emotions like anger, jealousy, betrayal, love

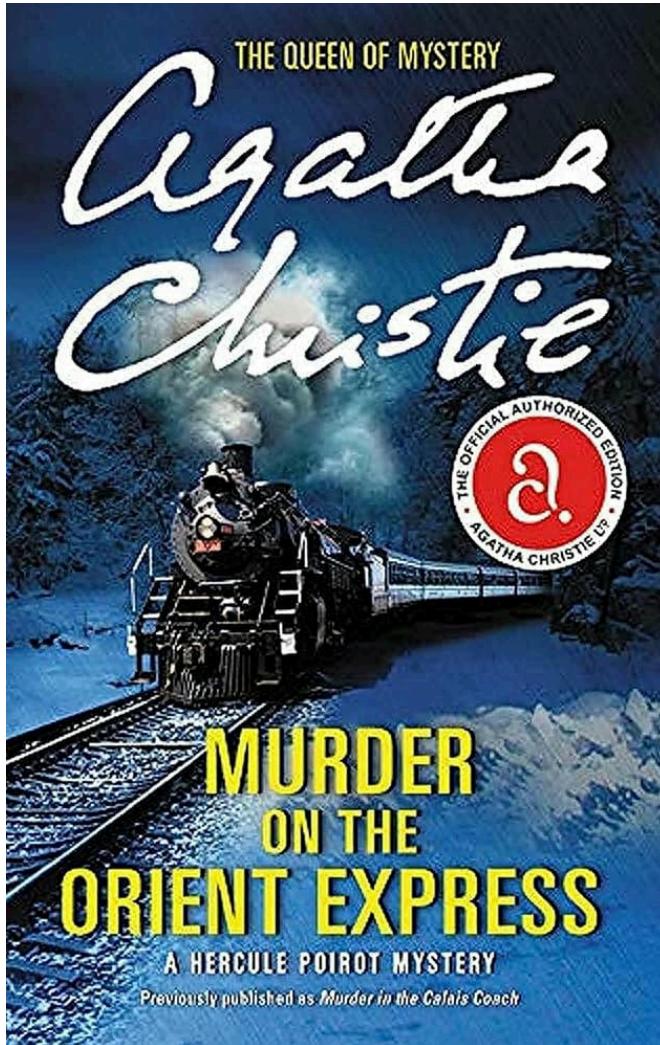
# Unsupervised Training



To complete the sentence, the model must understand:

- What a murder is
- What it means to be alive
- Emotions like anger, jealousy, betrayal, love
- Personalities of each character

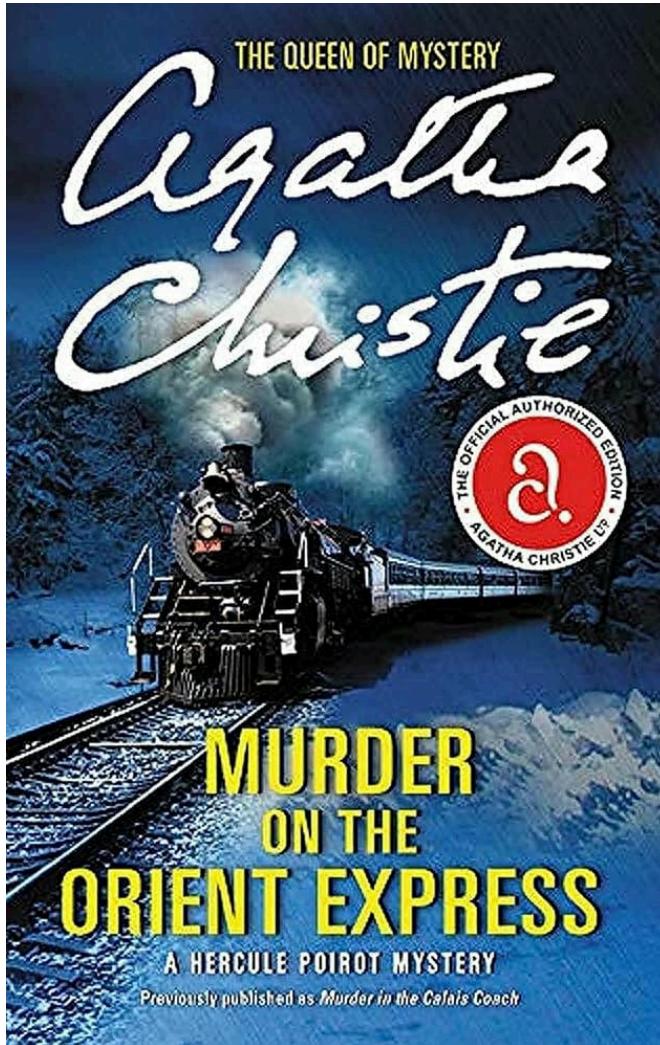
# Unsupervised Training



To complete the sentence, the model must understand:

- What a murder is
- What it means to be alive
- Emotions like anger, jealousy, betrayal, love
- Personalities of each character
- Why a human would murder another human

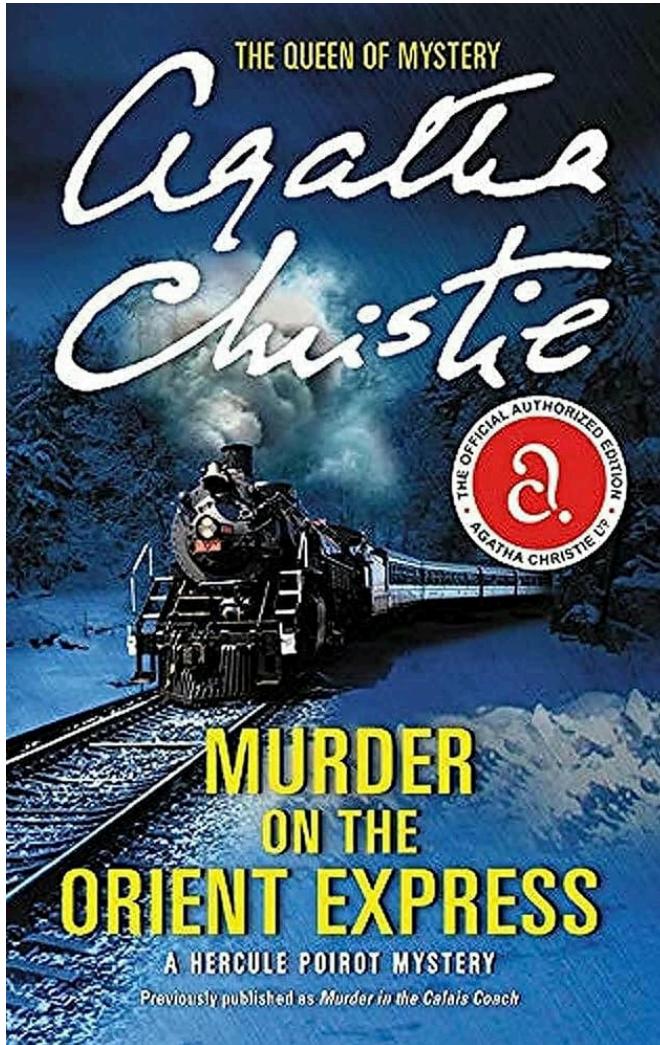
# Unsupervised Training



To complete the sentence, the model must understand:

- What a murder is
- What it means to be alive
- Emotions like anger, jealousy, betrayal, love
- Personalities of each character
- Why a human would murder another human
- How humans react to emotions

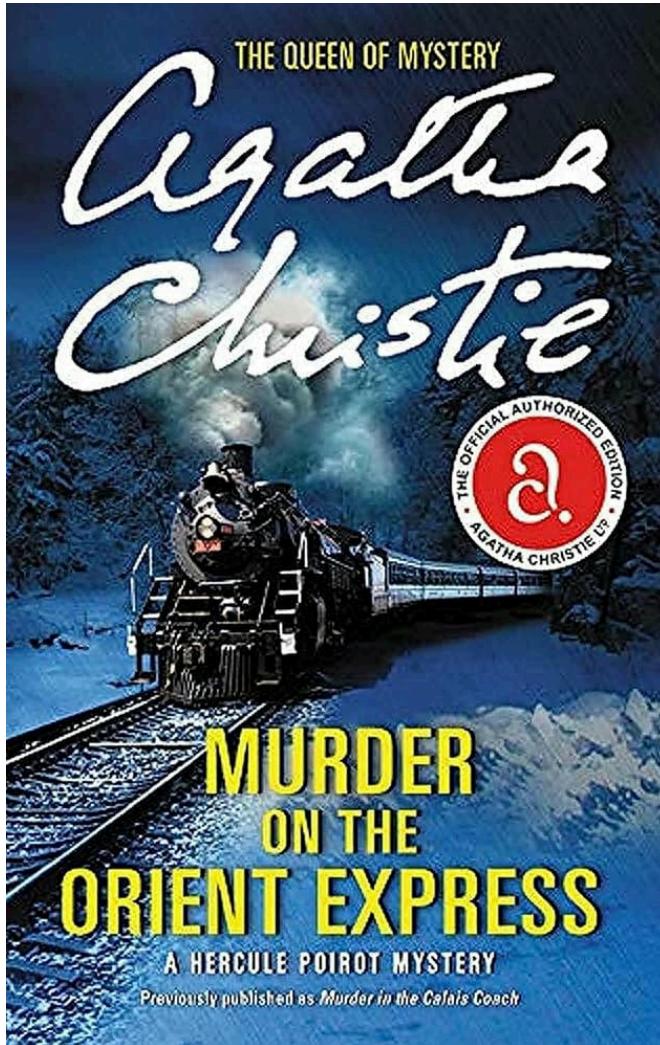
# Unsupervised Training



To complete the sentence, the model must understand:

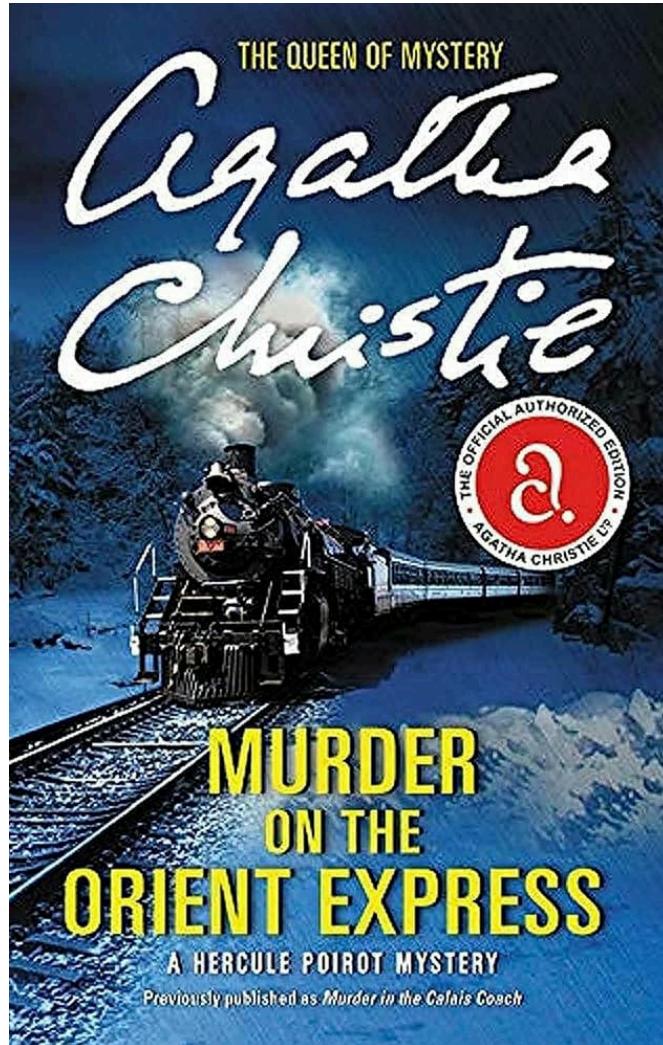
- What a murder is
- What it means to be alive
- Emotions like anger, jealousy, betrayal, love
- Personalities of each character
- Why a human would murder another human
- How humans react to emotions
- How to tell if someone lies

# Unsupervised Training



To predict the murderer, the model must understand so much about humans and our society

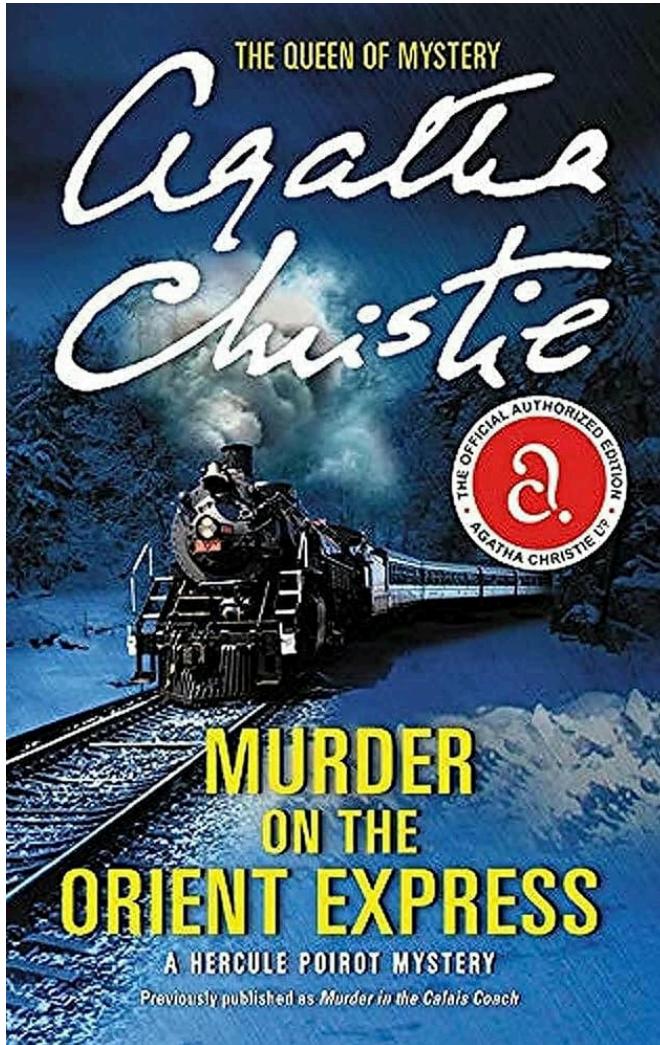
# Unsupervised Training



To predict the murderer, the model must understand so much about humans and our society

The Books3 dataset contains 200,000 books

# Unsupervised Training



To predict the murderer, the model must understand so much about humans and our society

The Books3 dataset contains 200,000 books

We train the model to predict the ending of all these books

# Unsupervised Training

But what about models like:

# Unsupervised Training

But what about models like:

- ChatGPT

# Unsupervised Training

But what about models like:

- ChatGPT
- DinoV2

# Unsupervised Training

But what about models like:

- ChatGPT
- DinoV2

We train these differently

# Unsupervised Training

We often follow a two-step process:

# Unsupervised Training

We often follow a two-step process:

- Learn the structure of the world (unsupervised learning)

# Unsupervised Training

We often follow a two-step process:

- Learn the structure of the world (unsupervised learning)
- Learn to be helpful to humans (reinforcement learning)

# Unsupervised Training

We often follow a two-step process:

- Learn the structure of the world (unsupervised learning)
- Learn to be helpful to humans (reinforcement learning)

# Unsupervised Training

Reading a book next token completion

# Unsupervised Training

World modeling

But

Generative pretraining

RL

# Unsupervised Training

Predict the future

# World Models

---

# World Models

What if transformers could interact with the world?

# World Models

# Closing Remarks

---

# Closing Remarks

This is the last in-person lecture

# Closing Remarks

This is the last in-person lecture

I will record a video on reinforcement learning next week

# Closing Remarks

This is the last in-person lecture

I will record a video on reinforcement learning next week

I will be here from 7:00PM on December 2 for questions/discussin on reinforcement learning

# Closing Remarks

In this course, we started from Gauss in 1795

# Closing Remarks

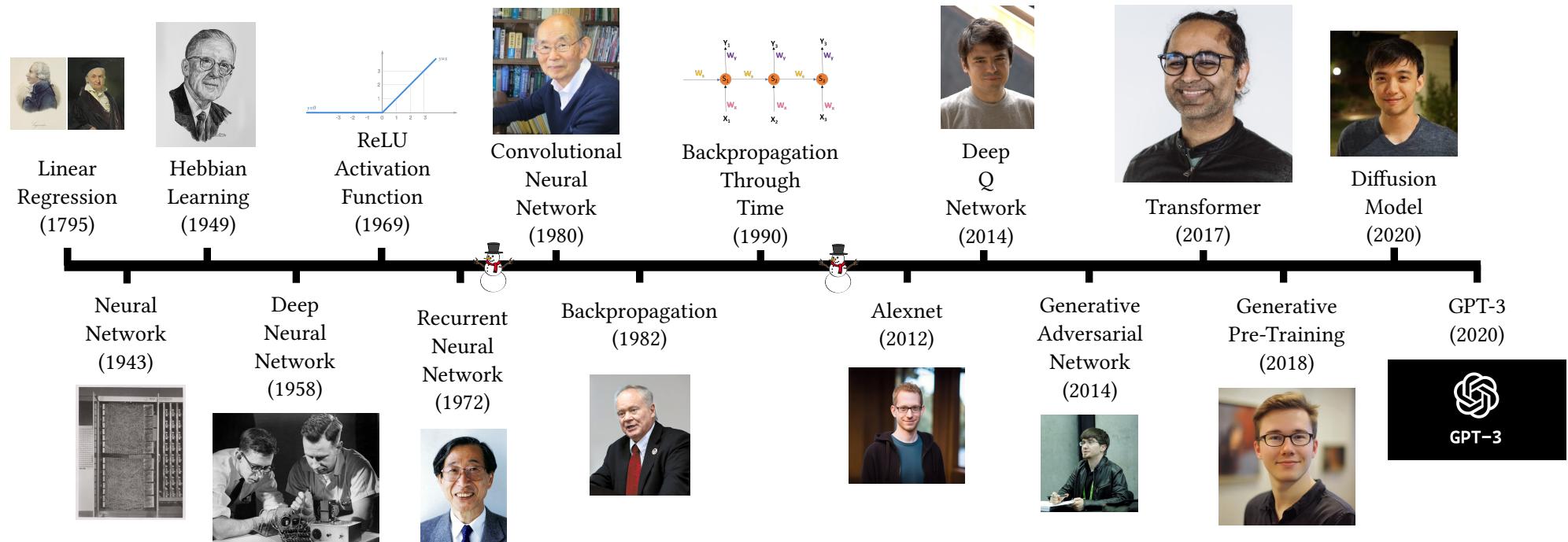
In this course, we started from Gauss in 1795

We built up concepts until we reached the modern age

# Closing Remarks

In this course, we started from Gauss in 1795

We built up concepts until we reached the modern age



# Closing Remarks

We learned about:

# Closing Remarks

We learned about:

- Linear regression

# Closing Remarks

We learned about:

- Linear regression
- Nonlinear/polynomial regression

# Closing Remarks

We learned about:

- Linear regression
- Nonlinear/polynomial regression
- Biological neurons

# Closing Remarks

We learned about:

- Linear regression
- Nonlinear/polynomial regression
- Biological neurons
- Artificial neurons

# Closing Remarks

We learned about:

- Linear regression
- Nonlinear/polynomial regression
- Biological neurons
- Artificial neurons
- Perceptrons

# Closing Remarks

We learned about:

- Linear regression
- Nonlinear/polynomial regression
- Biological neurons
- Artificial neurons
- Perceptrons
- Backpropagation

# Closing Remarks

We learned about:

- Linear regression
- Nonlinear/polynomial regression
- Biological neurons
- Artificial neurons
- Perceptrons
- Backpropagation
- Gradient descent

# Closing Remarks

We learned about:

- Linear regression
- Nonlinear/polynomial regression
- Biological neurons
- Artificial neurons
- Perceptrons
- Backpropagation
- Gradient descent
- Classification

# Closing Remarks

We learned about:

- Linear regression
- Nonlinear/polynomial regression
- Biological neurons
- Artificial neurons
- Perceptrons
- Backpropagation
- Gradient descent
- Classification
- Parameter initialization

# Closing Remarks

We learned about:

- Linear regression
- Nonlinear/polynomial regression
- Biological neurons
- Artificial neurons
- Perceptrons
- Backpropagation
- Gradient descent
- Classification
- Parameter initialization
- Many activation functions

# Closing Remarks

We learned about:

- Linear regression
- Nonlinear/polynomial regression
- Biological neurons
- Artificial neurons
- Perceptrons
- Backpropagation
- Gradient descent
- Classification
- Parameter initialization
- Many activation functions
- Stochastic gradient descent

# Closing Remarks

We learned about:

- Linear regression
- Nonlinear/polynomial regression
- Biological neurons
- Artificial neurons
- Perceptrons
- Backpropagation
- Gradient descent
- Classification
- Parameter initialization
- Many activation functions
- Stochastic gradient descent
- RMSProp and Adam

# Closing Remarks

We learned about:

- Linear regression
- Nonlinear/polynomial regression
- Biological neurons
- Artificial neurons
- Perceptrons
- Backpropagation
- Gradient descent
- Classification
- Parameter initialization
- Many activation functions
- Stochastic gradient descent
- RMSProp and Adam
- Convolutional neural networks

# Closing Remarks

We learned about:

- Linear regression
- Nonlinear/polynomial regression
- Biological neurons
- Artificial neurons
- Perceptrons
- Backpropagation
- Gradient descent
- Classification
- Parameter initialization
- Many activation functions
- Stochastic gradient descent
- RMSProp and Adam
- Convolutional neural networks
- Composite memory

# Closing Remarks

We learned about:

- Linear regression
- Nonlinear/polynomial regression
- Biological neurons
- Artificial neurons
- Perceptrons
- Backpropagation
- Gradient descent
- Classification
- Parameter initialization
- Many activation functions
- Stochastic gradient descent
- RMSProp and Adam
- Convolutional neural networks
- Composite memory
- Recurrent neural networks

# Closing Remarks

We learned about:

- Linear regression
- Nonlinear/polynomial regression
- Biological neurons
- Artificial neurons
- Perceptrons
- Backpropagation
- Gradient descent
- Classification
- Parameter initialization
- Many activation functions
- Stochastic gradient descent
- RMSProp and Adam
- Convolutional neural networks
- Composite memory
- Recurrent neural networks
- Autoencoders

# Closing Remarks

We learned about:

- Linear regression
- Nonlinear/polynomial regression
- Biological neurons
- Artificial neurons
- Perceptrons
- Backpropagation
- Gradient descent
- Classification
- Parameter initialization
- Many activation functions
- Stochastic gradient descent
- RMSProp and Adam
- Convolutional neural networks
- Composite memory
- Recurrent neural networks
- Autoencoders
- Variational autoencoders

# Closing Remarks

We learned about:

- Linear regression
- Nonlinear/polynomial regression
- Biological neurons
- Artificial neurons
- Perceptrons
- Backpropagation
- Gradient descent
- Classification
- Parameter initialization
- Many activation functions
- Stochastic gradient descent
- RMSProp and Adam
- Convolutional neural networks
- Composite memory
- Recurrent neural networks
- Autoencoders
- Variational autoencoders
- Graph neural networks

# Closing Remarks

We learned about:

- Linear regression
- Nonlinear/polynomial regression
- Biological neurons
- Artificial neurons
- Perceptrons
- Backpropagation
- Gradient descent
- Classification
- Parameter initialization
- Many activation functions
- Stochastic gradient descent
- RMSProp and Adam
- Convolutional neural networks
- Composite memory
- Recurrent neural networks
- Autoencoders
- Variational autoencoders
- Graph neural networks
- Attention and transformers

# Closing Remarks

Now, you have the tools to continue studying deep learning

# Closing Remarks

Now, you have the tools to continue studying deep learning

You also have the ability to train neural networks and solve real problems

# Closing Remarks

In the first lecture, I asked everyone in this class for something

# Closing Remarks

In the first lecture, I asked everyone in this class for something

**Question:** Do you remember what it was?

# Closing Remarks

You are now experts at deep learning

# Closing Remarks

You are now experts at deep learning

Deep learning is a powerful tool

# Closing Remarks

You are now experts at deep learning

Deep learning is a powerful tool

Like all powerful tools, deep learning can be used for good or evil

# Closing Remarks

You are now experts at deep learning

Deep learning is a powerful tool

Like all powerful tools, deep learning can be used for good or evil

- COVID-19 vaccine

# Closing Remarks

You are now experts at deep learning

Deep learning is a powerful tool

Like all powerful tools, deep learning can be used for good or evil

- COVID-19 vaccine
- Creating art

# Closing Remarks

You are now experts at deep learning

Deep learning is a powerful tool

Like all powerful tools, deep learning can be used for good or evil

- COVID-19 vaccine
- Creating art
- Autonomous driving

# Closing Remarks

You are now experts at deep learning

Deep learning is a powerful tool

Like all powerful tools, deep learning can be used for good or evil

- COVID-19 vaccine
- Creating art
- Autonomous driving
- Making DeepFakes

# Closing Remarks

You are now experts at deep learning

Deep learning is a powerful tool

Like all powerful tools, deep learning can be used for good or evil

- COVID-19 vaccine
- Creating art
- Autonomous driving
- Making DeepFakes
- Weapon guidance systems

# Closing Remarks

You are now experts at deep learning

Deep learning is a powerful tool

Like all powerful tools, deep learning can be used for good or evil

- COVID-19 vaccine
- Creating art
- Autonomous driving
- Making DeepFakes
- Weapon guidance systems
- Discrimination

**Before training a model, think about whether it is good or bad for the world**

# Course Evaluation

---

# Course Evaluation

Department instructed me to ask you for course feedback

# Course Evaluation

Department instructed me to ask you for course feedback

We take this feedback seriously

# Course Evaluation

Department instructed me to ask you for course feedback

We take this feedback seriously

Your feedback will impact future courses (and my job)

# Course Evaluation

Department instructed me to ask you for course feedback

We take this feedback seriously

Your feedback will impact future courses (and my job)

If you like the course, please say it!

# Course Evaluation

Department instructed me to ask you for course feedback

We take this feedback seriously

Your feedback will impact future courses (and my job)

If you like the course, please say it!

Your likes/dislikes will filter into your future courses

# Course Evaluation

Department instructed me to ask you for course feedback

We take this feedback seriously

Your feedback will impact future courses (and my job)

If you like the course, please say it!

Your likes/dislikes will filter into your future courses

If you are not comfortable writing English, write Chinese

# Course Evaluation

Department instructed me to ask you for course feedback

We take this feedback seriously

Your feedback will impact future courses (and my job)

If you like the course, please say it!

Your likes/dislikes will filter into your future courses

If you are not comfortable writing English, write Chinese

# Course Evaluation

I must leave the room to let you fill out this form

# Course Evaluation

I must leave the room to let you fill out this form

Please scan the QR code and complete the survey

# Course Evaluation

I must leave the room to let you fill out this form

Please scan the QR code and complete the survey

Department has suggested 10 minutes

# Course Evaluation

I must leave the room to let you fill out this form

Please scan the QR code and complete the survey

Department has suggested 10 minutes

<https://isw.um.edu.mo/siaweb>

# Course Evaluation

Research data labeling and collection

If you participated, come up