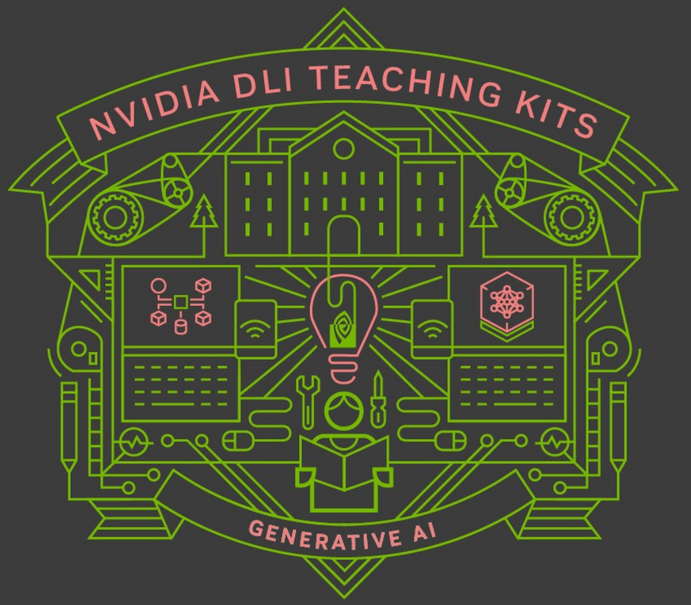
Lecture 3.2 - The Transformer Architecture

Generative AI Teaching Kit



This lecture

- Recap on challenges with LSTMs
- Attention is all you need – breakthrough
- The Transformer Block
- Multi-headed Attention
- Positional Encoding
- Original Transformer Architecture

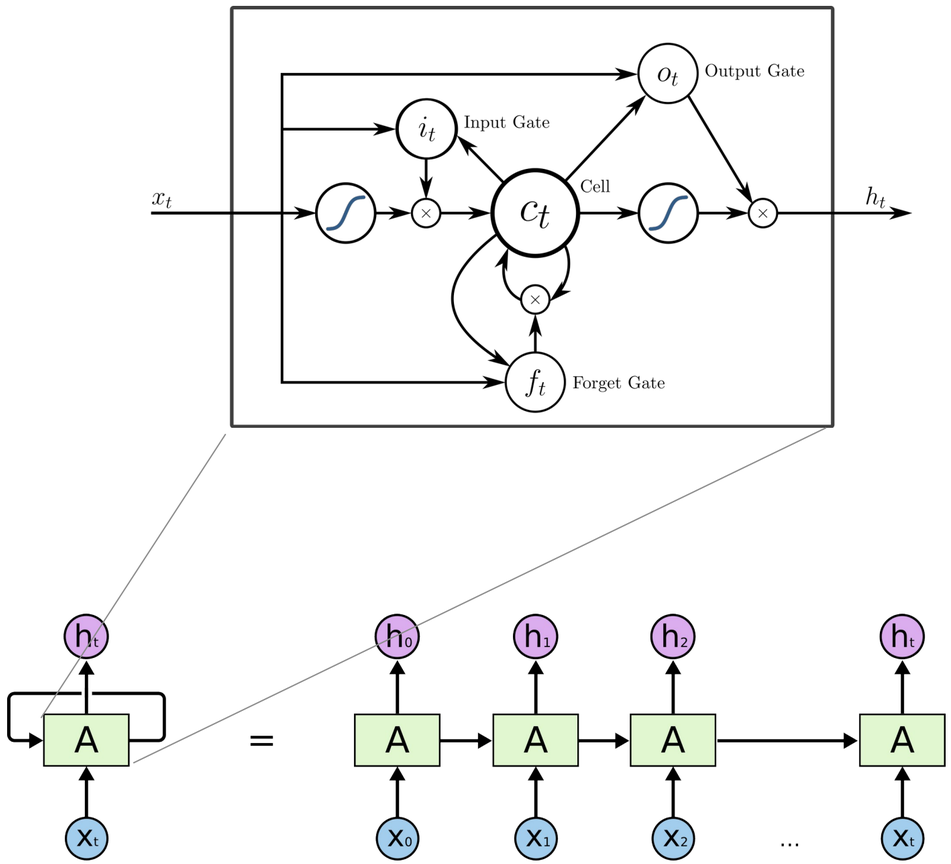
Recap on challenges with LSTMs

The limitations with sequence deep learning models

Last time we saw that deep learning language models required a special treatment of the causal-sequence nature of language.

This meant that special architectures like Recurrent Neural Networks and LSTMs became the standard for language modeling in deep learning.

These models used memory components to keep track of parts of sequences but still lacked some important components and encountered limitations.



The limitations with sequence deep learning models

One key missing feature to vanilla RNNs and LSTMs was the concept of **attention.**

Attention is defined as a mechanism that enables a model to focus on the most relevant parts of the input while making predictions.
Instead of treating all input information equally, it assigns varying levels of "importance" (weights) to different parts based on the task.

Innovations began to emerge in the late 20-`teens` but still all focused-on applications of the LSTMs/RNN models and were hampered by those models' limitations.

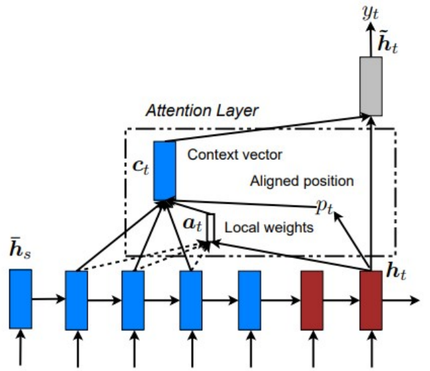


Figure 3: **Local attention model** – the model first predicts a single aligned position p_t for the current target word. A window centered around the source position p_t is then used to compute a context vector $\boldsymbol{c}_t$, a weighted average of the source hidden states in the window. The weights $\boldsymbol{a_t}$ are inferred from the current target state $\boldsymbol{h}_t$ and those source states $\bar{\boldsymbol{h}}_s$ in the window.

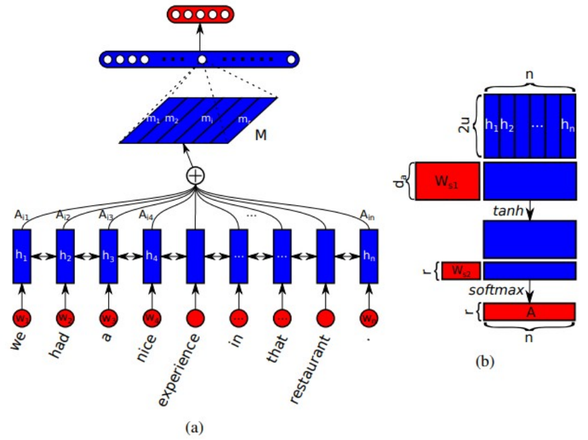


Figure 1: A sample model structure showing the sentence embedding model combined with a fully connected and softmax layer for sentiment analysis (a). The sentence embedding M is computed as multiple weighted sums of hidden states from a bidirectional LSTM ($\mathbf{h_1}, ..., \mathbf{h_n}$), where the summation weights ($A_{i1}, ..., A_{in}$) are computed in a way illustrated in (b). Blue colored shapes stand for hidden representations, and red colored shapes stand for weights, annotations, or input/output.

DARTMOUTH ENGINEERING | NVIDIA

Attention is all you need – breakthrough

Attention is all you need – the game changing paper

In 2017, a paper was presented at the 31st Neural Information Processing Systems Conference.

This paper introduced a new model architecture, the transformer, which **uses as a central point, self attention**, and **removed any need for recurrence or convolution layers**.

The focus of this new architecture on only attention, and some feedforward networks, meant that this model could process sequences in parallel, with drastically improved efficiency in training.

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

*Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

†Work performed while at Google Brain.

‡Work performed while at Google Research.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

DARTMOUTH ENGINEERING | NVIDIA

Attention is all you need – the game changing paper

This new model, the Transformer, was shown to outperform state-of-the-art machine translation models using a fraction of the training cost.

The benefits of this architecture was also in the generalizable nature of its design. This model was not designed for machine translation specifically and could be re-trained for other tasks such as English constituency parsing.

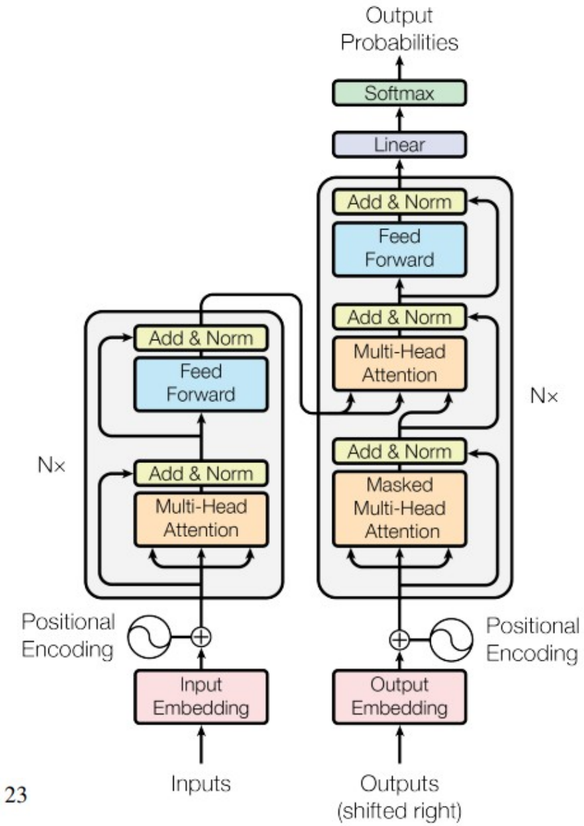


Figure 1: The Transformer - model architecture.

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Table 4: The Transformer generalizes well to English constituency parsing (Results are on Section 23 of WSJ)

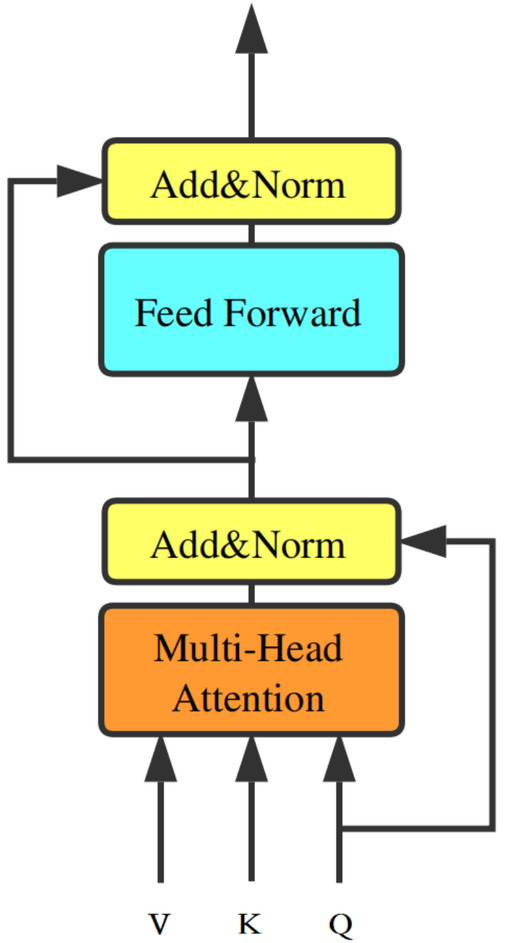
Parser	Training	WSJ 23 F1
Vinyals & Kaiser el al. (2014) [37]	WSJ only, discriminative	88.3
Petrov et al. (2006) [29]	WSJ only, discriminative	90.4
Zhu et al. (2013) [40]	WSJ only, discriminative	90.4
Dyer et al. (2016) [8]	WSJ only, discriminative	91.7
Transformer (4 layers)	WSJ only, discriminative	91.3
Zhu et al. (2013) [40]	semi-supervised	91.3
Huang & Harper (2009) [14]	semi-supervised	91.3
McClosky et al. (2006) [26]	semi-supervised	92.1
Vinyals & Kaiser el al. (2014) [37]	semi-supervised	92.1
Transformer (4 layers)	semi-supervised	92.7
Luong et al. (2015) [23]	multi-task	93.0
Dyer et al. (2016) [8]	generative	93.3

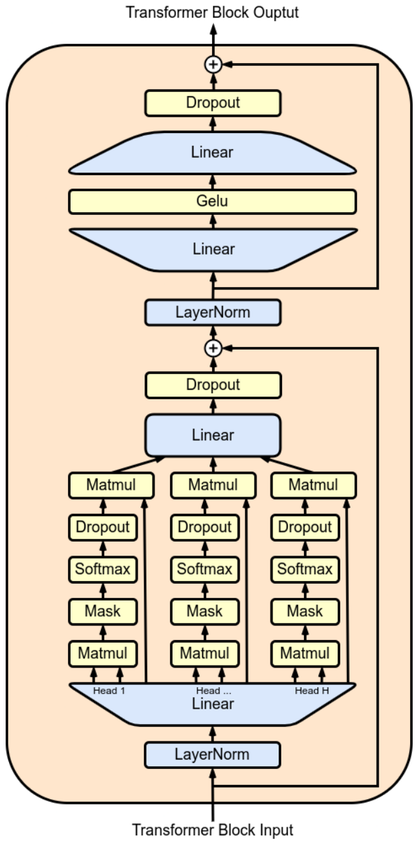
The Transformer Block

The Transformer Block

1. **Multi-Head Attention (Orange Block)**:
    1. **Input**: Queries (Q), Keys (K), and Values (V).
    2. **Process**: Attention weights are calculated to decide how much focus each token should give to others in the sequence. Multiple heads capture different types of relationships.
    3. **Output**: A weighted representation of the sequence.
2. **Add & Norm (Yellow Block)**:
    1. The output of the Multi-Head Attention is **added** back to the input (residual connection) to preserve the original information.
    2. Then, it is **normalized** to stabilize training and ensure smooth gradient flow.
3. **Feed Forward (Blue Block)**:
    1. A fully connected network processes each token independently, adding non-linearity and helping the model capture complex patterns.
4. **Add & Norm (Yellow Block)**:
    1. The output of the Feed Forward layer is **added** back to its input (another residual connection).
    2. **Normalization** is applied again for stability.

This flow repeats across multiple layers, enabling the Transformer to model relationships across all tokens efficiently while keeping computations stable and balanced.





The Transformer Block – Tokenization and Word Embeddings

Preparing inputs for the Transformer involves three main steps:

Tokenization

- The text input is split into smaller units called **tokens**. These tokens could represent words, subwords, or even individual characters, depending on the tokenization strategy.
- Each token is assigned a unique identifier based on the model's vocabulary, converting the text into a numerical sequence.

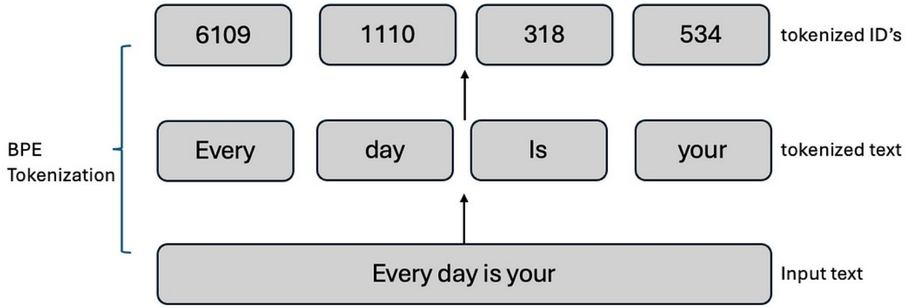
Word Embedding

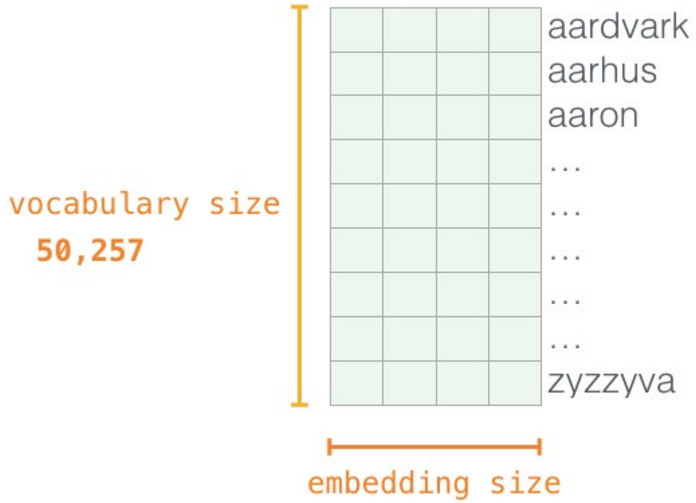
- Each token ID is mapped to a high-dimensional vector using a pre-trained or learned embedding matrix.
- These embeddings capture semantic meaning, allowing the model to understand relationships between words or tokens (e.g., synonyms or contextual similarities).

Positional Encoding*

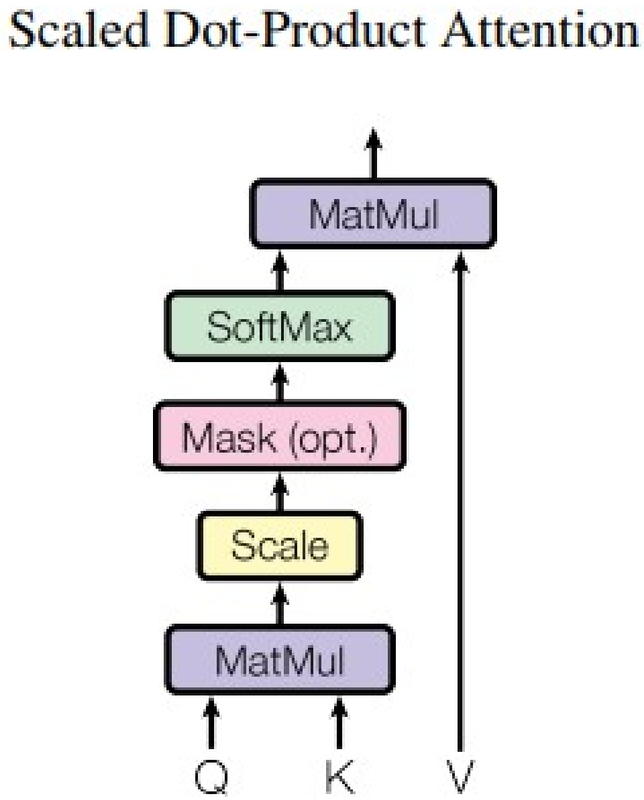
- Since Transformers process all tokens simultaneously, they lack an inherent sense of order.
- Positional encoding is added to embeddings to inject information about token positions in the sequence, enabling the model to capture word order.

*We'll see more of this soon.





Scaled Dot-Product Attention



Definition:
Attention maps a query and a set of key-value pairs to an output, where:

- **Query (Q):** What you're looking for.
- **Key (K):** What the data is indexed by.
- **Value (V):** The information being retrieved.

How It Works:
1. Compute the dot product of the query with all keys to measure compatibility.
2. Scale the result by dk\sqrt{d_k}dk to prevent large values.
3. Apply a **SoftMax function** to convert scores into probabilities (weights).
4. Use the weights to compute a weighted sum of the values.

Key Points:

- Faster and more space-efficient than additive attention due to matrix operations.
- Scaling improves stability by preventing the softmax from collapsing for large dimensions.

The Transformer Block – Position-wise Feedforward Network

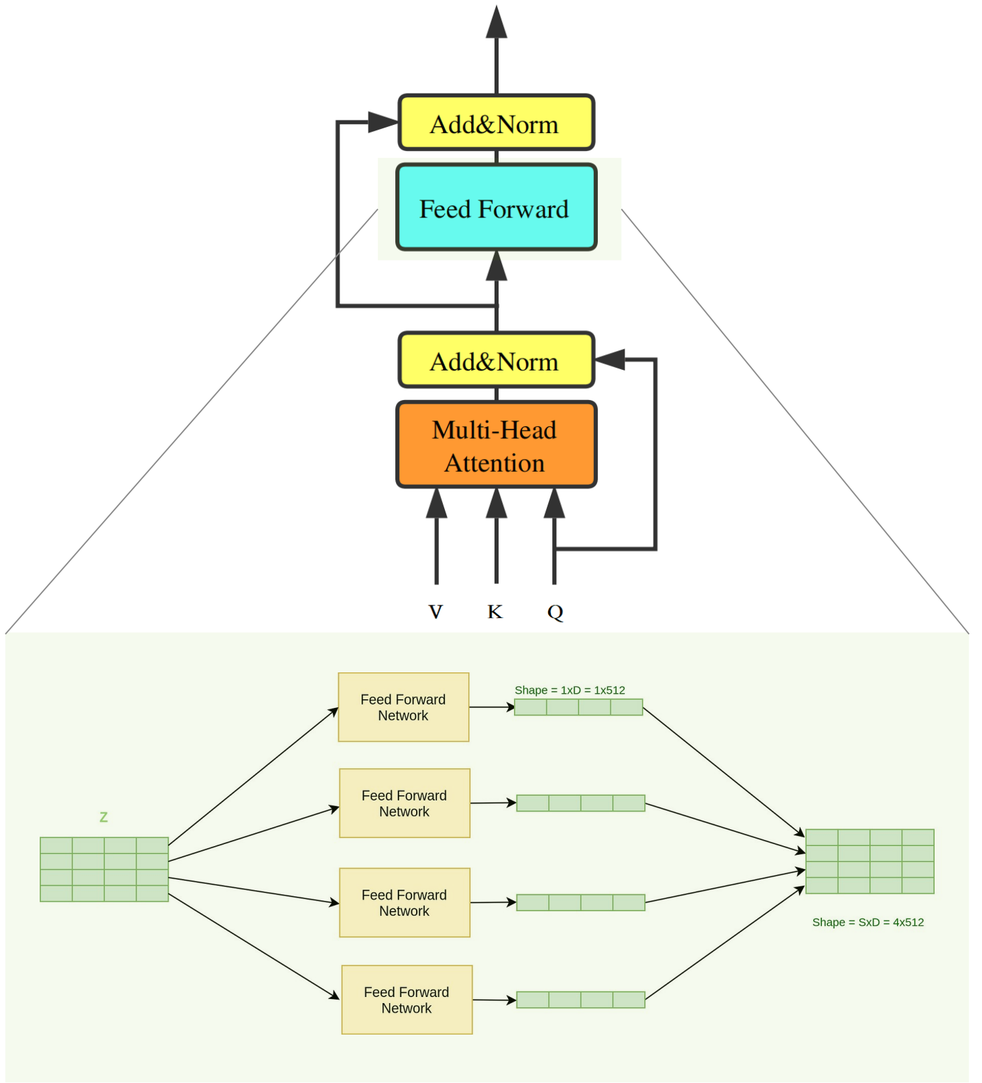
If we look at the progress of a token as it passes through the Transformer, we can see that it is processed in the following ways:

- Tokenization – *Linear Operation*
- Word Embedding – *Linear Operation*
- Self-Attention – *Linear Operation*

So, how does non-linearity get added into the Transformer? Ans: **Position-wise Feedforward Neural Network**

<u>What is it and how does it work?</u>
A feedforward neural network is applied independently to each token's embedding at each position in the sequence. This adds non-linearity and expressive power to the Transformer. Each feedforward layer consists of two fully connected layers with a non-linear activation in between

<u>Why It Works:</u>
Non-linearity (e.g., ReLU) enables the model to capture complex relationships that linear operations cannot. The position-wise nature means each token is processed independently, preserving parallelizability.

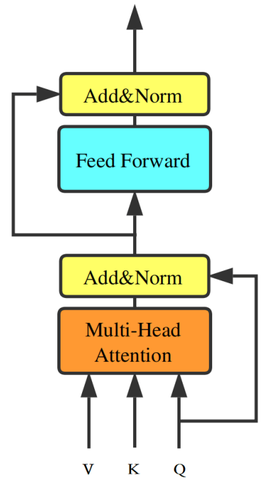


The Transformer Block – Residual Connection and Normalization

Residual Connections

Residual connections, also called skip connections, allow the input of a layer to bypass the layer and be added directly to its output.

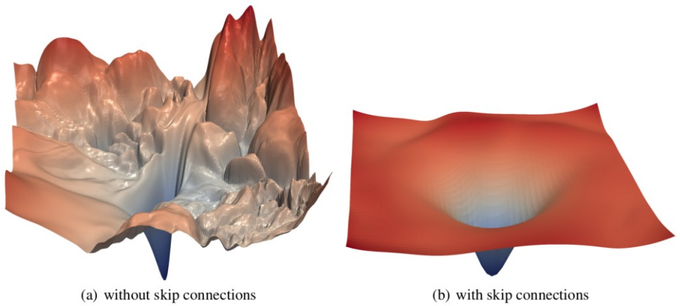
- **Facilitate Gradient Flow:** Prevent vanishing or exploding gradients in deep networks by ensuring gradients can propagate back through the network effectively.
- **Improve Convergence:** Enable faster training by providing a "shortcut" for information.
- **Preserve Information:** Retain information from earlier layers, which might otherwise be overwritten during training.



Layer Normalization

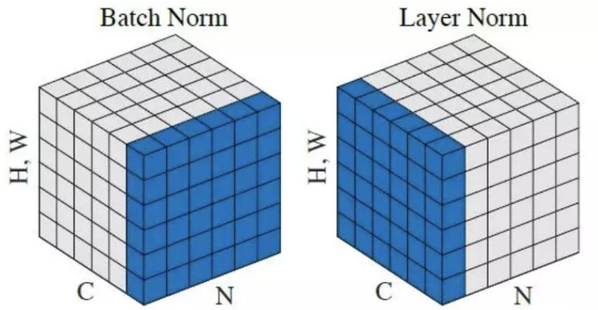
Layer normalization standardizes the activations across the features of a token at each position, ensuring that the input to each layer has zero mean and unit variance.

- **Stabilize Training:** Normalization ensures consistent scaling of activations, making training more stable.
- **Prevent Exploding Activations:** Keeps activations within a manageable range, preventing instability.
- **Enable Robust Generalization**: Helps the model generalize better by reducing sensitivity to the scale of the input data.



Why Both?

- Residual connections ensure information flow across the network.
- Layer normalization stabilizes the output at each step, improving gradient flow and model convergence.



DARTMOUTH ENGINEERING | NVIDIA

Multi-headed Attention

Self-Attention – Multiheaded Attention

The authors also implemented “multi-headed attention” to address the limitations of single-headed attention, which could only focus on one type of relationship or dependency at a time.

Parallel Attention Mechanisms:
Multi-headed attention allows the model to focus on different parts of the input sequence simultaneously, capturing diverse relationships (e.g., word-to-word, phrase-to-phrase).

Improved Representations:
Each "head" learns unique attention patterns, enabling the model to extract more nuanced information, such as syntactic and semantic dependencies.

Scalability and Flexibility:
By splitting the computation into smaller attention heads, the model efficiently handles large inputs and produces richer, context-aware embeddings.

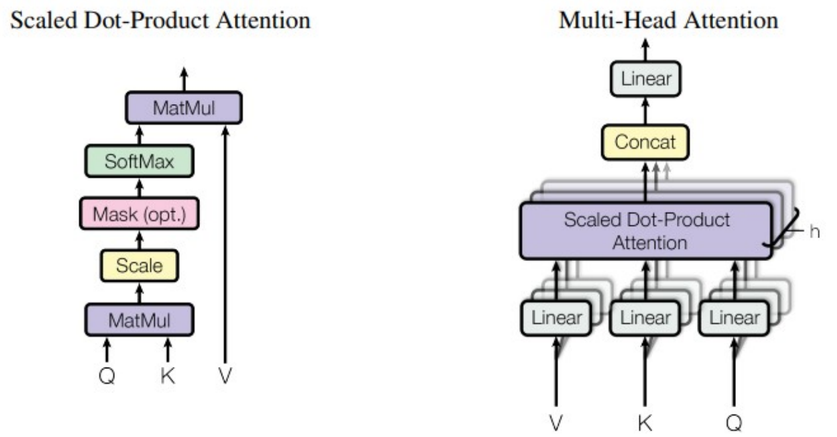
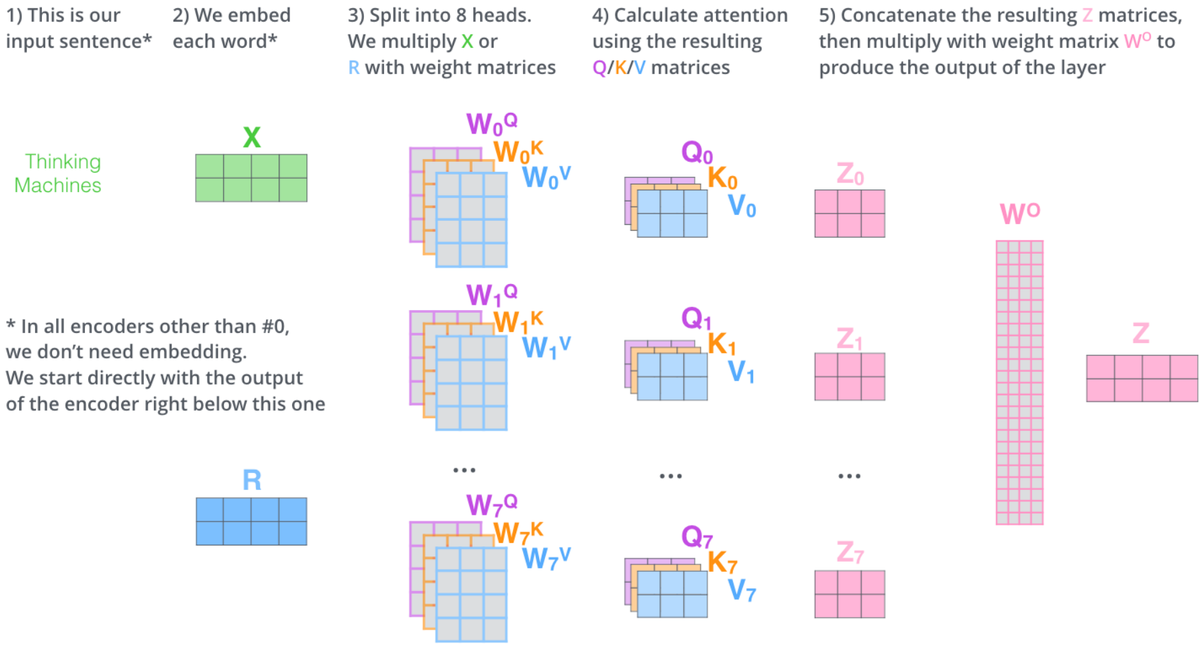


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.



Variations of Multiheaded Attention

GQA (Grouped-Query Attention)
Reduces the computational cost by grouping multiple queries together and computing attention collectively rather than individually.
Key Advantage: Drastically lowers memory and compute requirements, especially for large-scale models like GPT-3.

Multi-Query Attention
Instead of having separate key-value pairs for each query, all queries share the same set of key-value pairs.
Key Advantage: Reduces the memory overhead of storing multiple key-value pairs without significantly impacting model performance.

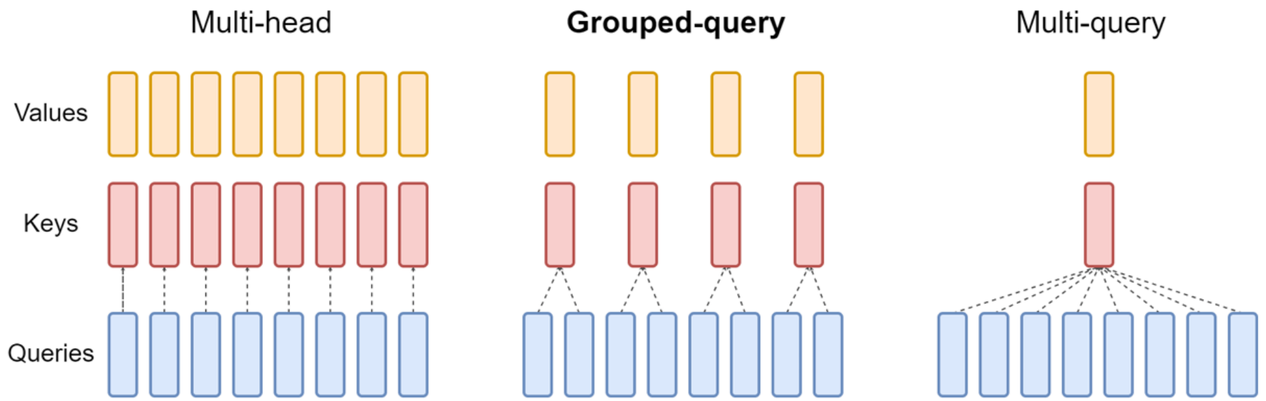


Figure 2: Overview of grouped-query method. Multi-head attention has H query, key, and value heads. Multi-query attention shares single key and value heads across all query heads. Grouped-query attention instead shares single key and value heads for each *group* of query heads, interpolating between multi-head and multi-query attention.

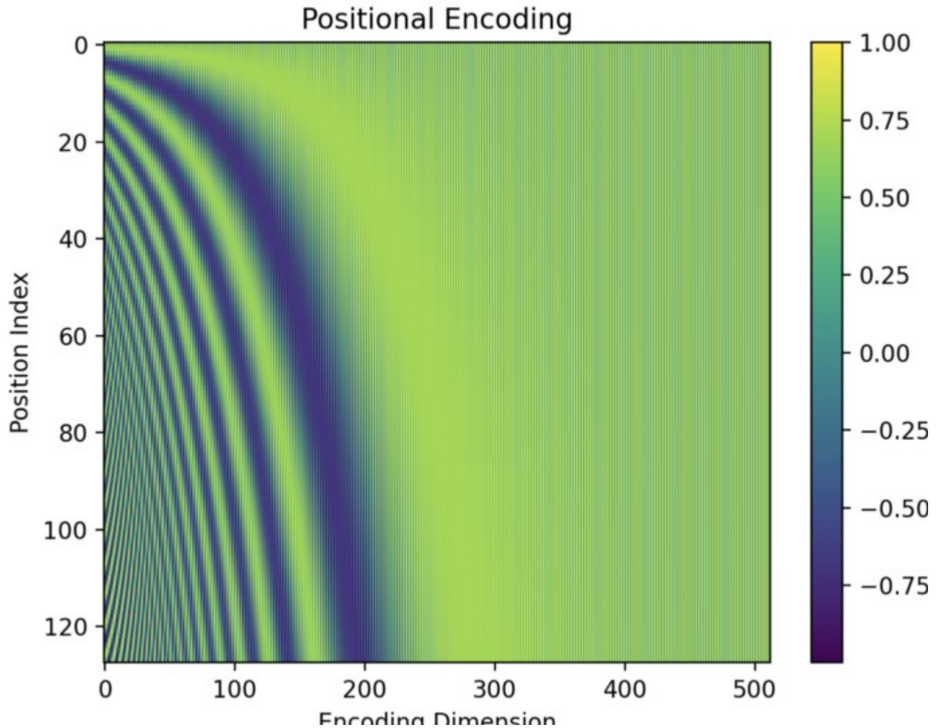
Positional Encoding

Positional Encoding

The original Transformer relies on positional encoding to inject information about the order of tokens into the model. The self-attention mechanism itself lacks positional awareness, and in language we need to preserve this sequential understanding.

How Positional Encoding Works

- Positional encodings are added to the input embeddings to encode the order of tokens in the sequence.
- The sinusoidal functions used in the original paper allow the model to generalize to sequences longer than those seen during training because the values are periodic and predictable.
- Relative positions between tokens are naturally encoded through phase differences in sine and cosine values.

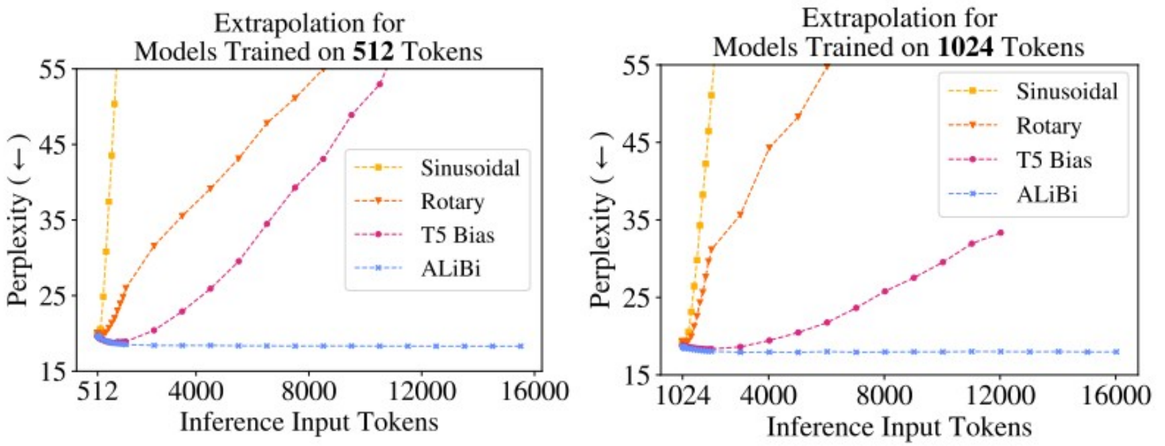


$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Limitations of Sinusoidal Positional Encoding

While sinusoidal positional encodings from the original Transfor
paper have been highly effective, they come with notable limitati



Fixed and Non-Adaptive:

Sinusoidal encodings are fixed and do not adapt to specific datasets or t
This inflexibility can limit their ability to capture task-specific positional information.

Absolute Position Encoding:

They encode absolute positions rather than explicitly modeling relative distances between tokens. While the differences in sine/cosine phases indirectly encode relative positions, this is not optimized for tasks requiring strong relative positioning (e.g., question answering or summarization).

Lack of Scalability:

When handling very long sequences, the sin\sinsin and cos\coscos functions may lose precision due to numerical instability in their periodicity.

Inefficiency for Long Contexts:

Sinusoidal encodings struggle to efficiently represent relationships for extremely long sequences, which are increasingly common in models like GPT-style LLMs designed for multi-turn dialogues or document understanding.

Improvements to Sinusoidal Positional Encoding

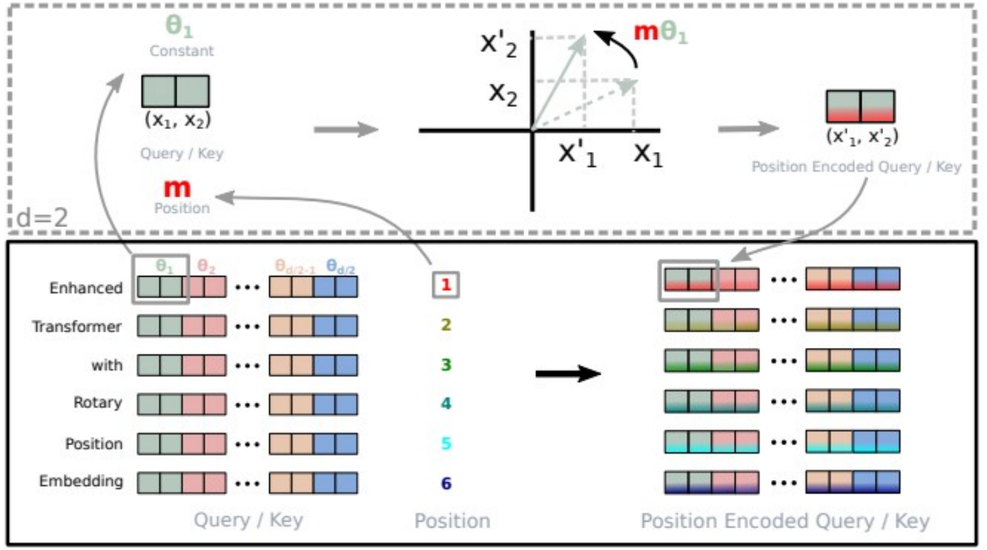
Rotary Positional Embedding (RoPE)

Rotary Positional Embedding introduces relative position encoding directly into the attention mechanism, making the model more effective at capturing positional relationships. This transformation embeds relative positional information naturally into the attention scores without modifying the self-attention computation pipeline.

Key Advantages:

Relative Position Awareness: Unlike sinusoidal embeddings, RoPE explicitly models relative positions, which is crucial for tasks involving long-term dependencies.

Efficient for Long Contexts: Handles long sequences better than sinusoidal encodings due to its explicit relative positioning.



ALiBi (Attention with Linear Biases)

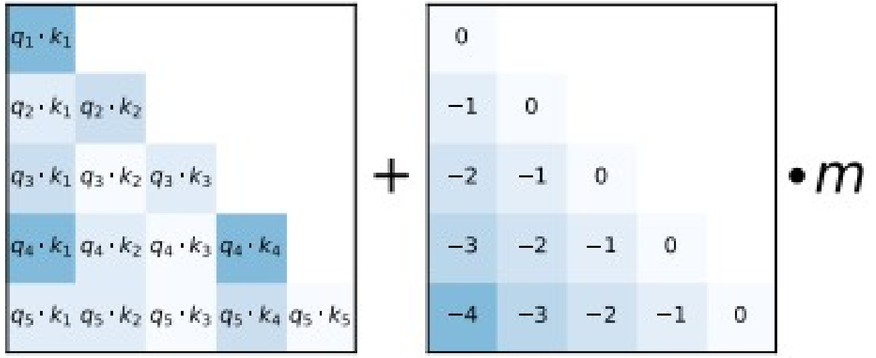
Attention with Linear Biases, modifies the attention mechanism by adding a learnable linear bias to the attention scores based on the relative distance between tokens.

Key Advantages:

Efficient Scaling: ALiBi introduces no additional parameters or overhead, as the bias is directly applied to attention scores.

Strong Relative Position Encoding: By modifying attention weights directly, ALiBi enhances the model's ability to capture relative positional information efficiently.

Supports Extrapolation: ALiBi enables models to generalize better to longer sequences during inference, even when trained on shorter sequences.



Original Transformer Architecture

Cross-Attention

Cross-attention is a mechanism where one sequence (e.g., the decoder input) **attends to another sequence** (e.g., the encoder output).
This allows the model to integrate information from a source sequence (like an input sentence in machine translation) into the target sequence generation process.

In the Transformer architecture, cross-attention is implemented in the decoder block, where the decoder uses the encoder's output to guide the generation of the target sequence.

Cross-Attention Inputs
Key-Value Pairs (K, V):
These come from the encoder outputs.They represent the context learned from the source sequence (e.g., input sentence in a translation task).

Query (Q):
This comes from the decoder's previous layer (or the embeddings of the previously generated tokens in the decoder).

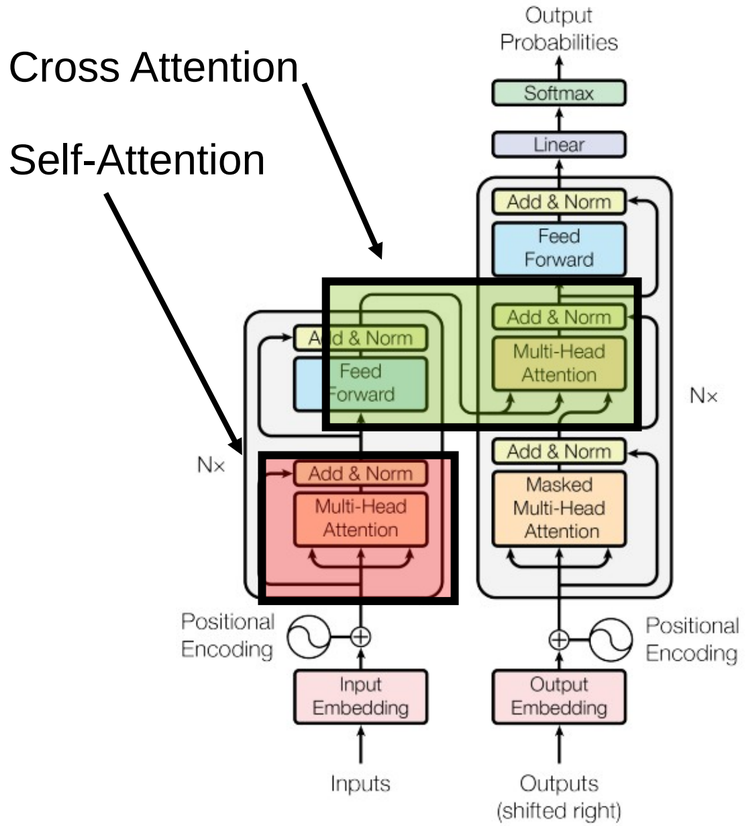


Figure 1: The Transformer - model architecture.

The Transformer

Now that we can see what is inside the transformer blocks, let's talk about the original transformer and what it was used for.

Architecture Details:

- Transformer Blocks/Layers: **6** encoder and **6** decoder layers.
- Hidden Dimension for the word embeddings: **1024**
- Feedforward Hidden Dimension: **4096** (ie a 4x expansion internally)
- Attention Heads for multi-headed attention: **16**
- Sequence Length: Up to 512 tokens.

Total Parameters: 213 million (213M)

Training Details

- Task: English to German Translation
- Batch Size: 131,072 tokens per batch
- Training Steps: 100 k
- Dataset: WMT 2014 English-to-German (4.5M sentence pairs).

Training Time: ~3.5 days on 8xP100 GPUs for English-to-German

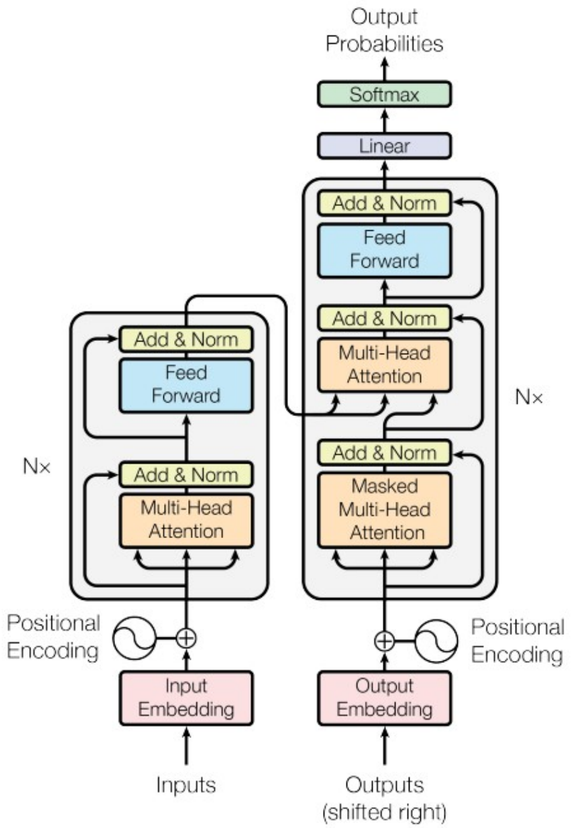


Figure 1: The Transformer - model architecture.

Wrap Up

The Transformer Architecture

- Today we introduced one of the most exciting innovations of the last decade, the Transformer.
- We looked inside the model to see how each part of this architecture is constructed and why it offered the breakthrough in sequence analysis over existing models.
- We looked at the original attention mechanism in the attention is all you need paper and how it was scaled to produce the Transformer.
- Now that we have an understanding of the fundamental piece of Generative AI, we can look further and into even more exciting and exotic models.

In the next class we will explore what it takes to train a model like the transformer. The data preparation and the considers required for autoregressive modeling.

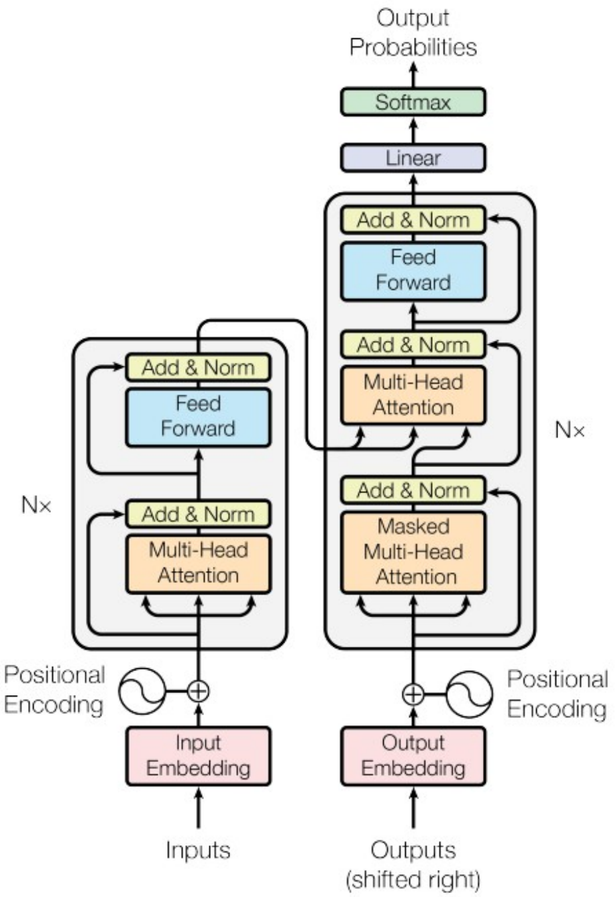


Figure 1: The Transformer - model architecture.

Thank you!