Attention Is All You Need:

Deriving the Seminal Transformer Architecture from First Principles

J. Setpal

September 3, 2024

MACHINE LEARNING @ PURDUE

Glossary

Some context-relevant terms:

a. **Neuron:** The unit of a nueral network $y = \sigma(XW)$
b. **Logit:** Pre-activation scores for the *final layer*.
c. **Dot / Inner Product:** $\langle a, b\rangle = \sum_{i=1}^{N} a_i b_i$
d. **Matrix Multiplication:** For $X \in \mathbb{R}^{a\times b},\ Y \in \mathbb{R}^{b\times c}$, $Z = XY \in \mathbb{R}^{a\times c}$ $s.t.$ $z_{ij} = \sum_{k=1}^{b} a_{ik} b_{kj}$
e. **Activation:** Non-linear function over the output of matrix multiplication.
f. **Gradient:** $\nabla_w \mathcal{L}(w)$, derivative of $\mathcal{L} : \mathbb{R}^a \rightarrow \mathbb{R}$
g. **Latent Vector:** $h^{(\ell)}$, intermediary output from within a neural network.
h. **Embedding:** A look-up table that translates categorical values (words) to vectors.

Some Intuitive Insights

Q: Differentiate between the meanings of each underlined word below:

a. I had a picnic by the river <u>bank</u> yesterday.

b. I am NOT going to rob a <u>bank</u> tomorrow at 8:30am ET.

Some Intuitive Insights

Q: Differentiate between the meanings of each underlined word below:

a. I had a picnic by the river bank yesterday.

b. I am NOT going to rob a bank tomorrow at 8:30am ET.

Insight: Context is important.

Some Intuitive Insights

Q: Differentiate between the meanings of each underlined word below:

a. I had a picnic by the river <u>bank</u> yesterday.

b. I am NOT going to rob a <u>bank</u> tomorrow at 8:30am ET.

Insight: Context is important.

n-gram models use the following (Markov) assumption:

$$p\left(x_t \middle| \{x_i\}_{i=1}^{t-1}; \theta\right) \approx p\left(x_t | x_{t-1}; \theta\right) \tag{1}$$

Some Intuitive Insights

Q: Differentiate between the meanings of each underlined word below:

a. I had a picnic by the river bank yesterday.

b. I am NOT going to rob a bank tomorrow at 8:30am ET.

Insight: Context is important.

n-gram models use the following (Markov) assumption:

$$p\left(x_t \middle| \{x_i\}_{i=1}^{t-1};\theta\right) \approx p\left(x_t | x_{t-1};\theta\right) \tag{1}$$

This is **incorrect.**

Some Intuitive Insights

Q: Differentiate between the meanings of each underlined word below:

a. I had a picnic by the river bank yesterday.

b. I am NOT going to rob a bank tomorrow at 8:30am ET.

Insight: Context is important.

n-gram models use the following (Markov) assumption:

$$p\left(x_t \middle| \{x_i\}_{i=1}^{t-1}; \theta\right) \approx p\left(x_t | x_{t-1}; \theta\right) \tag{1}$$

This is **incorrect.** Context is important!

How Expensive is Context?

Q: So, why do *n*-gram models use the Markov assumption?

How Expensive is Context?

Q: So, why do *n*-gram models use the Markov assumption?
A: Efficiency.

How Expensive is Context?

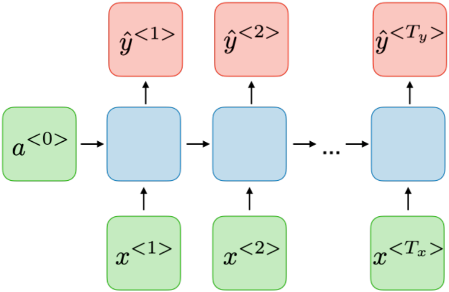
Q: So, why do *n*-gram models use the Markov assumption?
A: Efficiency.

Q: Can we do better?

How Expensive is Context?

Q: So, why do *n*-gram models use the Markov assumption?
A: Efficiency.

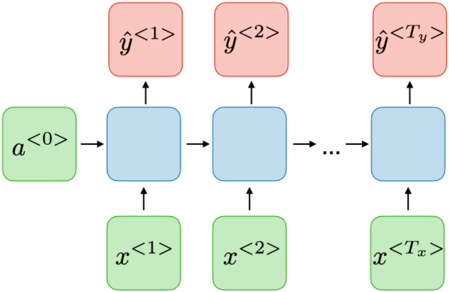
Q: Can we do better?
A: Yes. Enter, Recurrent Neural Networks (RNNs):



How Expensive is Context?

Q: So, why do *n*-gram models use the Markov assumption?
A: Efficiency.

Q: Can we do better?
A: Yes. Enter, Recurrent Neural Networks (RNNs):

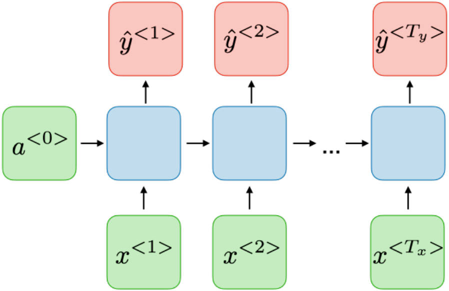


Here, a hidden representation $a^{\langle \ell \rangle}$ is propogated, encoding relevant context.

How Expensive is Context?

Q: So, why do *n*-gram models use the Markov assumption?
A: Efficiency.

Q: Can we do better?
A: Yes. Enter, Recurrent Neural Networks (RNNs):



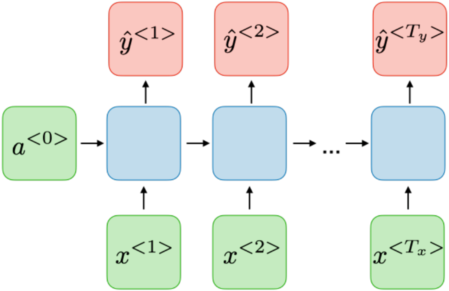
Here, a hidden representation $a^{\langle \ell \rangle}$ is propogated, encoding relevant context.

Caveat: $a^{\langle \ell \rangle} \in \mathbb{R}^d$, with *fixed* d.

How Expensive is Context?

Q: So, why do *n*-gram models use the Markov assumption?
A: Efficiency.

Q: Can we do better?
A: Yes. Enter, Recurrent Neural Networks (RNNs):



Here, a hidden representation $a^{\langle \ell \rangle}$ is propogated, encoding relevant context.

Caveat: $a^{\langle \ell \rangle} \in \mathbb{R}^d$, with *fixed* d. This mitigates capability for long-term memory & recollection.

One way to improve this is by incorporating *attention* into RNNs.[1]

[1] https://distill.pub/2016/augmented-rnns/#attentional-interfaces

Now, Let's Pay Attention

One way to improve this is by incorporating *attention* into RNNs.[1]

Q: Well, what does it mean to "incorporate attention"?

[1] `https://distill.pub/2016/augmented-rnns/#attentional-interfaces`

Now, Let's Pay Attention

One way to improve this is by incorporating *attention* into RNNs.[1]

Q: Well, what does it mean to "incorporate attention"?
A: For latent vectors $\{\ell_i\}_{i=1}^N$:

$$a^{\langle \ell \rangle} = \sum_{i=1}^{N} w_i^{(\ell)} \ell_i \quad \text{where} \quad \sum_{i=1}^{N} w_i^{(\ell)} = 1, \; w_i^{(\ell)} \geq 0 \; \forall i, \ell \tag{2}$$

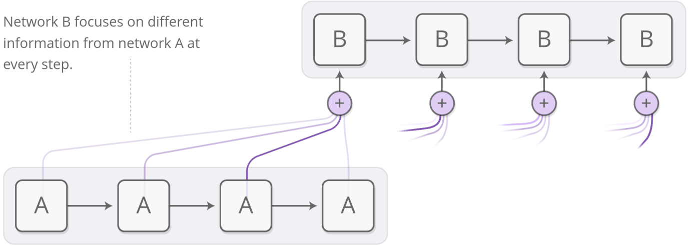
[1] https://distill.pub/2016/augmented-rnns/#attentional-interfaces

Now, Let's Pay Attention

One way to improve this is by incorporating *attention* into RNNs.[1]

Q: Well, what does it mean to "incorporate attention"?
A: For latent vectors $\{\ell_i\}_{i=1}^{N}$:

$$a^{\langle \ell \rangle} = \sum_{i=1}^{N} w_i^{(\ell)} \ell_i \quad \text{where} \quad \sum_{i=1}^{N} w_i^{(\ell)} = 1,\ w_i^{(\ell)} \geq 0\ \forall i, \ell \tag{2}$$



[1] https://distill.pub/2016/augmented-rnns/#attentional-interfaces

Self-Attention (1/2)

Self-Attention makes attention self-referential, by effectively creating a **trainable database**.

Self-Attention (1/2)

Self-Attention makes attention self-referential, by effectively creating a **trainable database**.

We query this database to extract important information from our input sequence. For $\{x_i\}_{i=1}^{t}$,

$$W_Q, W_K, W_V \in \mathbb{R}^{d_{in} \times d_{out}} \tag{3}$$

$$K = XW_K, Q = XW_Q, V = XW_V \tag{4}$$

(5)

(6)

Self-Attention (1/2)

Self-Attention makes attention self-referential, by effectively creating a **trainable database**.

We query this database to extract important information from our input sequence. For $\{x_i\}_{i=1}^{t}$,

$$W_Q, W_K, W_V \in \mathbb{R}^{d_{in} \times d_{out}} \tag{3}$$

$$K = XW_K, Q = XW_Q, V = XW_V \tag{4}$$

$$\alpha_i = \sigma_{softmax}\left(\frac{q_i k_i^T}{\sqrt{d_k}}\right) \tag{5}$$

(6)

Self-Attention (1/2)

Self-Attention makes attention self-referential, by effectively creating a **trainable database**.

We query this database to extract important information from our input sequence. For $\{x_i\}_{i=1}^{t}$,

$$W_Q, W_K, W_V \in \mathbb{R}^{d_{in} \times d_{out}} \tag{3}$$

$$K = XW_K, Q = XW_Q, V = XW_V \tag{4}$$

$$\alpha_i = \sigma_{softmax}\left(\frac{q_i k_i^T}{\sqrt{d_k}}\right) \tag{5}$$

$$h(x) = \sum_{i=1}^{t} \alpha_i v_i \tag{6}$$

Self-Attention (1/2)

Self-Attention makes attention self-referential, by effectively creating a **trainable database**.

We query this database to extract important information from our input sequence. For $\{x_i\}_{i=1}^{t}$,

$$W_Q, W_K, W_V \in \mathbb{R}^{d_{in} \times d_{out}} \tag{3}$$

$$K = XW_K, Q = XW_Q, V = XW_V \tag{4}$$

$$\alpha_i = \sigma_{softmax}\left(\frac{q_i k_i^T}{\sqrt{d_k}}\right) \tag{5}$$

$$h(x) = \sum_{i=1}^{t} \alpha_i v_i \tag{6}$$

Where q_i, k_i, v_i are each independently computed latent matrices.

$$\text{Self-Attention}(Q, K, V) = \left(\frac{QK^T}{\sqrt{d_{out}}}\right) V \tag{7}$$

Positional Encoding

A consequence of this setup, howeveer is that we are not considering the order of the tokens anymore. It is **permutation invariant**.

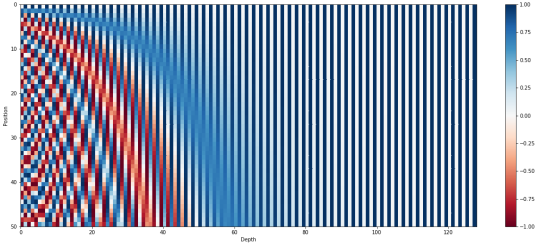
Positional Encoding

A consequence of this setup, howeveer is that we are not considering the order of the tokens anymore. It is **permutation invariant**.

To resolve this, we add positional encodings to each word embedding:

$$PE_{(pos,2i)} = \sin\left({}^{pos}\!/\!{}_{1E4^{2i/d_{model}}}\right) \tag{8}$$

$$PE_{(pos,2i+1)} = \cos\left({}^{pos}\!/\!{}_{1E4^{2i/d_{model}}}\right) \tag{9}$$



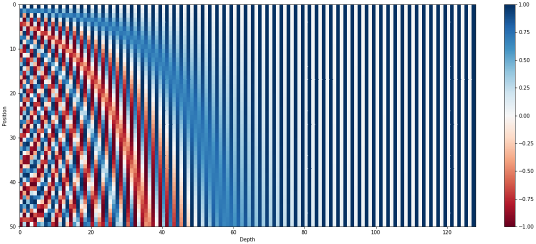
Positional Encoding

A consequence of this setup, howeveer is that we are not considering the order of the tokens anymore. It is **permutation invariant**.

To resolve this, we add positional encodings to each word embedding:

$$PE_{(pos,2i)} = \sin\left({}^{pos}/{}_{1E4^{2i/d_{model}}}\right) \tag{8}$$

$$PE_{(pos,2i+1)} = \cos\left({}^{pos}/{}_{1E4^{2i/d_{model}}}\right) \tag{9}$$



Modern-day references: **Rotary Positional Encodings (RoPE)**.

Multi-Headed Setting

Each block of self-attention constitutes it's own head.

Multi-Headed Setting

Each block of self-attention constitutes it's own head.

Intuitively, **multiple interpretations are desired from a given input**.
e.g: NER, Sentence-Structure Decomposition, POS tagging.

Multi-Headed Setting

Each block of self-attention constitutes it's own head.

Intuitively, **multiple interpretations are desired from a given input**.
e.g: NER, Sentence-Structure Decomposition, POS tagging.

Therefore, we instantiate *multiple* heads within each layer, and concatenate to construct a final output represetnation.

$$MHA = \text{Concat}(\{h_i\}_{i=1}^{H})W_O \tag{10}$$

Multi-Headed Setting

Each block of self-attention constitutes it's own head.

Intuitively, **multiple interpretations are desired from a given input**.
e.g: NER, Sentence-Structure Decomposition, POS tagging.

Therefore, we instantiate *multiple* heads within each layer, and concatenate to construct a final output represetnation.

$$MHA = \text{Concat}(\{h_i\}_{i=1}^{H})W_O \tag{10}$$

To keep computational costs consistent, we proportionally reduce the hidden dimension for each head:

$$W_{K,i}, W_{Q,i}, W_{V,i} \in \mathbb{R}^{d_{in} \times d_{out}/n_{head}} \quad \forall i \in \{1, \ldots, n_{head}\} \tag{11}$$

Multi-Headed Setting

Each block of self-attention constitutes it's own head.

Intuitively, **multiple interpretations are desired from a given input**.
e.g: NER, Sentence-Structure Decomposition, POS tagging.

Therefore, we instantiate *multiple* heads within each layer, and concatenate to construct a final output represetnation.

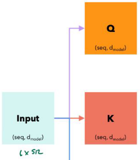
$$MHA = \text{Concat}(\{h_i\}_{i=1}^{H})W_O \tag{10}$$

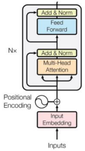
To keep computational costs consistent, we proportionally reduce the hidden dimension for each head:

$$W_{K,i}, W_{Q,i}, W_{V,i} \in \mathbb{R}^{d_{in} \times d_{out}/n_{head}} \quad \forall i \in \{1, \ldots, n_{head}\} \tag{11}$$

Modern-day references: **Flash Attention**, **Multi-Head Latent Attention**.

Self-Attention (2/2)



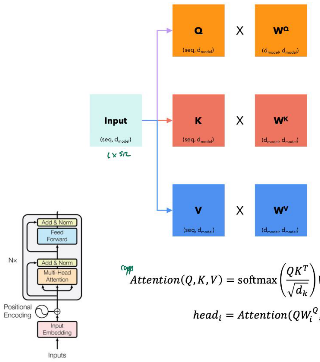


$$\mathit{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$\mathit{head}_i = \mathit{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

seq	= sequence length
d_{model}	= size of the embedding vector
h	= number of heads
$d_k = d_v$	= d_{model} / h

Self-Attention (2/2)



$$Attention(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

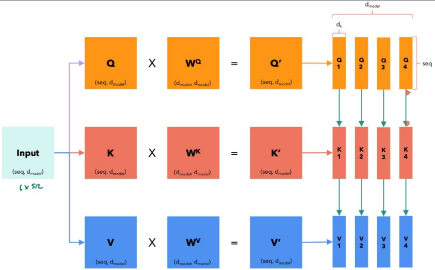
$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

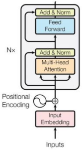
seq = sequence length

d_{model} = size of the embedding vector

h = number of heads

$d_k = d_v$ = d_{model} / h





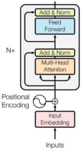
$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

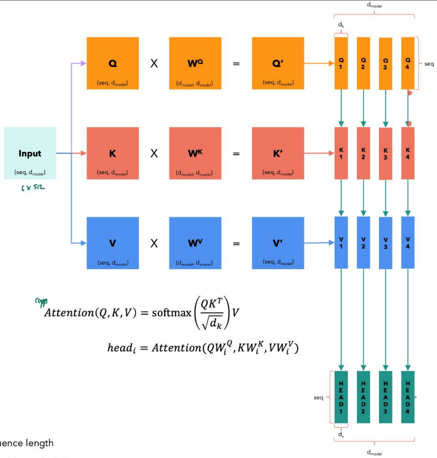
$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

s

seq	= sequence length
d_{model}	= size of the embedding vector
h	= number of heads
$d_k = d_v$	= d_{model} / h

Self-Attention (2/2)



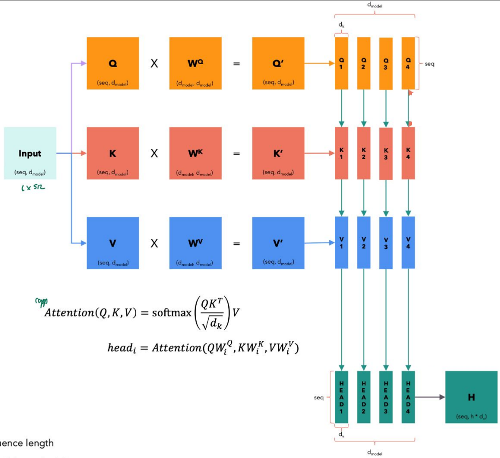


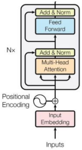
$$Attention(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

seq	= sequence length
d_{model}	= size of the embedding vector
h	= number of heads
$d_k = d_v$	= d_{model} / h

Self-Attention (2/2)



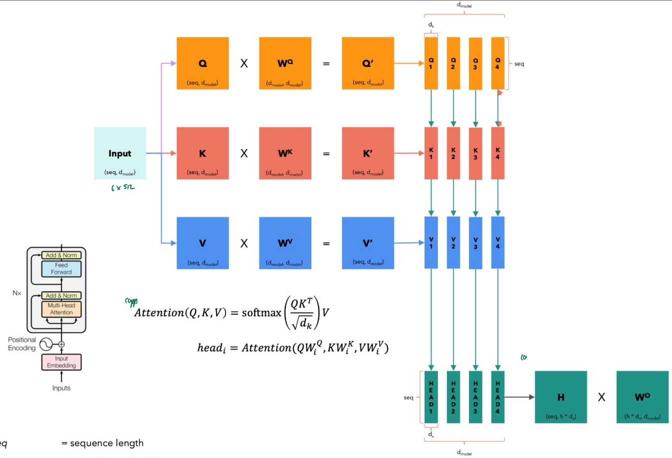


$$Attention(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

seq	= sequence length
d_{model}	= size of the embedding vector
h	= number of heads
$d_k = d_v$	= d_{model} / h

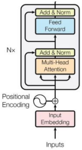
Self-Attention (2/2)

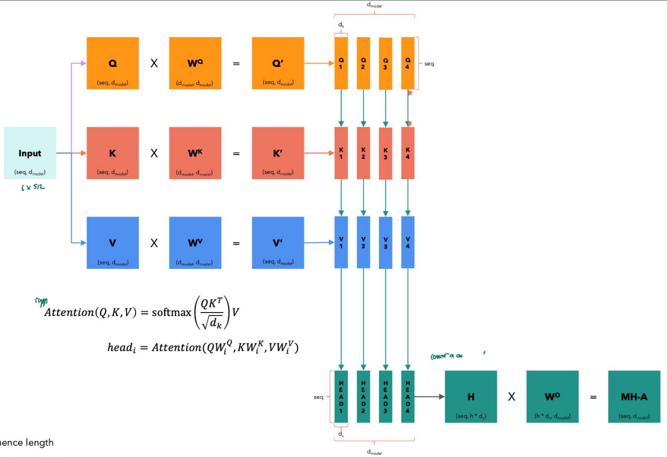


$$Attention(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

seq	= sequence length
d_{model}	= size of the embedding vector
h	= number of heads
$d_k = d_v$	= d_{model} / h



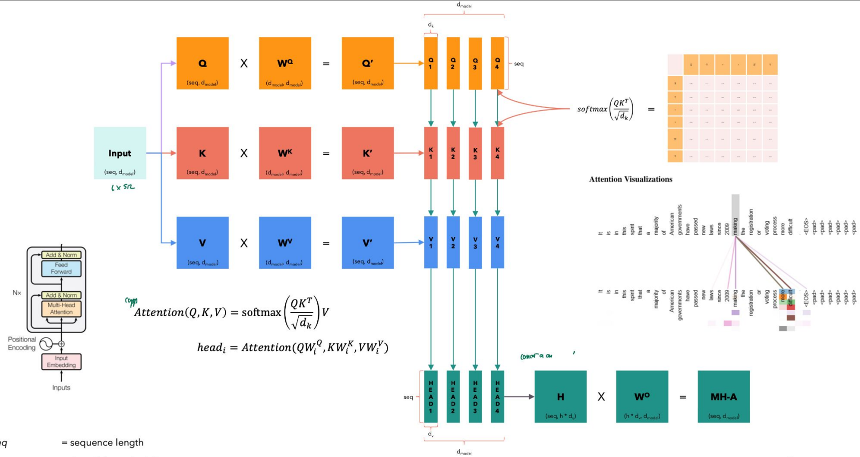


$$Attention(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

$$MultiHead(Q,K,V) = Concat(head_1 \ldots head_h)W^O$$

seq	= sequence length
d_{model}	= size of the embedding vector
h	= number of heads
$d_k = d_v$	$= d_{model} / h$



copy

$$Attention(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

concat on

$$MultiHead(Q,K,V) = Concat(head_1 \ldots head_h)W^O$$

seq	= sequence length
d_{model}	= size of the embedding vector
h	= number of heads
$d_k = d_v$	= d_{model} / h

Feedforward MLP

Using attention, we now have an updated representation of our input. We now need to perform operations on this contextualized input to make inferences about our actual objective.

Feedforward MLP

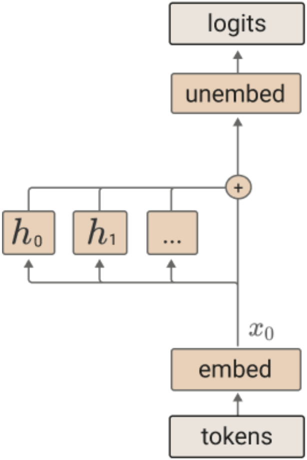
Using attention, we now have an updated representation of our input. We now need to perform operations on this contextualized input to make inferences about our actual objective.

For this, a two-layer MLP is instantiated that expands and consequently contracts the input dimension.

$$FFN(x) = \sigma_{relu}(xW_1 + b_1)W_2 + b_2 \tag{12}$$

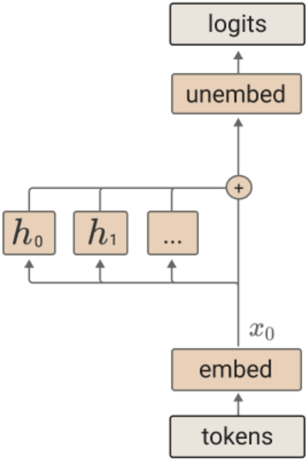
The original paper uses a factor of 4.

Scaling to Deeper Networks – A Better Interpretation



Deep neural architectures struggle with vanishing gradients, when trained at scale.

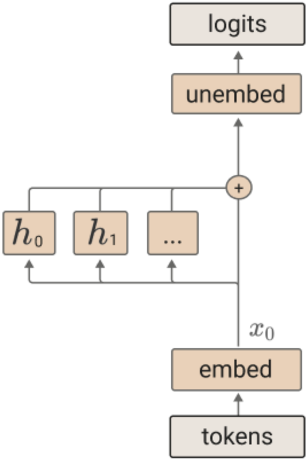
Scaling to Deeper Networks – A Better Interpretation



Deep neural architectures struggle with vanishing gradients, when trained at scale.

The approach used universally is to add "skip connections" to maintain a short gradient path, and mitigate this concern.

Scaling to Deeper Networks – A Better Interpretation

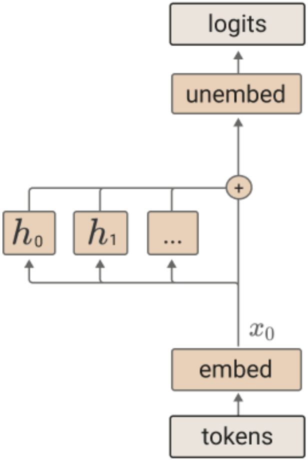


Deep neural architectures struggle with vanishing gradients, when trained at scale.

The approach used universally is to add "skip connections" to maintain a short gradient path, and mitigate this concern.

A result of this setup is that we can interpret each attention head and MLP as "reading from" and "writing to" a residual stream.

Scaling to Deeper Networks – A Better Interpretation



Deep neural architectures struggle with vanishing gradients, when trained at scale.

The approach used universally is to add "skip connections" to maintain a short gradient path, and mitigate this concern.

A result of this setup is that we can interpret each attention head and MLP as "reading from" and "writing to" a residual stream.

In addition, we also perform layer normalization over the latent vectors before MLP & self-attention.

Inference & Training

Autoregressive Sampling: The output of the model during a single forward pass is a token. added to the input, and the forward pass is run once again. This creates the final output, detokenized to a sentence.

Inference & Training

Autoregressive Sampling: The output of the model during a single forward pass is a token. added to the input, and the forward pass is run once again. This creates the final output, detokenized to a sentence.

Training: Each token sequence $\{x_i\}_{t=1}^{T}$ contains $T-1$ targets.

$$\mathcal{L}_{CE}(\{x\}_{t=1}^{T}, \theta) = \frac{1}{T-1} \sum_{t'=1}^{T-1} \mathcal{L}_{CE}(f_\theta(\{x_{t'+1} | x_t\}_{t=1}^{t'}), x_{t'+1}) \tag{13}$$

Inference & Training

Autoregressive Sampling: The output of the model during a single forward pass is a token. added to the input, and the forward pass is run once again. This creates the final output, detokenized to a sentence.

Training: Each token sequence $\{x_i\}_{t=1}^{T}$ contains $T-1$ targets.

$$\mathcal{L}_{CE}(\{x\}_{t=1}^{T}, \theta) = \frac{1}{T-1} \sum_{t'=1}^{T-1} \mathcal{L}_{CE}(f_\theta(\{x_{t'+1} | x_t\}_{t=1}^{t'}), x_{t'+1}) \tag{13}$$

However, training these blocks means attention would be able to *look into the future*.

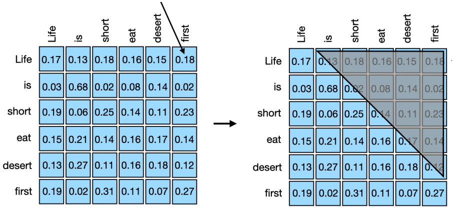
Inference & Training

Autoregressive Sampling: The output of the model during a single forward pass is a token. added to the input, and the forward pass is run once again. This creates the final output, detokenized to a sentence.

Training: Each token sequence $\{x_i\}_{t=1}^{T}$ contains $T-1$ targets.

$$\mathcal{L}_{CE}(\{x\}_{t=1}^{T}, \theta) = \frac{1}{T-1} \sum_{t'=1}^{T-1} \mathcal{L}_{CE}(f_\theta(\{x_{t'+1} | x_t\}_{t=1}^{t'}), x_{t'+1}) \quad (13)$$

However, training these blocks means attention would be able to *look into the future*. We correct this by applying an upper-trianguler causal mask:



Reviewing NanoGPT

If you can view this screen, I am making a mistake.

Thank you!

Have an awesome rest of your day!

Slides: `https://cs.purdue.edu/homes/jsetpal/slides/transformer.pdf`