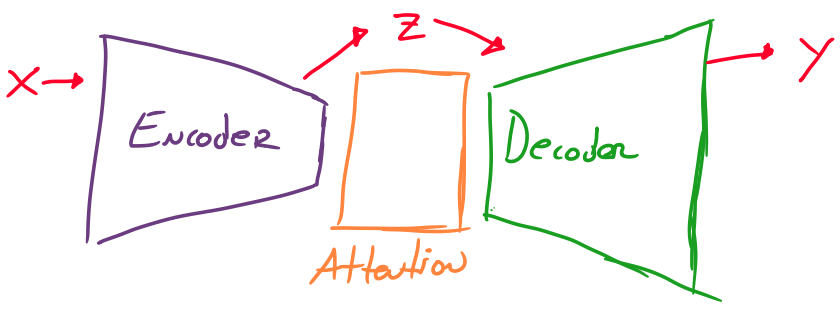
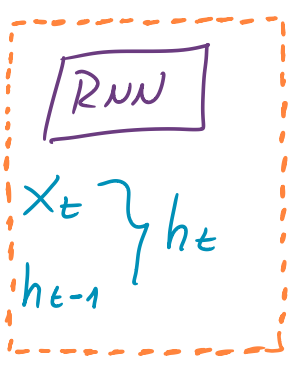
Attention is all you need



the best performing models connect the encoder and decoder through an attention mechanism.

transformer = Based solely on attention mechanisms dispensing with recurrence and convolutions.



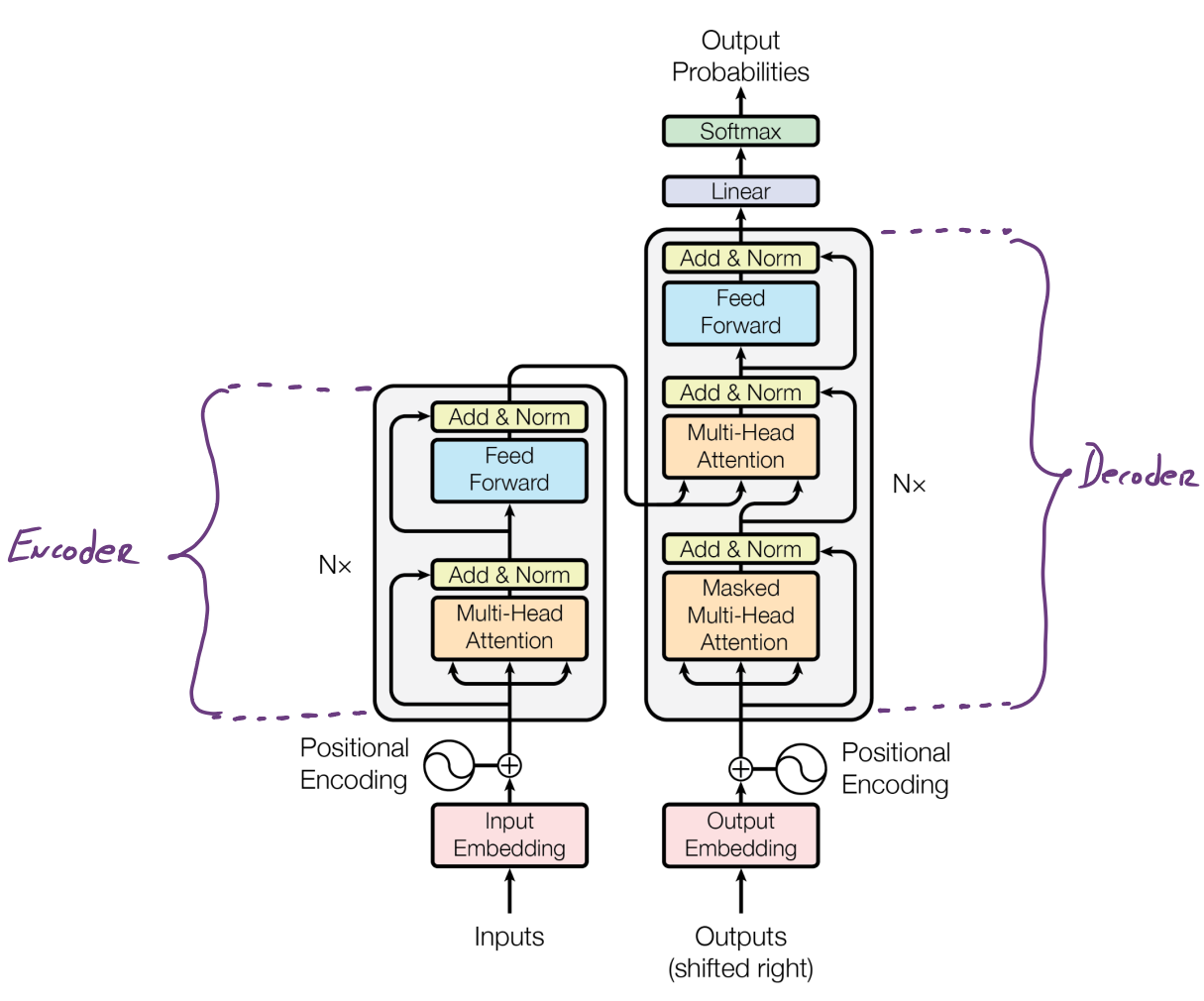
→ this structure precludes parallelization.

→ Have a bigger requirement of memory.

TRANSFORMERS allow more parallelization.

Self attention: Compute a representation of one sequence.

ARCHITECTURE



ENCODER: - Stack of N=6 identical layers. Each layer has two sub-layers.

First sublayer = Multi Head Attention Mechanism

Second sublayer = Simple position-wise fully connected feed forward network.

After every sublayer there is a residual connection followed by normalization.

the output of every sub-layer is: $\text{LayerNorm}(x + \text{Sublayer}(x))$

DECODER: - Stack of N=6 identical layer.

- the decoder has a third sublayer that performs as a multihead attention over the output of the encoder.

Similar to the encoder we apply residual connections around each of the sublayer followed by Layer Normalization

ATTENTION: - An attention function can be described as mapping a query and a set of key-value pairs to an output. the output is computed as weighted sum.

SCALED DOT-PRODUCT ATTENTION

$$\text{Attention}(Q,K,V) = \underbrace{\text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right)}_{A'} \cdot V$$

$$Q \in \mathbb{R}^{m_Q \times d_k},\ K \in \mathbb{R}^{m_K \times d_k}$$

$$Q \cdot K^T = \mathbb{R}^{m_Q \times d_k} \cdot \mathbb{R}^{d_k \times m_K} = \mathbb{R}^{m_Q \times m_K}$$

Softmax $\dfrac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$ by rows.

$A' = \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}$ $\Sigma = 1$

MULTIHEAD ATTENTION

$$\text{Multihead}(Q,K,V) = \text{Concat}(\text{head}_1, \dots, \text{head}_n) \cdot W^O$$

$$\text{head}_i = \text{Attention}(Q \cdot W_i^Q, K \cdot W_i^K, V \cdot W_i^V)$$

Projections are parameter matrix $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$,

$W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$

$W_O \in \mathbb{R}^{h \cdot d_v \times d_{model}}$

POSITION-WISE FEED FORWARD NETWORKS

$$FFN(x) = W_2 \cdot (0, x \cdot W_1 + b_1) + b_2$$

POSITIONAL ENCODING

$$PE(pos, 2i) = \sin\left(\frac{pos}{10.000^{\frac{2i}{d_{model}}}}\right)$$

$$PE(pos, 2i+1) = \cos\left(\frac{pos}{10.000^{\frac{2i}{d_{model}}}}\right)$$