Transformers

CISC 7026 - Introduction to Deep Learning

Steven Morad

University of Macau

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

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

Review

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

We started with composite memory

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

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

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

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

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

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 γ

Review

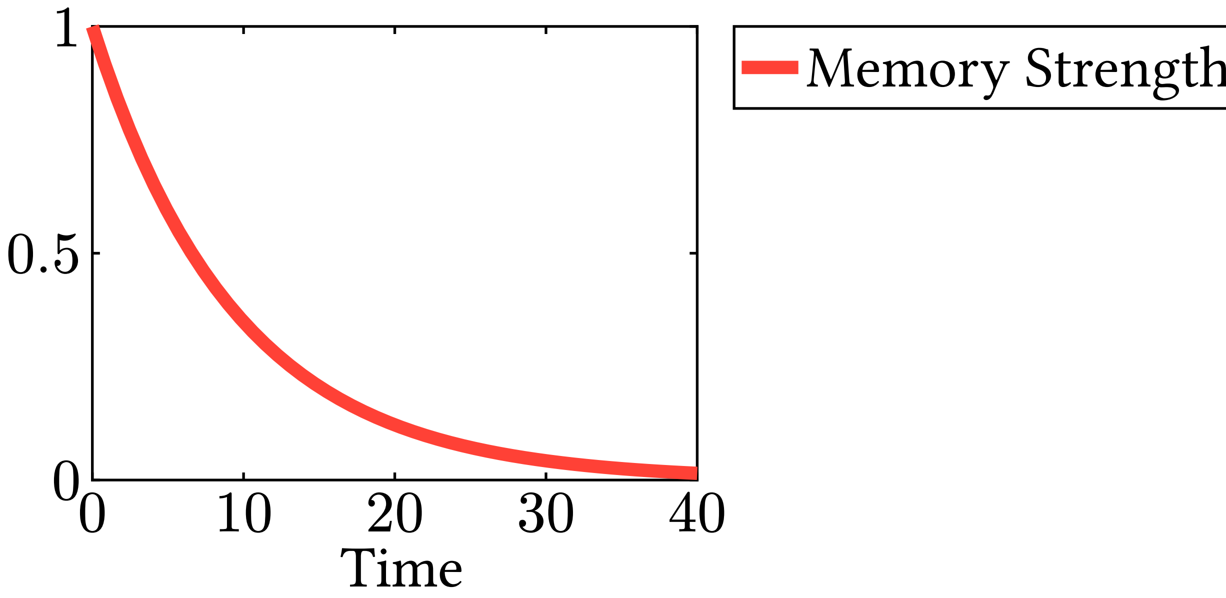
So we introduced a forgetting term γ

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

Review

So we introduced a forgetting term γ

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



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

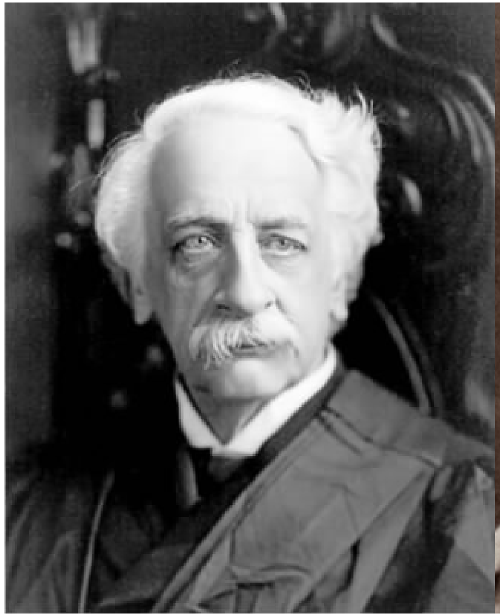
Review

We went to a party and the forgetting seemed ok

10 PM 11 PM 12 AM

Review

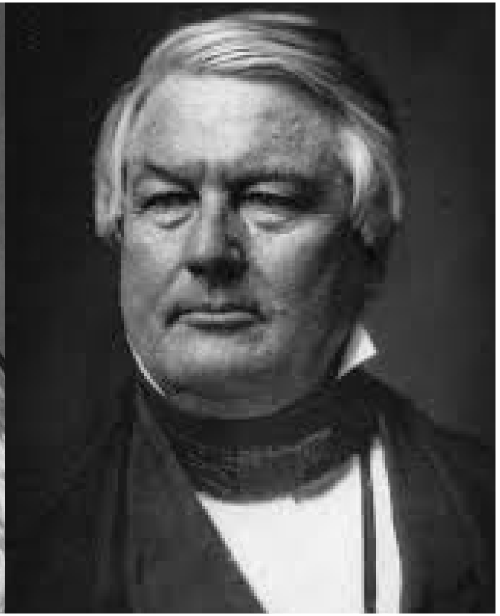
We went to a party and the forgetting seemed ok



10 PM

11 PM

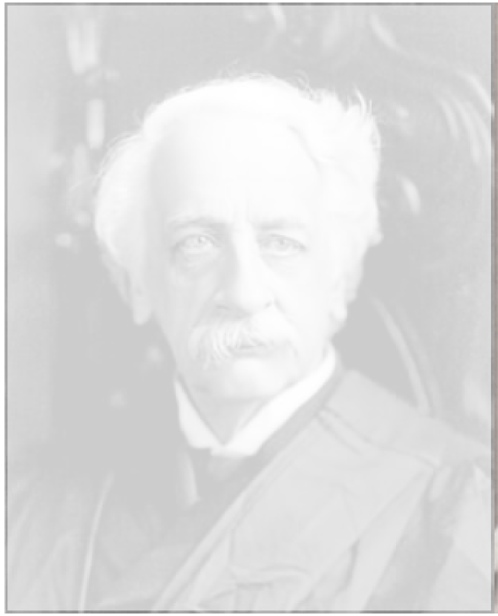
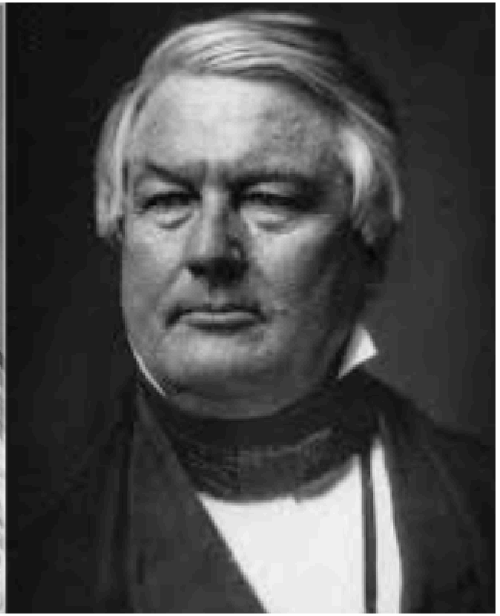
12 AM



1 AM

Review

Review



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

Review

$\gamma^3 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1$ $\gamma^2 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2$

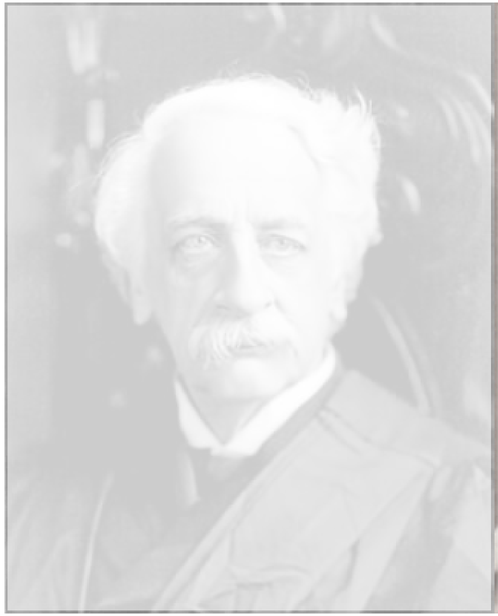
Review

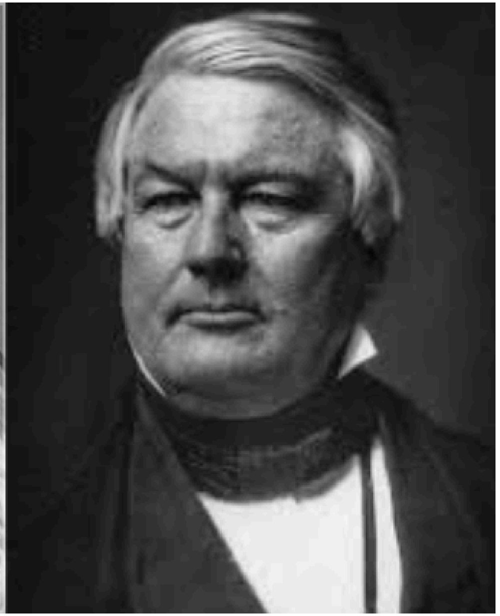
$\gamma^3 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1$ $\gamma^2 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2$ $\gamma^1 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3$

Review

$\gamma^3 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1$ $\gamma^2 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2$ $\gamma^1 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3$ $\gamma^0 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_4$

Review



$\gamma^3 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1$ $\gamma^2 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2$ $\gamma^1 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3$ $\gamma^0 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_4$

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

$\gamma^4 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1$ $\gamma^3 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2$ $\gamma^2 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3$ $\gamma^1 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_4$ $\gamma^0 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_5$

Review

But we encountered problems when Taylor Swift arrived at the party

$\gamma^4 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1$ $\quad$ $\gamma^3 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2$ $\quad$ $\gamma^2 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3$ $\quad$ $\gamma^1 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_4$ $\quad$ $\gamma^0 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_5$

Review

$\gamma^4 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1$ $\gamma^3 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2$ $\gamma^2 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3$ $\gamma^1 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_4$ $\gamma^0 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_5$

Review

$$\gamma^4 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1 \qquad \gamma^3 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2 \qquad \gamma^2 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3 \qquad \gamma^1 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_4 \qquad \gamma^0 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_5$$

With our current model, we forget Taylor Swift!

Review

$\gamma^4 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1$ $\gamma^3 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2$ $\gamma^2 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3$ $\gamma^1 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_4$ $\gamma^0 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_5$

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

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

Going Deeper

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

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

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

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

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

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

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

$$\boldsymbol{y} = f_k(...f_2(f_1(\boldsymbol{x}, \boldsymbol{\theta}_1), \boldsymbol{\theta_2}), ..., \boldsymbol{\theta}_k)$$

Going Deeper

$$\boldsymbol{y} = f_k(...f_2(f_1(\boldsymbol{x}, \boldsymbol{\theta}_1), \boldsymbol{\theta_2}), ..., \boldsymbol{\theta}_k)$$

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

$$\boldsymbol{y} = f_k(...f_2(f_1(\boldsymbol{x}, \boldsymbol{\theta}_1), \boldsymbol{\theta_2}), ..., \boldsymbol{\theta}_k)$$

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

$$\boldsymbol{x} = f_k(...f_2(f_1(\boldsymbol{x}, \boldsymbol{\theta}_1), \boldsymbol{\theta_2}), ..., \boldsymbol{\theta}_k)$$

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

Going Deeper

$$\boldsymbol{y} = f_k(...f_2(f_1(\boldsymbol{x}, \boldsymbol{\theta}_1), \boldsymbol{\theta}_{\mathbf{2}}), ..., \boldsymbol{\theta}_k)$$

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

$$\boldsymbol{x} = f_k(...f_2(f_1(\boldsymbol{x}, \boldsymbol{\theta}_1), \boldsymbol{\theta}_{\mathbf{2}}), ..., \boldsymbol{\theta}_k)$$

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

Question: Do you agree or disagree with the claim?

Going Deeper

$$\boldsymbol{y} = f_k(...f_2(f_1(\boldsymbol{x}, \boldsymbol{\theta}_1), \boldsymbol{\theta_2}), ..., \boldsymbol{\theta}_k)$$

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

$$\boldsymbol{x} = f_k(...f_2(f_1(\boldsymbol{x}, \boldsymbol{\theta}_1), \boldsymbol{\theta_2}), ..., \boldsymbol{\theta}_k)$$

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/1qVIbQKpTuBYIa7FvC4IH-kJq-E0jmc0d#scrollTo=bg74S-AvbmJz

Going Deeper

$$\boldsymbol{x} = f_k(...f_2(f_1(\boldsymbol{x}, \boldsymbol{\theta}_1), \boldsymbol{\theta_2}), ..., \boldsymbol{\theta}_k)$$

Going Deeper

$$\boldsymbol{x} = f_k(...f_2(f_1(\boldsymbol{x}, \boldsymbol{\theta}_1), \boldsymbol{\theta_2}), ..., \boldsymbol{\theta}_k)$$

Very deep networks struggle to learn the identity function

Going Deeper

$$\boldsymbol{x} = f_k(...f_2(f_1(\boldsymbol{x}, \boldsymbol{\theta}_1), \boldsymbol{\theta}_{\mathbf{2}}), ..., \boldsymbol{\theta}_k)$$

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

$$\boldsymbol{z}_1 = f_1(\boldsymbol{x}, \boldsymbol{\theta}_1)$$

$$\boldsymbol{z}_2 = f_2\left(\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{z}_1 \end{bmatrix}, \boldsymbol{\theta}_2\right)$$

$$\vdots$$

$$\boldsymbol{z}_k = f_k\left(\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{z}_1 \\ \vdots \\ \boldsymbol{z}_{k-1} \end{bmatrix}, \boldsymbol{\theta}_k\right)$$

Going Deeper

$$
\begin{aligned}
\boldsymbol{z}_1 &= f_1(\boldsymbol{x}, \boldsymbol{\theta}_1) \\
\boldsymbol{z}_2 &= f_2\left(\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{z}_1 \end{bmatrix}, \boldsymbol{\theta}_2\right) \\
&\vdots \\
\boldsymbol{z}_k &= f_k\left(\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{z}_1 \\ \vdots \\ \boldsymbol{z}_{k-1} \end{bmatrix}, \boldsymbol{\theta}_k\right)
\end{aligned}
$$

Going Deeper

$$\begin{aligned}
\boldsymbol{z}_1 &= f_1(\boldsymbol{x}, \boldsymbol{\theta}_1) \\
\boldsymbol{z}_2 &= f_2\left(\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{z}_1 \end{bmatrix}, \boldsymbol{\theta}_2\right) \\
&\vdots \\
\boldsymbol{z}_k &= f_k\left(\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{z}_1 \\ \vdots \\ \boldsymbol{z}_{k-1} \end{bmatrix}, \boldsymbol{\theta}_k\right)
\end{aligned}$$

Question: Any issues with the DenseNet approach?

Going Deeper

$$
\begin{aligned}
\boldsymbol{z}_1 &= f_1(\boldsymbol{x}, \boldsymbol{\theta}_1) \\
\boldsymbol{z}_2 &= f_2\left(\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{z}_1 \end{bmatrix}, \boldsymbol{\theta}_2\right) \\
&\vdots \\
\boldsymbol{z}_k &= f_k\left(\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{z}_1 \\ \vdots \\ \boldsymbol{z}_{k-1} \end{bmatrix}, \boldsymbol{\theta}_k\right)
\end{aligned}
$$

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

$$\begin{aligned}
\boldsymbol{z}_1 &= f_1(\boldsymbol{x}, \boldsymbol{\theta}_1) \\
\boldsymbol{z}_2 &= f_2(\boldsymbol{x}, \boldsymbol{\theta}_2) + \boldsymbol{x} \\
\boldsymbol{z}_3 &= f_2(\boldsymbol{x}, \boldsymbol{\theta}_3) + \boldsymbol{z}_2 \\
&\vdots \\
\boldsymbol{z}_k &= f_k(\boldsymbol{x}, \boldsymbol{\theta}_k) + \boldsymbol{z}_{k-1}
\end{aligned}$$

Going Deeper

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

$$\begin{aligned}
\boldsymbol{z}_1 &= f_1(\boldsymbol{x}, \boldsymbol{\theta}_1) \\
\boldsymbol{z}_2 &= f_2(\boldsymbol{x}, \boldsymbol{\theta}_2) + \boldsymbol{x} \\
\boldsymbol{z}_3 &= f_2(\boldsymbol{x}, \boldsymbol{\theta}_3) + \boldsymbol{z}_2 \\
&\vdots \\
\boldsymbol{z}_k &= f_k(\boldsymbol{x}, \boldsymbol{\theta}_k) + \boldsymbol{z}_{k-1}
\end{aligned}$$

This allows information to flow around the layers

Going Deeper

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

$$
\begin{aligned}
\boldsymbol{z}_1 &= f_1(\boldsymbol{x}, \boldsymbol{\theta}_1) \\
\boldsymbol{z}_2 &= f_2(\boldsymbol{x}, \boldsymbol{\theta}_2) + \boldsymbol{x} \\
\boldsymbol{z}_3 &= f_2(\boldsymbol{x}, \boldsymbol{\theta}_3) + \boldsymbol{z}_2 \\
&\vdots \\
\boldsymbol{z}_k &= f_k(\boldsymbol{x}, \boldsymbol{\theta}_k) + \boldsymbol{z}_{k-1}
\end{aligned}
$$

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

$$
\begin{aligned}
\boldsymbol{z}_1 &= f_1(\boldsymbol{x}, \boldsymbol{\theta}_1) \\
\boldsymbol{z}_2 &= f_2(\boldsymbol{x}, \boldsymbol{\theta}_2) + \boldsymbol{x} \\
\boldsymbol{z}_3 &= f_2(\boldsymbol{x}, \boldsymbol{\theta}_3) + \boldsymbol{z}_2 \\
&\vdots \\
\boldsymbol{z}_k &= f_k(\boldsymbol{x}, \boldsymbol{\theta}_k) + \boldsymbol{z}_{k-1}
\end{aligned}
$$

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 $f(\boldsymbol{x}) + \boldsymbol{x}$ a **residual connection**

Going Deeper

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

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

Going Deeper

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

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

For example, for an identity function we can easily learn

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

Going Deeper

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

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

For example, for an identity function we can easily learn

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

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

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

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

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

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

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

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

Going Deeper

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

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

What if we add another layer with the same d_x and θ?

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

Going Deeper

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

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

What if we add another layer with the same d_x and θ?

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

Question: What is the problem?

Going Deeper

Let us look at the gradient

Going Deeper

Let us look at the gradient

$$\nabla_{\boldsymbol{\theta}_1} f_2(f_1(\boldsymbol{x}, \boldsymbol{\theta}_1), \boldsymbol{\theta}_2) =$$

Going Deeper

Let us look at the gradient

$$\nabla_{\boldsymbol{\theta}_1} f_2(f_1(\boldsymbol{x}, \boldsymbol{\theta}_1), \boldsymbol{\theta}_2) = \nabla_{\boldsymbol{\theta}_1}[f_2](f_1(\boldsymbol{x}, \boldsymbol{\theta}_1)) \cdot \nabla_{\boldsymbol{\theta}_1}[f_1](\boldsymbol{x}, \boldsymbol{\theta}_1)$$

Going Deeper

Let us look at the gradient

$$\nabla_{\boldsymbol{\theta}_1} f_2(f_1(\boldsymbol{x}, \boldsymbol{\theta}_1), \boldsymbol{\theta}_2) = \nabla_{\boldsymbol{\theta}_1}[f_2](f_1(\boldsymbol{x}, \boldsymbol{\theta}_1)) \cdot \nabla_{\boldsymbol{\theta}_1}[f_1](\boldsymbol{x}, \boldsymbol{\theta}_1)$$
$$\approx 100 \cdot 10$$

Can cause exploding or vanishing gradient

Going Deeper

Let us look at the gradient

$$\nabla_{\boldsymbol{\theta}_1} f_2(f_1(\boldsymbol{x}, \boldsymbol{\theta}_1), \boldsymbol{\theta}_2) = \nabla_{\boldsymbol{\theta}_1}[f_2](f_1(\boldsymbol{x}, \boldsymbol{\theta}_1)) \cdot \nabla_{\boldsymbol{\theta}_1}[f_1](\boldsymbol{x}, \boldsymbol{\theta}_1)$$
$$\approx 100 \cdot 10$$

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

$$\nabla_{\boldsymbol{\theta}_1} f_2(f_1(\boldsymbol{x}, \boldsymbol{\theta}_1), \boldsymbol{\theta}_2) = \nabla_{\boldsymbol{\theta}_1}[f_2](f_1(\boldsymbol{x}, \boldsymbol{\theta}_1)) \cdot \nabla_{\boldsymbol{\theta}_1}[f_1](\boldsymbol{x}, \boldsymbol{\theta}_1)$$
$$\approx 100 \cdot 10$$

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

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

$$f(\boldsymbol{x}, \boldsymbol{\theta}) - \mu$$

Question: What does this do?

Going Deeper

We can use **layer normalization**

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

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

$$f(\boldsymbol{x}, \boldsymbol{\theta}) - \mu$$

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

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

$$f(\boldsymbol{x}, \boldsymbol{\theta}) - \mu$$

Question: What does this do?

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

Going Deeper

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

Then, layer normalization **rescales** the outputs

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

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

Now, the output of the layer is normally distributed

Going Deeper

If the output is normally distrbuted:

Going Deeper

If the output is normally distrbuted:

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

Going Deeper

If the output is normally distrbuted:

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

Going Deeper

If the output is normally distrbuted:

- 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 distrbuted:

- 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

Going Deeper

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

Going Deeper

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

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

Going Deeper

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

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

Going Deeper

Layer Norm

Initialization and L2 normalization keep the weights small

But the outputs of each layer can still be large or small

Chain rule example

Transformers

Transformers

Now we have everything we need to implement a transformer

Positional Encoding

Text Transformers

Image Transformers

Unsupervised Training

Unsupervised Training

Predict the future

World Models

World Models

What if transformers could interact with 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)

Be specific on what you like and do not like

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)

Be specific on what you like and do not like

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)

Be specific on what you like and do not like

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)

Be specific on what you like and do not like

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