# Study Material - Youtube

## Document Information

- **Generated:** 2025-08-26 10:45:13
- **Source:** https://www.youtube.com/watch?v=NAWe0F0PmRU
- **Platform:** Youtube
- **Word Count:** 2,351 words
- **Estimated Reading Time:** ~11 minutes
- **Number of Chapters:** 5
- **Transcript Available:** Yes (analyzed from video content)

## Table of Contents



---

# Video Overview

This lecture, "Mathematical Foundations of Generative AI: Transformers Architecture," provides a deep dive into the core components that make up the decoder side of a Transformer model, particularly in the context of autoregressive generative tasks. The instructor meticulously explains the necessity and implementation of **Causal Masking** to enforce the autoregressive property. The lecture then builds upon this by introducing the **Multi-Head Attention** mechanism, detailing how it enhances the model's ability to capture diverse relationships within the data. Finally, it covers the role of **Fully Connected Layers** and the final **Output Layer** in the overall architecture.

### Learning Objectives

Upon completing this lecture, students will be able to: - Understand the concept of **causality** in autoregressive models and why standard attention mechanisms violate it. - Mathematically formulate and explain **Causal Masking** using a mask matrix. - Describe the intuition and architecture of **Multi-Head Attention**, including the role of parallel heads and projection matrices. - Follow the mathematical flow of data through the Multi-Head Attention block, from input splitting to final concatenation and projection. - Explain the function of the **position-wise Fully Connected Layer** within a Transformer block. - Understand how the final output layer generates a probability distribution over the vocabulary.

### Prerequisites

To fully grasp the concepts in this video, students should have a foundational understanding of: - **Linear Algebra**: Matrix multiplication, matrix dimensions, and transpose operations. - **Calculus**: The concept of the exponential function and its behavior. - **Basic Neural Networks**: The idea of learnable weights, biases, and activation functions like Softmax and ReLU. - **Core Attention Mechanism**: Familiarity with the Query (Q), Key (K), and Value (V) concepts and the scaled dot-product attention formula. - **Autoregressive Models**: A basic understanding of how models generate sequences token by token, where each new token depends on the previously generated ones.

### Key Concepts

- Causal Masking (or Masked Attention)

- Autoregressive Property
- Multi-Head Attention
- Position-wise Fully Connected Layers
- Output Layer with Softmax

---

# Causal Masking and Causal Attention

This section explains a critical modification to the standard attention mechanism that is essential for building autoregressive models like GPT.

## Intuitive Foundation

**(00:26 - 02:15)**

In an **autoregressive (AR) model**, the goal is to predict the next token in a sequence based on all the preceding tokens. For example, to predict the 5th word in a sentence, the model should only use information from the 1st, 2nd, 3rd, and 4th words. It should **not** have access to the 6th, 7th, or any subsequent words. This is the principle of **causality**: the future cannot influence the past.

The standard attention mechanism, however, is inherently non-causal. Its attention score matrix, $A$, is calculated as:

$$A = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)$$

This calculation allows every token to attend to every other token in the sequence, including those that appear later. This is often called "looking into the future" and it violates the fundamental requirement of an autoregressive model.

> **Key Insight:** To use the powerful attention mechanism in an autoregressive setting, we must prevent tokens from attending to future tokens. **Causal Masking** (also known as **Masked Attention** or **Causal Attention**) is the technique used to achieve this. It works by "masking" or hiding the scores of future tokens before the softmax function is applied, effectively forcing their attention weights to zero.

The following diagram illustrates the difference in information flow between standard (non-causal) attention and causal attention for a sequence of tokens.

```
graph TD
    subgraph Standard Attention (Non-Causal)
        direction LR
        T1_S["Token 1"]
        T2_S["Token 2"]
        T3_S["Token 3"]

        T2_S --> T1_S
        T2_S --> T3_S
        T1_S <--> T3_S
    end

    subgraph Causal Attention
        direction LR
        T1_C["Token 1"] --> T2_C["Token 2"] --> T3_C["Token 3"]
    end
```

```
linkStyle 0,1,2 stroke-width:2px,stroke:red,stroke-dasharray: 5 5
linkStyle 3,4 stroke-width:2px,stroke:green
```

*Caption: In standard attention, a token (e.g., Token 2) can "see" both past (Token 1) and future (Token 3) tokens. In causal attention, information flows only from past to future, so Token 2 can only see Token 1.*

## Mathematical Analysis of Causal Masking

**(02:15 - 08:35)**

To enforce causality, we modify the attention score calculation by adding a special **mask matrix**, denoted as $M$, before applying the softmax function.

### The Mask Matrix (M)

The mask matrix $M$ is designed to have the same dimensions as the score matrix $QK^T$, which is $T \times T$ (where $T$ is the sequence length). Its elements are defined as follows:

**(03:32) - Definition of the Mask Matrix**

$$M_{ij} = \begin{cases} 0 & \text{if } j \leq i \\ -\infty & \text{if } j > i \end{cases}$$

- **Intuitive Explanation:**
  - The indices `i` and `j` represent the row and column of the matrix, corresponding to the "querying" token and the "key" token, respectively.
  - **`j  i` (Past or Present):** When a token at position `i` is attending to a token at position `j` that is in its past or is itself, the mask value is `0`. Adding `0` to the attention score does not change it. This allows the token to attend to its past.
  - **`j > i` (Future):** When a token at position `i` attempts to attend to a token at position `j` that is in its future, the mask value is `-∞`.

Visually, the mask matrix for a sequence of length 4 would look like this:

$$M = \begin{pmatrix} 0 & -\infty & -\infty & -\infty \\ 0 & 0 & -\infty & -\infty \\ 0 & 0 & 0 & -\infty \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

### Modified Attention Calculation

The new, **masked attention** score matrix, $A_m$, is calculated by adding $M$ to the scaled dot-product scores before the softmax.

**(04:52) - Masked Attention Formula**

$$A_m = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + M\right)$$

- **Why this works:** The softmax function involves an exponential, $e^x$.
  - For future positions where we added $-\infty$, the term becomes $e^{-\infty}$, which approaches `0`.
  - After normalization (dividing by the sum of all exponentials in the row), the attention weights for all future positions become zero.
  - This effectively "masks" the future, ensuring that the output for a given token is only a weighted sum of the values from past and present tokens.

The resulting attention matrix $A_m$ will be a **lower-triangular matrix**, where all entries above the main diagonal are zero.

---

# Multi-Head Attention

**(08:37 - 21:20)**

Multi-head attention is an enhancement to the basic attention mechanism that allows the model to jointly attend to information from different representation subspaces at different positions.

## Intuitive Foundation

**(08:37 - 10:00)**

Instead of having just one attention mechanism, multi-head attention proposes running several attention mechanisms—called **"heads"**—in parallel. Each head can learn to focus on different types of relationships or features in the input sequence.

> **Analogy:** Imagine you are analyzing a complex sentence. You might have several "experts" looking at it simultaneously: - One expert (head) focuses on grammatical structure (e.g., subject-verb agreement). - Another expert (head) focuses on semantic relationships (e.g., "king" is related to "queen"). - A third expert (head) might track pronoun references (e.g., "it" refers to "the ball").

Multi-head attention allows the model to do this. It splits the model's representation space into several subspaces and learns a separate set of Query, Key, and Value matrices for each head. The final output is a combination of the insights from all these heads, providing a richer and more nuanced representation.

## Architectural and Mathematical Breakdown

**(10:00 - 21:20)**

Let $h$ be the number of attention heads. The core idea is to split the model's dimension, $d_m$, into $h$ smaller parts. Typically, the key dimension $d_k$ and value dimension $d_v$ for each head are set to be equal:

$$d_k = d_v = \frac{d_m}{h}$$

The process for multi-head attention is as follows:

1. **Create Separate Projections for Each Head:** For each head $j \in \{1, ..., h\}$, we learn a unique set of weight matrices for Query, Key, and Value:
   - $W_j^Q \in \mathbb{R}^{d_m \times d_k}$
   - $W_j^K \in \mathbb{R}^{d_m \times d_k}$
   - $W_j^V \in \mathbb{R}^{d_m \times d_v}$
2. **Calculate Q, K, V for Each Head:** The input matrix $X \in \mathbb{R}^{T \times d_m}$ is projected using these head-specific weights:
   - $Q_j = XW_j^Q$
   - $K_j = XW_j^K$
   - $V_j = XW_j^V$ Each of these resulting matrices $(Q_j, K_j, V_j)$ will have dimensions that reflect the smaller head-specific dimensions (e.g., $Q_j \in \mathbb{R}^{T \times d_k}$).
3. **Compute Attention for Each Head in Parallel:** The standard (masked) attention mechanism is applied to each head's Q, K, V triplet to produce an output matrix $Z_j$:

   $$Z_j = \text{Attention}(Q_j, K_j, V_j) = \text{softmax}\left(\frac{Q_j K_j^T}{\sqrt{d_k}} + M\right) V_j$$

   Each $Z_j$ is a matrix in $\mathbb{R}^{T \times d_v}$.

4. **Concatenate the Heads:** The outputs from all $h$ heads are concatenated along the feature dimension.

$$Z_c = \text{Concat}(Z_1, Z_2, \dots, Z_h)$$

The resulting concatenated matrix $Z_c$ will have dimensions $\mathbb{R}^{T \times (h \cdot d_v)}$. Since we set $d_v = d_m/h$, this dimension is $\mathbb{R}^{T \times d_m}$.

5. **Final Linear Projection:** The concatenated matrix $Z_c$ is passed through a final linear projection layer with a weight matrix $W^C \in \mathbb{R}^{d_m \times d_m}$ to produce the final output of the multi-head attention block.

$$\text{MultiHead}(X) = Z_c \cdot W^C$$

This final output is a matrix in $\mathbb{R}^{T \times d_m}$, the same dimension as the input $X$.

The entire process can be visualized with the following flowchart:

```
graph TD
    subgraph Multi-Head Attention Block
        X["Input (X)"] --> P1["Project to Q1, K1, V1<br/>(Head 1)"]
        X --> P2["Project to Q2, K2, V2<br/>(Head 2)"]
        X --> P_h["...<br/>(Head h)"]

        P1 --> A1["Attention(Q1, K1, V1)"] --> Z1["Output Z1"]
        P2 --> A2["Attention(Q2, K2, V2)"] --> Z2["Output Z2"]
        P_h --> A_h["..."] --> Z_h["Output Zh"]

        subgraph Combine
            Z1 --> C["Concatenate"]
            Z2 --> C
            Z_h --> C
        end

        C --> LP["Final Linear Projection<br/>(W<sup>C</sup>)"] --> Out["Output (Z)"]
    end
```

*Caption: Flowchart of the Multi-Head Attention mechanism, showing parallel attention computations followed by concatenation and a final linear projection.*

---

# Fully Connected and Output Layers

## Position-wise Fully Connected Layer (FCL)

**(21:20 - 28:25)**

After the multi-head attention sub-layer (and a residual connection/layer normalization step, not detailed in this segment), the Transformer architecture includes a **Fully Connected Layer (FCL)**, also known as a Feed-Forward Network (FFN).

- **Key Property:** This FCL is applied **position-wise**. This means the same network (with the same weights) is applied independently to each token's representation vector. It does not mix information across token positions; that is the job of the attention layer.
- **Architecture:** It typically consists of two linear transformations with a non-linear activation function in between.
- **Formula:** For the representation $z_j$ of the j-th token:

$$\text{FCL}(z_j) = \left[\sigma(z_j W_1 + b_1)\right] W_2 + b_2$$

  - $z_j \in \mathbb{R}^{d_m}$ is the input vector for the j-th position.
  - $W_1, b_1, W_2, b_2$ are learnable parameters.
  - $\sigma$ is a non-linear activation function, commonly **ReLU**.
- **Purpose:** This layer introduces non-linearity and allows the model to learn more complex transformations of the token representations.

### Output Layer

**(29:05 - 32:28)**

The final step in a decoder-only Transformer is to convert the model's final representation of each token into a probability distribution over the entire vocabulary.

1. **Final Representation:** After passing through all the Transformer blocks, we have a final representation matrix, let's call it $\tilde{Z} \in \mathbb{R}^{T \times d_m}$. Each row $\tilde{z}_j$ is the final vector for the j-th token.
2. **Linear Projection:** This representation is projected into a space whose dimension is the size of the vocabulary, $|V|$. This is done with a final linear layer.

   $$\text{logits}_j = \tilde{z}_j W_p + b_p$$

   Here, $W_p \in \mathbb{R}^{d_m \times |V|}$. The result, `logits_j`, is a vector of raw scores (logits) for each word in the vocabulary.
3. **Softmax:** The softmax function is applied to the logits to convert them into a valid probability distribution.

   $$\hat{y}_j = \text{softmax}(\text{logits}_j)$$

   The vector $\hat{y}_j \in \mathbb{R}^{|V|}$ contains the probabilities for the next token at position `j+1`, and its elements sum to 1.

---

## Self-Assessment for This Video

1. **Question:** Why is the standard attention mechanism unsuitable for autoregressive language modeling? How does causal masking solve this issue?
2. **Question:** Write down the mathematical formula for the causal mask matrix $M$ for a sequence of length 3. Explain what happens when this mask is added to the attention scores before the softmax function.
3. **Question:** What is the primary motivation for using Multi-Head Attention instead of a single attention mechanism?
4. **Question:** If a Transformer model has a model dimension $d_m = 512$ and uses $h = 8$ attention heads, what are the typical dimensions for the key ($d_k$) and value ($d_v$) vectors for each head?
5. **Question:** Describe the two main steps that occur *after* the individual attention heads have computed their output matrices $(Z_1, Z_2, \ldots, Z_h)$ in a multi-head attention block.
6. **Question:** What does it mean for the Fully Connected Layer in a Transformer to be "position-wise"? How does its function differ from the attention layer's function?
7. **Question:** What is the purpose of the final output layer in a decoder-only Transformer? What function is used to ensure its output is a valid probability distribution?

---

## Key Takeaways from This Video

- **Causality is Key for Autoregressive Models:** To generate sequences, a model must not see future information. Causal masking enforces this by setting future attention scores to $-\infty$ before the softmax, effectively zeroing them out.

- **Multi-Head Attention Captures Diverse Features:** By using multiple parallel attention "heads," the model can learn different types of relationships (syntactic, semantic, etc.) within the data simultaneously, leading to a richer final representation.
- **Architecture is a Stack of Operations:** The Transformer decoder block is a sequence of well-defined operations: (Masked) Multi-Head Attention, followed by a Position-wise Fully Connected Network. These blocks are stacked to build deep models.
- **Final Output is a Probability Distribution:** The model's final task is to predict the next token. This is achieved by a linear layer that projects the final representation to the vocabulary size, followed by a softmax function to create a probability distribution.