Assignment 2

Tian Yu

Feb 5, 2026

Timeline

Tutorial 1 & A2 Release	Feb 6, 2026
Tutorial 2	Feb 13, 2026
Office Hour	Feb 27, 2026
A2 Deadline	Mar 5, 2026

Tut1 Outline

- What is Neural Machine Translation
- The Transformer Architecture (Section 1-2 in the handout)

The Neural Machine Translation

Neural Machine Translation

- Task is to automatically translate a sentence in a **source** language to a sentence in a **target** language

- $X = x_1, x_2, \ldots, x_S$ be in source language of S tokens
- $Y = y_1, y_2, \ldots, y_T$ be in target language of T tokens

Neural Machine Translation - Goal

- Assume we have the oracle probability P*, the goal of a **translation problem** is to find

$$Y^* = Argmax_Y \, P^*(Y|X)$$

- In SMT, do the approximation by

$$P^*(Y|X) = \frac{P^*(X|Y)P^*(Y)}{P^*(X)} \propto P_{TM}(X|Y)P_{LM}(Y)$$

Parallel Corpus

Target Corpus

- In NMT, we train the network to approximate

$$P^*(Y|X) \approx P_\theta(Y|X)$$

Neural Machine Translation – Achieve the goal

- We **assume** the training set is a good representation of the real-world translation distribution
- The best we can do is to have $P_\theta(Y|X)$ return a high probability for reference translation in the training corpus, ie: we choose theta that **maximizes the likelihood** of the data under the model
- That's why we can use **Cross-Entropy Loss**

Neural Machine Translation – Side Note

- Start of Sequence token (SOS) ; End of Sequence token (EOS)
- Both Y, and X will start with <SOS>, and end by <EOS> in the assignment implementation

The Transformer Architecture

An overview

$P(y_i \mid y_{<i}, X)$

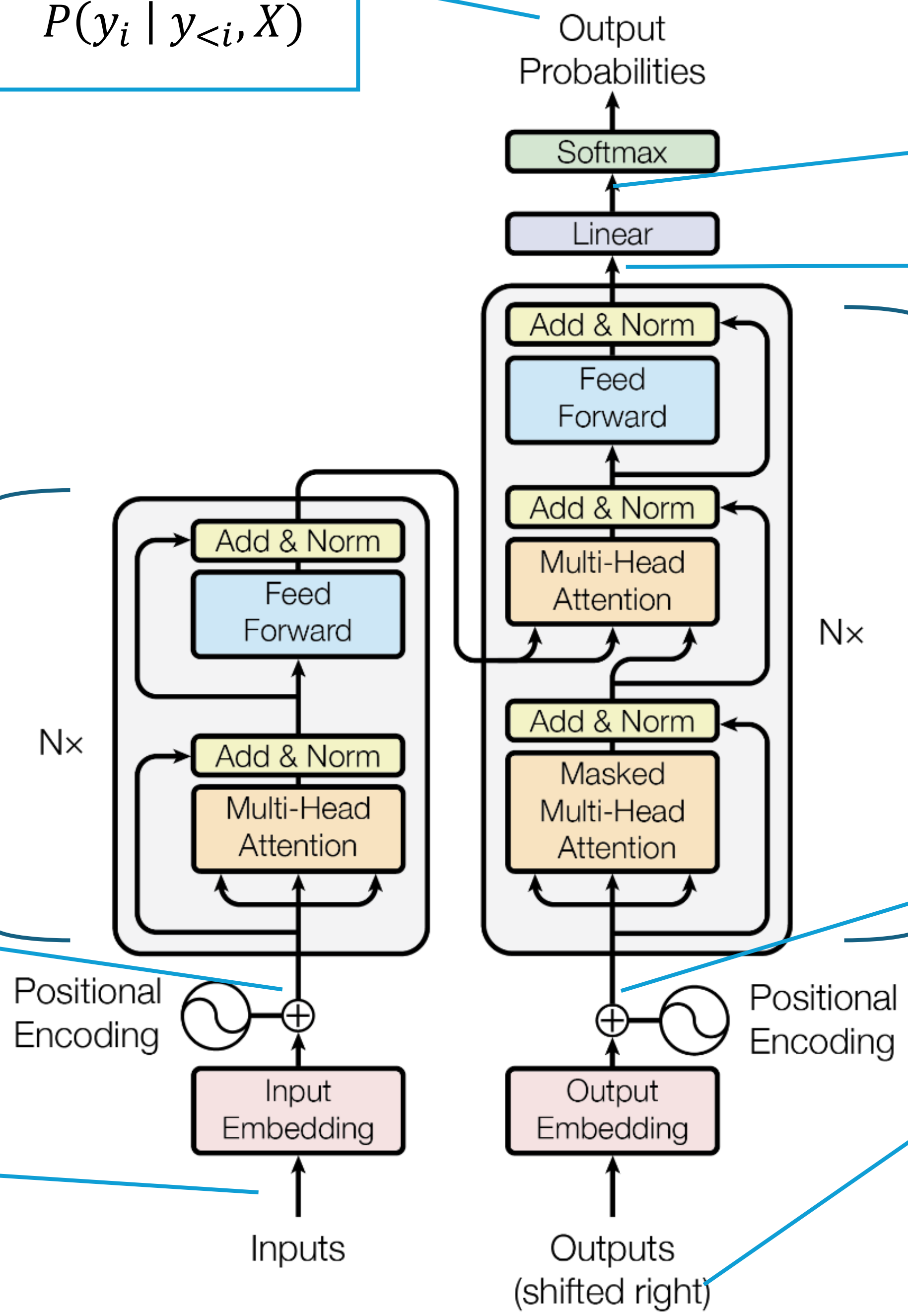
[B, T, V]
V is the vocabulary size

[B, T, D]

Just for intuition, we will cover masking later

Encoder

Decoder



[B, T, D]

[B, S, D], where D represents the learnt latent representation of the token

[B, T]
$Y = SOS, y_1, \ldots, y_{T-1}$
(training with teacher forcing)

Batch of padded sequence
$X_b = SOS, x_1, x_2, \ldots, x_i, EOS, PAD \ldots$
[B, S], where x_i is the token idx, S is the padded length

Image Source: https://arxiv.org/pdf/1706.03762

The building blocks (A2 – Section 1)

- Layer Norm
- Feed Forward Layer
- Multi-Head Attention

Layer Norm

LayerNorm The normalization layer computes the following. Given an input representation $\mathbf{h}$, the normalization layer computes its mean μ and the standard deviation σ. Then, it outputs the normalized features.

$$\mathbf{h} \leftarrow \frac{\gamma(\mathbf{h}-\mu)}{\sigma+\varepsilon}+\beta \tag{1}$$

Normalize across the token's latent feature (hidden) dimension, **independently for each token**

Feed Forward Layer

FeedForwardLayer The feed-forward layer is a two-layer fully connected feed-forward network. As shown in the following equation, the input representation $\mathbf{h}$ is fed through two layers of fully connected layers. Dropout is applied after each layer, and ReLU is the activation function.

$$\begin{aligned} \mathbf{h} &\leftarrow \text{dropout}(\text{ReLU}(\mathbf{W}_1\mathbf{h} + \mathbf{b}_1)) \\ \mathbf{h} &\leftarrow \text{dropout}(\mathbf{W}_2\mathbf{h} + \mathbf{b}_2) \end{aligned} \tag{2}$$

The Feed Forward Layer is applied to each hidden state (d_model), independently to each token

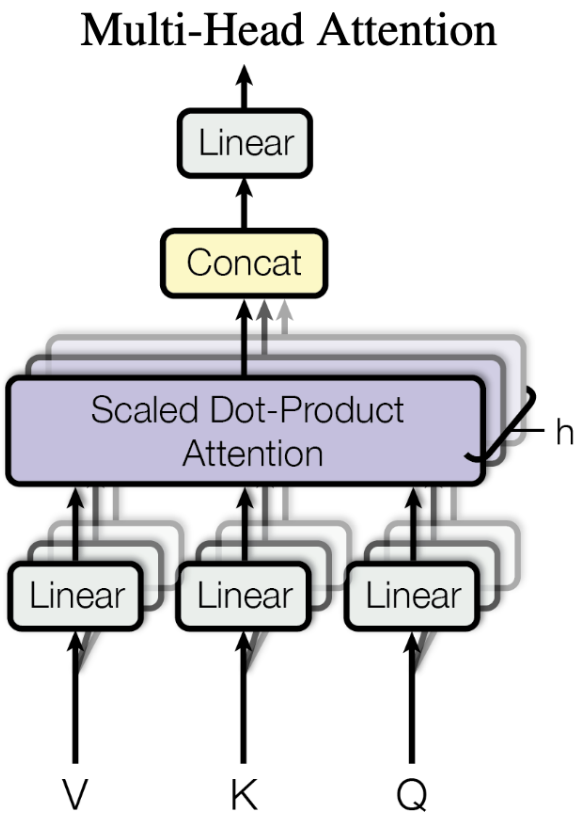
The Attention Mechanism

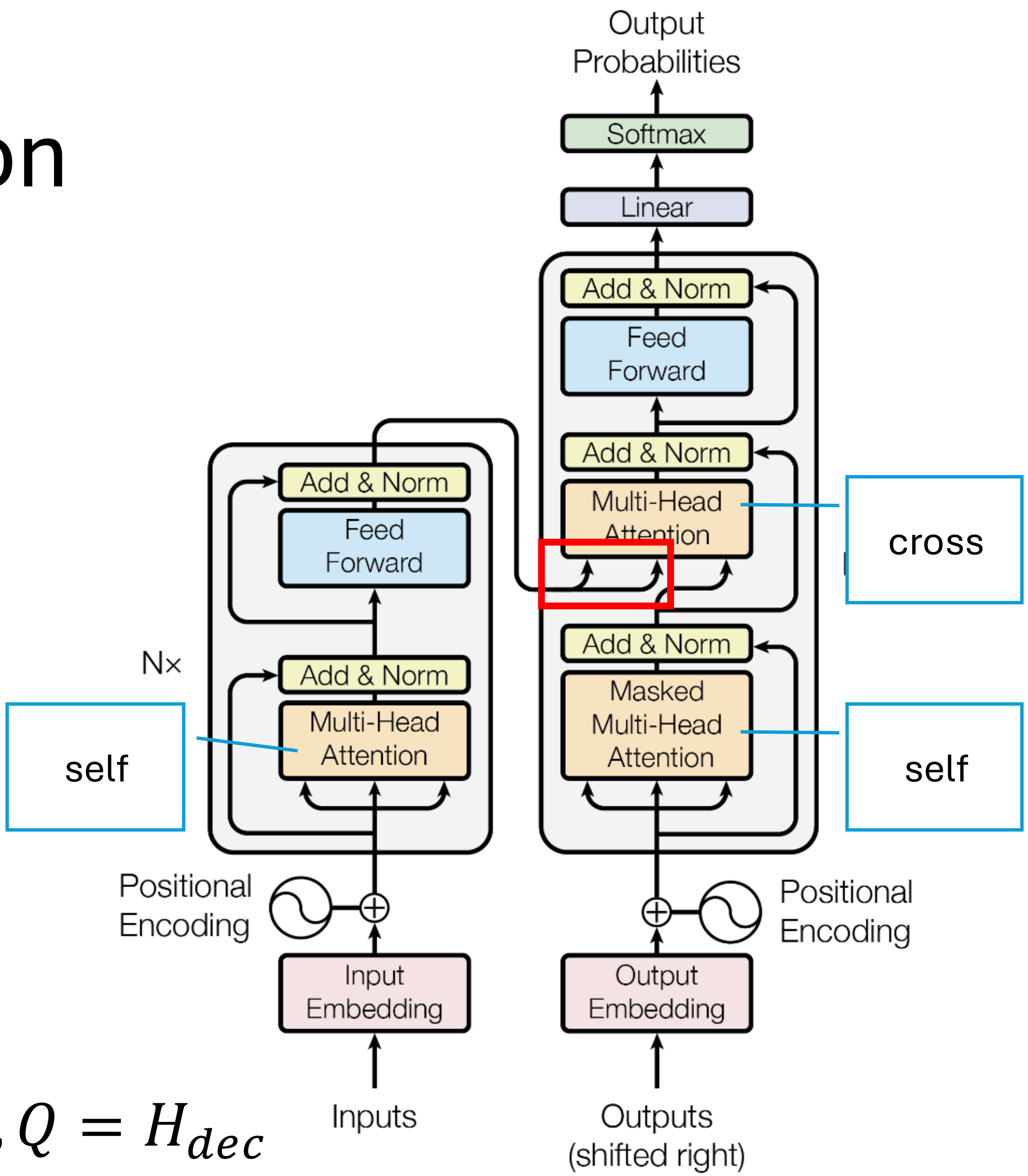
- What is the problem?
  - **Run** the code
  - **Run** a company
  - **Run** fast

The Attention Mechanism

- Q: Query, K: Key, V: Value
- Q, K, V have a linear projection applied to them
- For multi-head attention, the Q, K, V are partitioned into h heads via reshaping of tensors
- Hint: Take a look at the docstring of MultiHeadAttention, for the input shape of the methods

Self / Cross Attention





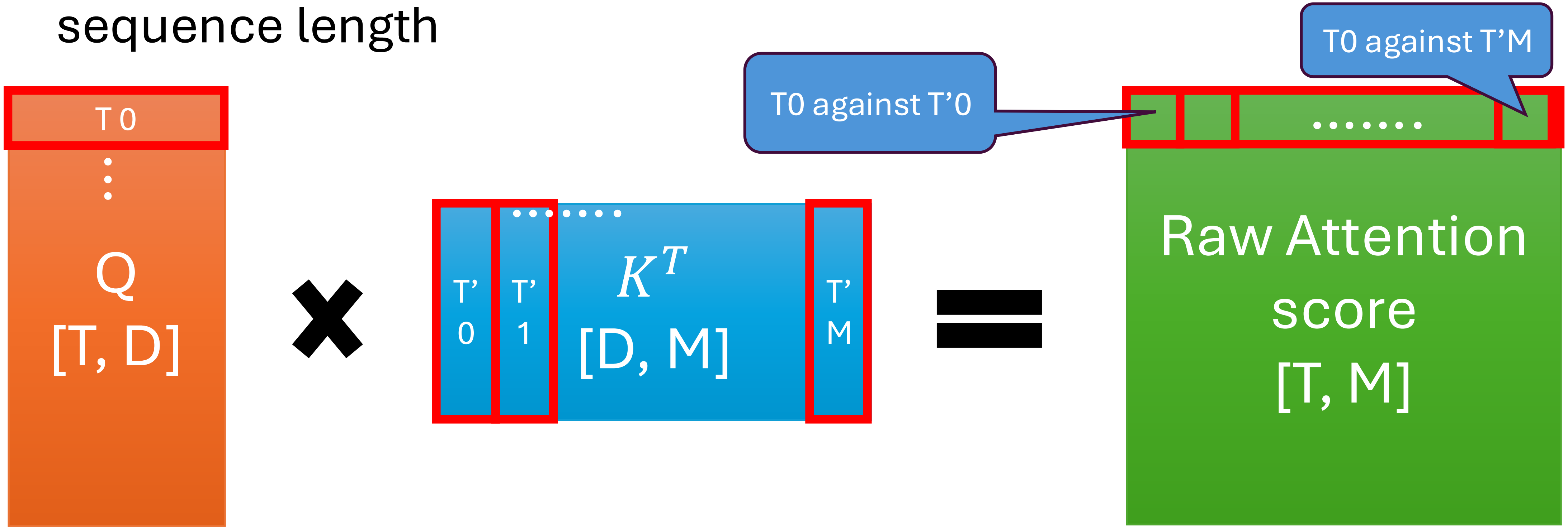
For self-attention, $V = K = Q = H$

For cross attention, $V = K = H_{enc}$, $Q = H_{dec}$

Image Source: https://arxiv.org/pdf/1706.03762

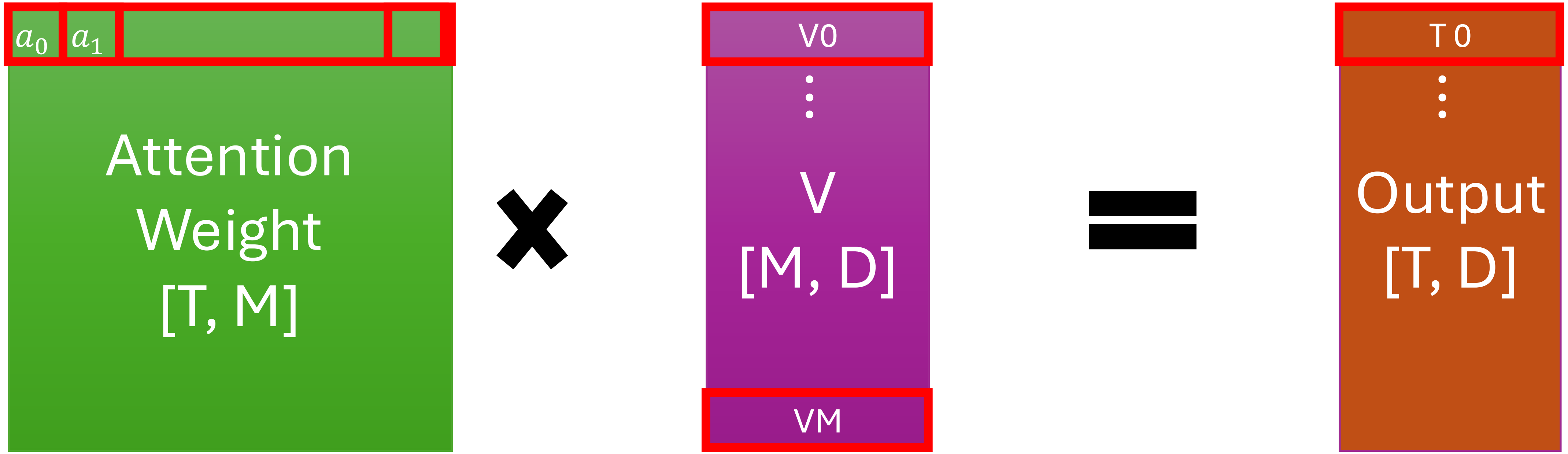
The Attention Mechanism – B=1, H=1

- $dropout\left(softmax\left(\frac{Q\,K^T}{\sqrt{d_k}}\right)\right)V \quad (d_k = d_{head}\ below)$
- T is the query sequence length, D is the latent dimension of a token **(d_head = d_model when H=1),** M is the key / value sequence length



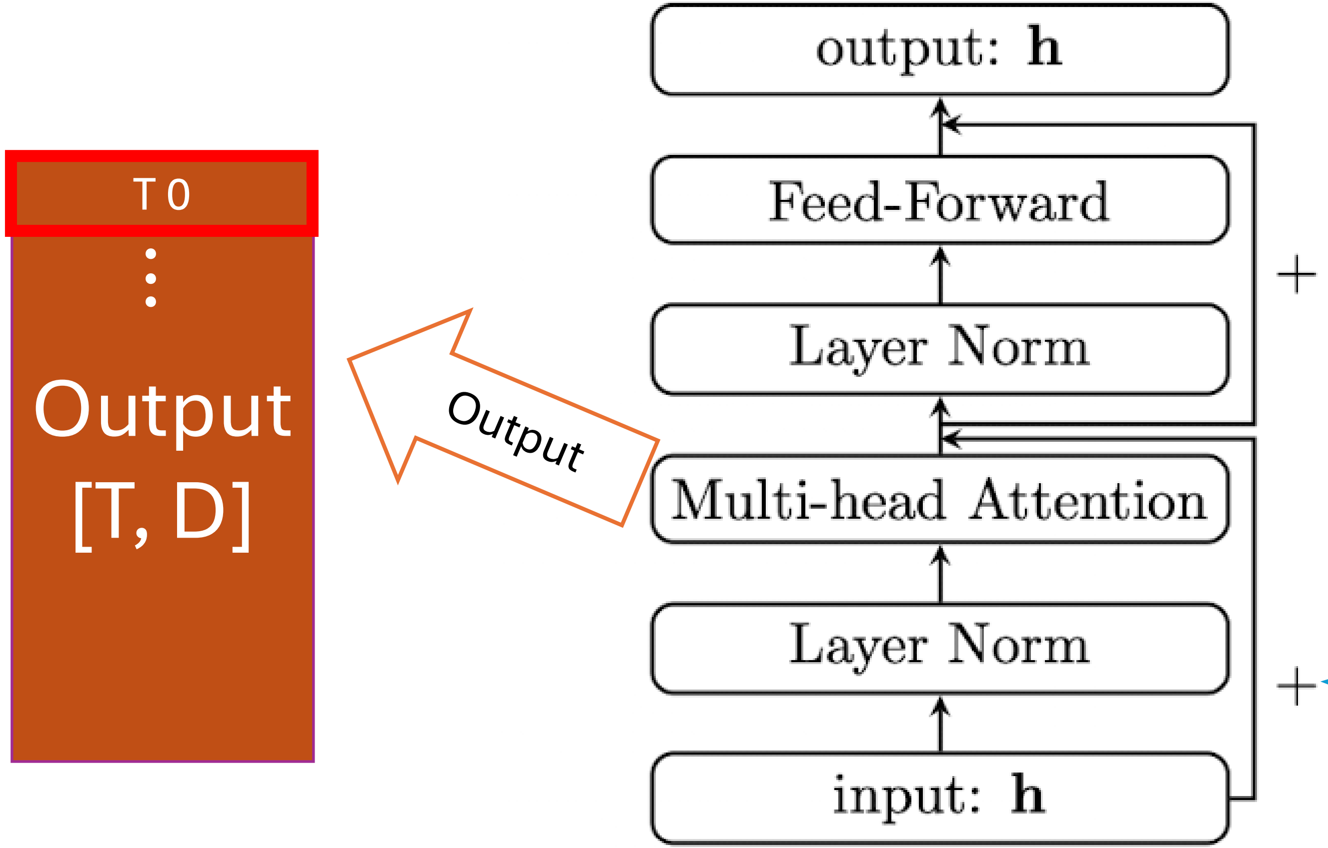
The Attention Mechanism – B=1, H=1

- $dropout\left(softmax\left(\frac{Q\,K^T}{\sqrt{d_k}}\right)\right)V$
- T is the query sequence length, D is the latent dimension of a token **(d_head = d_model when H=1),** M is the key / value sequence length



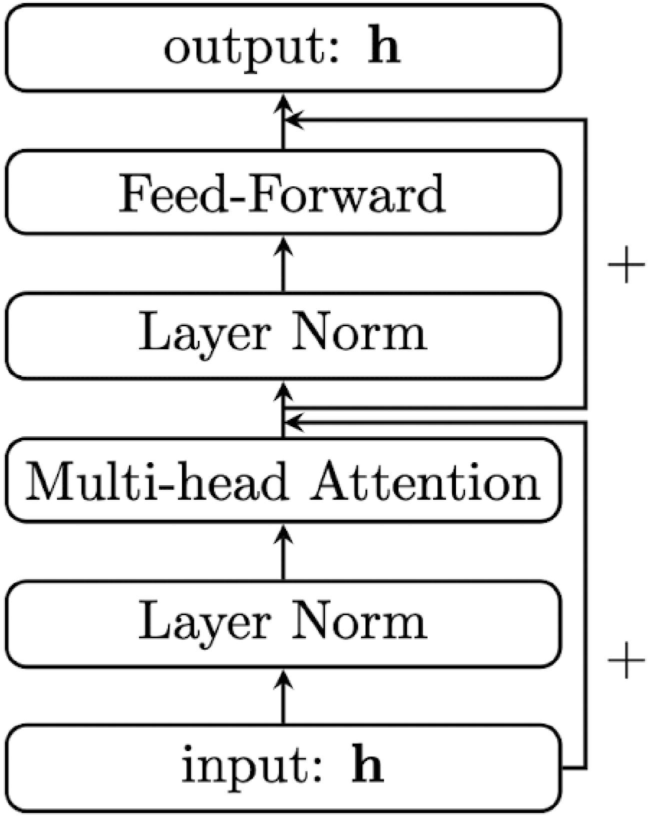
The Attention Mechanism- Intuition

Run the code
Run a company
Run fast

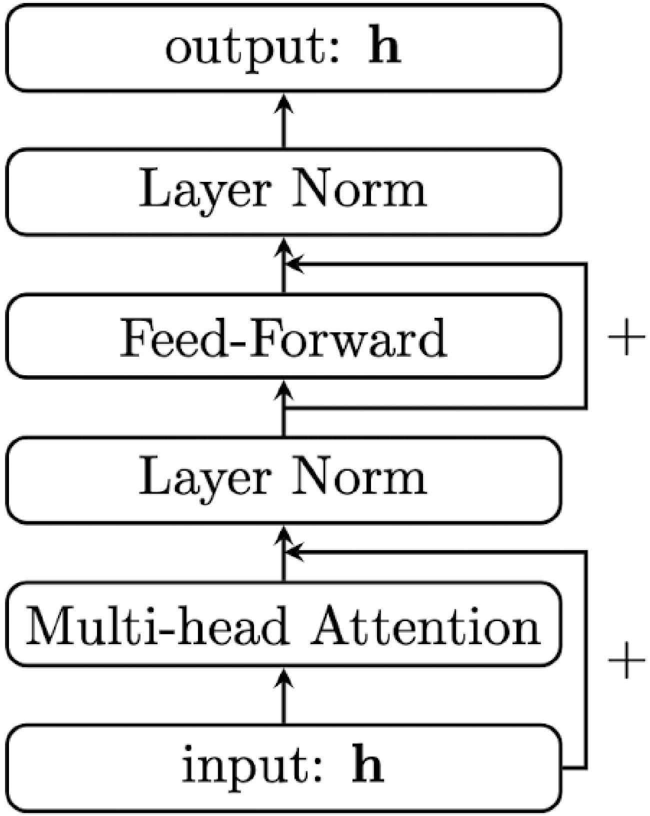


Residual Connection means:
the attention output acts as an update (**shift**) in the latent space on the hidden state

Encoder



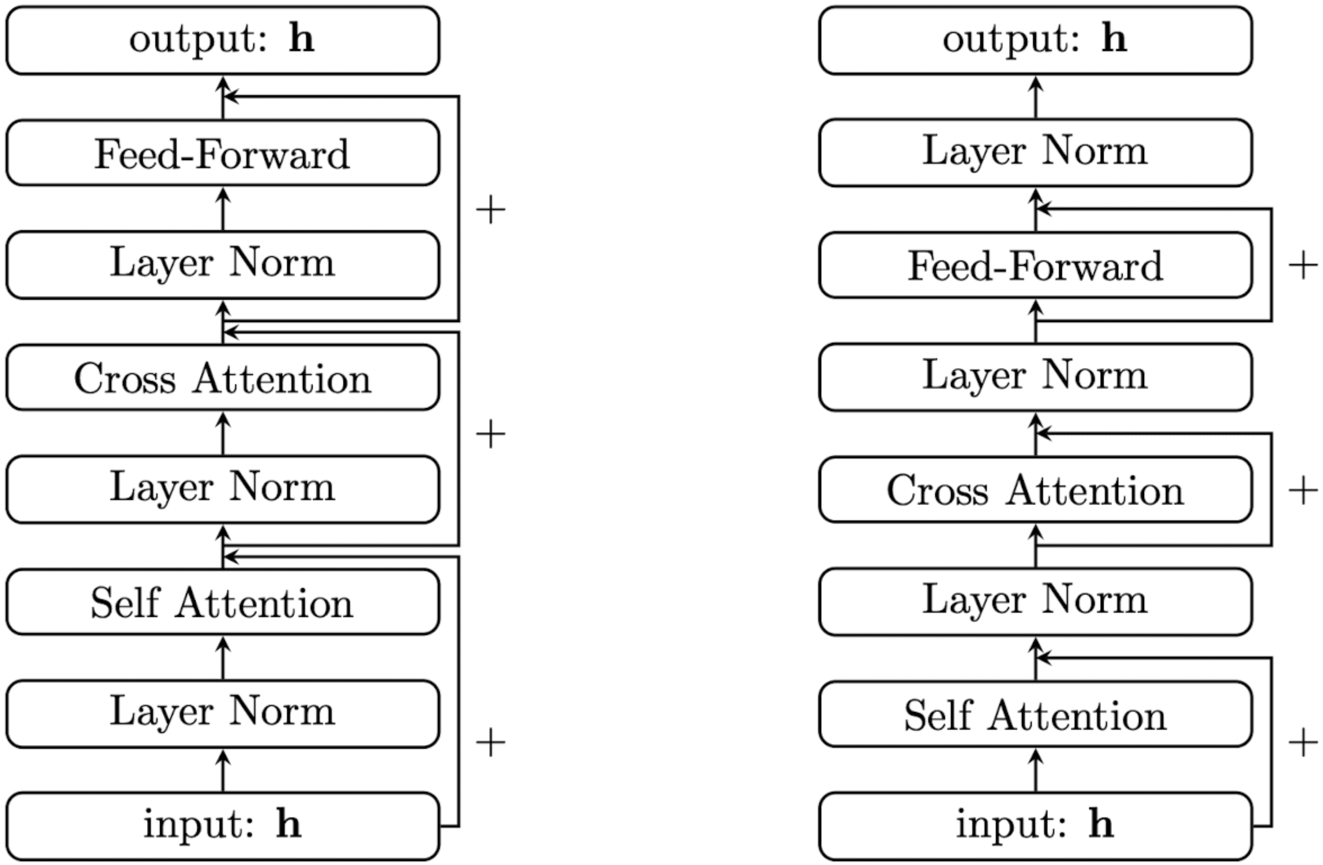
(a) Pre-layer normalization for encoders.



(b) Post-layer normalization for encoders.

Figure 1: Two types of `TransformerEncoderLayer`.

Decoder



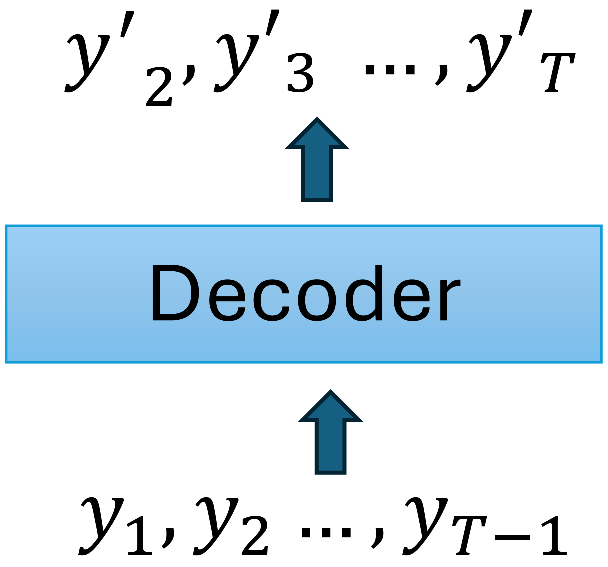
(a) Pre-layer normalization for decoders. (b) Post-layer normalization for decoders.

Figure 2: Two types of `TransformerDecoderLayer`.

Masked Attention

- Our data is batched sequences, which means they might be in **different lengths**
- They are padded to length in the batch with a special pad token (‘pad_idx’ in the code)
- We need to have a pad mask so that these tokens won’t attend to other tokens (method ‘create_pad_mask’), or in other words, the attention score should be 0
- How can we make the **softmax output** become **0** for certain positions?

Causal Mask and Teacher Forcing



- In the decoder, we want to train in parallel
- But when generating y_2, the attention mechanism shouldn't see $y_3, \ldots, y_{T-1}$ (the future)
- This can be resolved by having a triangular causal mask

Assignment2 Tips

- Match tensor shapes like Lego and always comment and print out the shapes of any given tensor.
- A lot of work:
    - Don’t be overwhelmed, it’s not difficult, but there are a lot of things you need to consider at once, which can be challenging
    - Start early! You don’t want to compete for GPUs in the end

Next Tutorial

- Decoding
- Training and Testing