

# 1 Attention

**Motivation:** One of the core mechanisms inside of current LLMs.

**Warning:** Convolution takes in a single node, while attention takes in all nodes.

## 1.1 Transformer Architecture

Notes:

- **Transformer Layer:**
  - Attention mechanism (multi-headed)
  - Positional encodings
- With massive unsupervised datasets:
  - Masked self-supervised training
  - Contrastive training

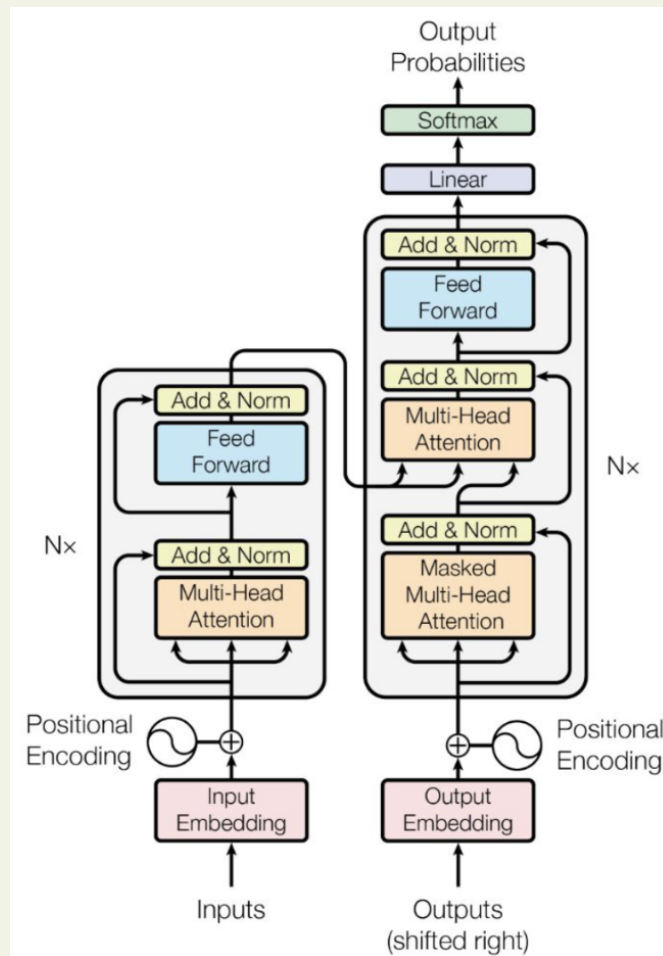


Figure 1: LS: Encoder, RS: Decoder

### 1.1.1 Transformer Layer

#### Summary:

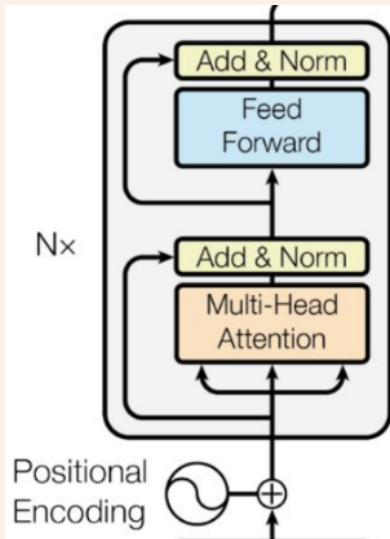


Figure 2

Component	Description
Positional encoding	Learn to map integer positions into a vectorized representation. $PE_{(pos,i)} = \begin{cases} \sin\left(\frac{pos}{10000^{\frac{i}{d_{model}}}}\right) & \text{if } i \text{ is even} \\ \cos\left(\frac{pos}{10000^{\frac{i-1}{d_{model}}}}\right) & \text{if } i \text{ is odd} \end{cases}$
Multi-Head Attention	Computes attention scores for each token in the sequence.
LayerNorm	Stabilizes activations and accelerates training.
Residual Connection	Preserve information and enable deeper networks.
FFN/MLP	Increases the expressive power of the learned representation, often using GELU activations.

## 1.2 Transformers are GNNs

#### Summary: Transformers are a special case of GNN

	GNN	Transformer
<b>Connectivity (Adjacency)</b>	Sparse	Full
<b>Edge Learning</b>	Yes	No (Implicitly)
<b>Message Computation</b>	$M(n_i, n_j, e_{ij})$	$\langle n_i, n_j \rangle$
<b>Communication per step</b>	# Number of Neighboring nodes	$\sim$ # Number of Heads
<b>Data requirements</b>	Low	High
<b>Computation</b>	Slow due to gather operations	Fast, Optimizable $\sim$ Matrix Multiplications
<b>Training</b>	Straightforward	Pre-training is needed

### 1.3 Attention Mechanism

#### Process:

1. **Inputs:** Tokens tensor, Mask
  - **Tokens:** Inputs for Transformer/Attention Layers, which is a numerical representation of pieces of data.
  - **Mask:** A binary matrix that indicates which tokens to give attention to.
2. **Preprocessing:** Linear maps Tokens into Queries, Keys, and Values.
  - $Q = \text{Tokens} \cdot W_Q$ : Represents the current token's context.
  - $K = \text{Tokens} \cdot W_K$ : Represents the context of all tokens.
  - $V = \text{Tokens} \cdot W_V$ : Represents the information to be passed on.
3. **Attention scores:**  $\text{Scores} = \frac{QK^T}{\sqrt{d_k}} \cdot \text{Mask}$ 
  - $\text{score}_{ij} = \frac{(q_i \cdot k_j)m_{ij}}{\sqrt{d_k}}$
4. **Attention Normalization:** Attention Weights =  $\text{softmax}(\text{scores})$ 
  - $\text{score}_{ij}^{\text{normalized}} = \frac{\exp(\text{score}_{ij})}{\sum_{k=1}^n \exp(\text{score}_{ik})}$
5. **Value update:** New Values = Attention Weights  $\cdot V$ 
  - $v_i^{\text{new}} = \sum_{j=1}^n \text{score}_{ij}^{\text{normalized}} v_j$
6. **Post Processing:** Apply LayerNorm, Residual connections, and a FFN.
7. **Outputs:** Updated tokens tensor

$$\text{Attention}(Q, K, V, M) = \text{softmax}\left(\frac{QK^T M}{\sqrt{d_k}}\right) V \quad (1)$$

#### 1.3.1 Attention Maps: Visualizing Where the Model Attends

**Notes:** Softmax bias:

- Values between 0 and 1
- Categorical like
- Attend to one token at a time

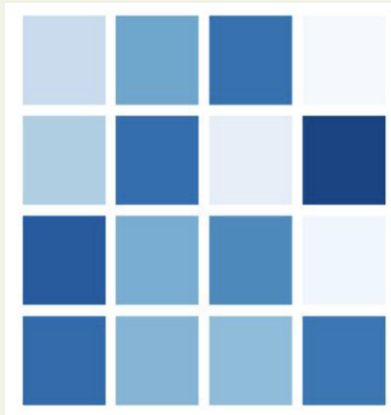


Figure 3

### 1.3.2 Self-Attention vs. Cross-Attention

Notes:

- **Self-Attention:** Attention allows connections between the same sequence without masking.

$$\begin{aligned}\text{Self-Attention}(x, \text{mask}) &= \text{Attention}(\text{Linear}(x), \text{Linear}(x), \text{Linear}(x), \text{mask}) \\ &= \text{Attention}(Q(x), K(x), V(x), \text{mask})\end{aligned}$$

- **Cross-Attention:** Attention allows connections between different sequences.

$$\begin{aligned}\text{Cross-Attention}(x, x', \text{mask}) &= \text{Attention}(\text{Linear}(x), \text{Linear}(x'), \text{Linear}(x'), \text{mask}) \\ &= \text{Attention}(Q(x), K(y), V(x), \text{mask})\end{aligned}$$

### 1.3.3 Multi-Headed Attention

Notes: Multiple attention mechanisms in parallel, each with different linear maps.

- Ensemble-like approach.
- Same compute and # of parameters.
- **Strategy:** Expand and contract tensors to new axis: number of heads.

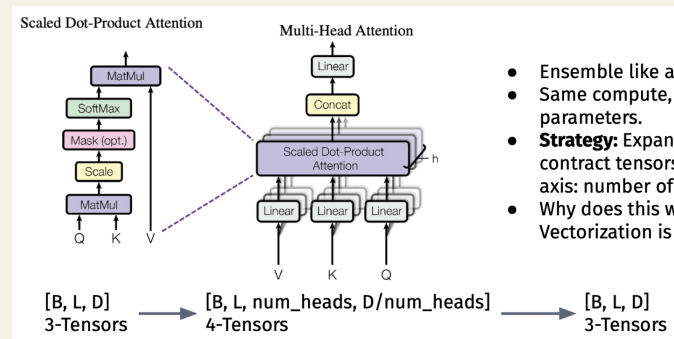


Figure 4

## 1.4 Examples

### 1.4.1 Tokens

Example:



Figure 5

### 1.4.2 Positional Encoding

Example:

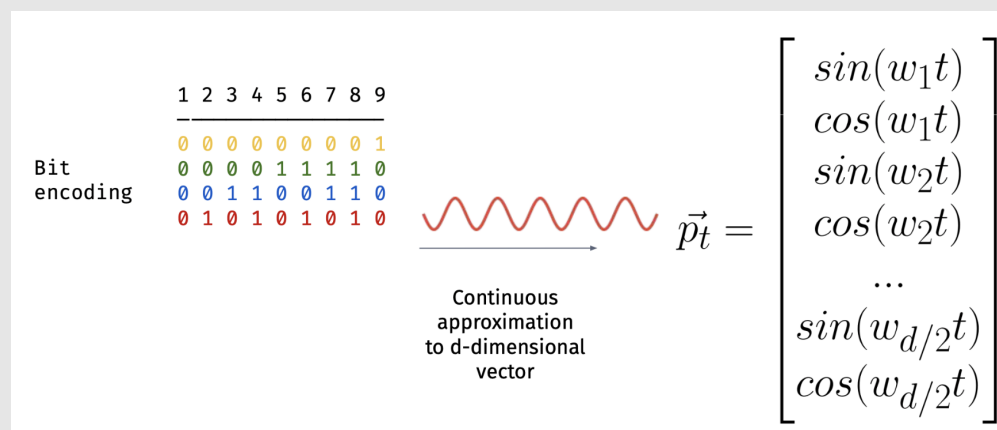


Figure 6

### 1.4.3 Cross-Attention

Example:

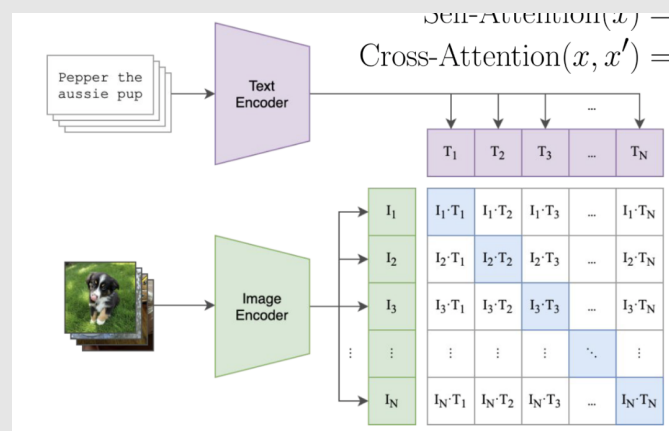


Figure 7

## 2 LLMs

### Notes:

- Transformers on large text-like datasets.
- Transformers on "tokens" (discretized data)
- Foundational models

### 2.1 Transformers & LLMs

#### Summary:

- 2.1.1 Inputs: Tokenizing Text & Embedding Layers
- 2.1.2 Outputs: Auto-Regressive Decoding of Tokens
- 2.1.3 Sizes of Text Datasets for LLMs
- 2.1.4 Text to Text Tasks
- 2.1.5 Transformers and Masking: Encoders and Decoders
- 2.1.6 Masking Language Modelling (Self-Supervised)

## 2.2 Scaling LLMs

Motivation:

### 2.2.1 Techniques

**Summary:** Table format

### 2.2.2 High-Level Impacts

**Summary:**



### 3 Transformers