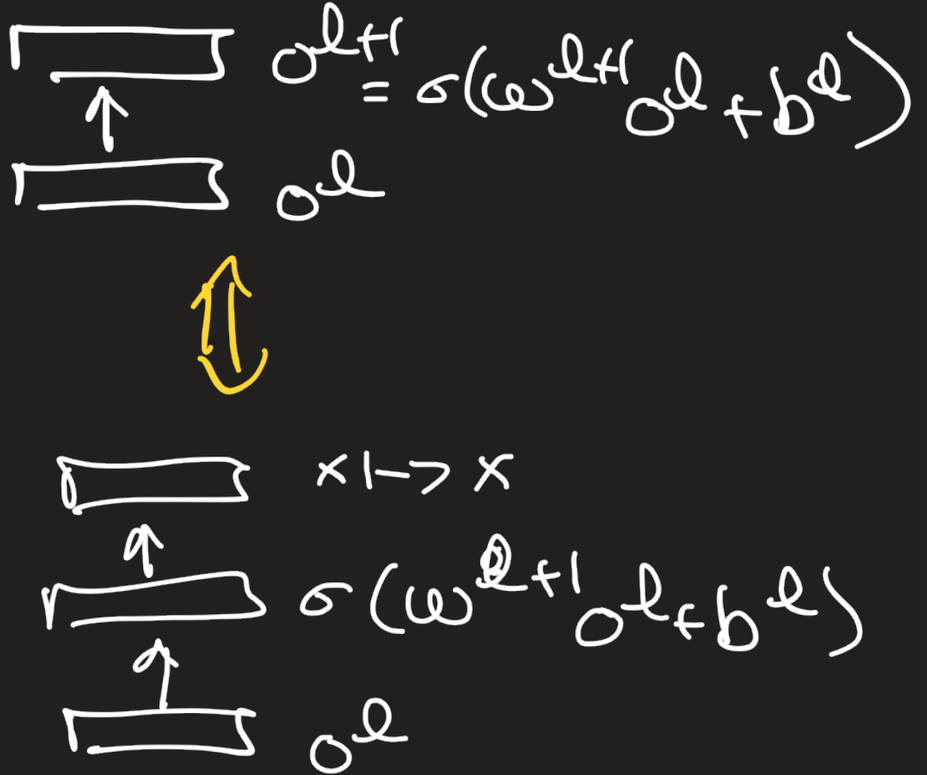
ML and Optim Lec 26

- Transformer Seq2Seq Architecture
- Skip-connections
- Layer-normalization

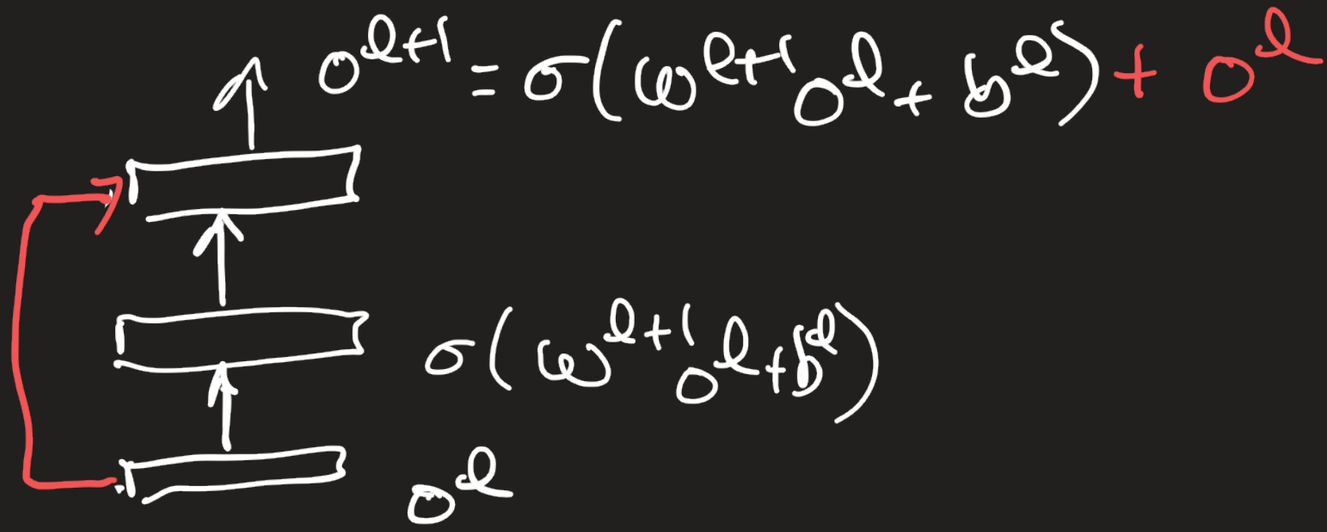
Skip-connections

Skip connections are connections between layers l and layers higher than $l+1$, e.g. DenseNet, where all layers are connected.

An efficient & popular type of skip-connection is a residual connection.



Residual connection



Layer-normalization

Like batch normalization this helps stabilize training by ensuring the distribution of the outputs of layer l doesn't vary much over time, so layer $l+1$ doesn't have to waste optimization adapting.

$$\text{LayerNorm}(o) = \gamma \cdot \frac{o - \bar{o}}{\sigma} + \beta,$$

where $\gamma \in \mathbb{R}^{n_l}$ and $\beta \in \mathbb{R}^{n_l}$ are the parameters of the layer normalization layers and

$$\bar{o} = \frac{1}{n_l} \sum_{i=1}^{n_l} o_i, \quad \sigma^2 = \frac{1}{n_l} \sum_{i=1}^{n_l} (o_i - \bar{o})^2$$

Batch normalization

output neuron index

index into minibatch

O_1

O_m

operates on minibatches

Batch normalization computes per neuron means and stdevs over the minibatch

then batch normalization of an o_i is given by

$$\gamma \cdot \left(\frac{o_i - \mu}{\sigma}\right) + \beta$$

Layer normalization

operates on a single output

$\bar{o}$ = mean of o

σ = stdv of o

then layer normalization of o is given by

$$\gamma \cdot \left(\frac{o - \bar{o}}{\sigma}\right) + \beta$$

Expressions for Attention

Self-attention:

then $\tilde{\alpha}_i^T = \begin{bmatrix} q_i^T k_1 & \cdots & q_i^T k_m \end{bmatrix}$

is the logits vector (row vector) for the i^{th} attention vector

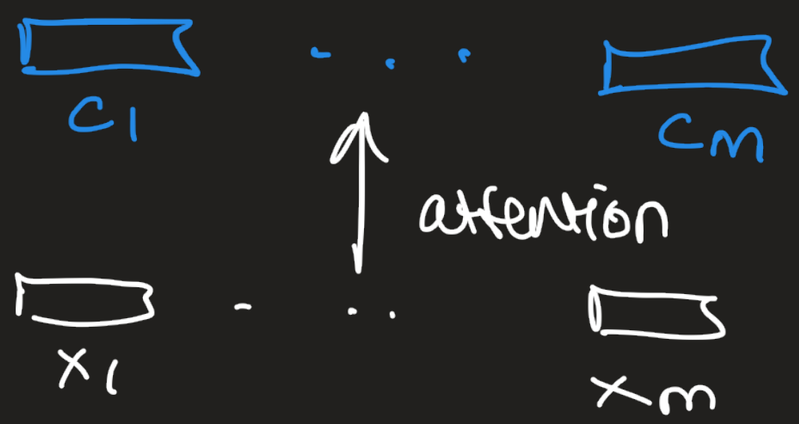
$$\alpha_i^T = \text{softmax}(\tilde{\alpha}_i^T)$$

is my attention vector for the i^{th} token is

then our contextual embedding

$$c_i^T = \sum_{j=1}^{m} (\alpha_i)_j \, v_j^T$$

let $X = \begin{bmatrix} x_1^T \\ \vdots \\ x_m^T \end{bmatrix} \in \mathbb{R}^{m \times d}$



and $C = \begin{bmatrix} c_1^T \\ \vdots \\ c_m^T \end{bmatrix} \in \mathbb{R}^{m \times d_v}$

Then

$$C = \mathrm{Aff}(X, X)$$

where the attention layer has parameters $\omega_Q, \omega_K, \omega_V$

To compute C, write

$$K = X\omega_K = \underbrace{\begin{bmatrix} x_1^T\omega_K \\ \vdots \\ x_m^T\omega_K \end{bmatrix}}_{d_K\text{-dim}} \qquad Q = X\omega_Q = \underbrace{\begin{bmatrix} x_1^T\omega_Q \\ \vdots \\ x_m^T\omega_Q \end{bmatrix}}_{d_K\text{-dim}}$$

$$V = X\omega_V = \begin{bmatrix} x_1^T\omega_V \\ \vdots \\ x_m^T\omega_V \end{bmatrix}$$

, then note that

$$C = \begin{bmatrix} c_1^T \\ \vdots \\ c_m^T \end{bmatrix} = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Similarly, a general attention layer takes two sequences $X \in \mathbb{R}^{m \times d}$ $X' \in \mathbb{R}^{T \times d}$

so a general attention layer computes

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

where

$$K = X\omega_K$$

$$V = X\omega_V$$

$$Q = X'\omega_Q$$

Transformer Architecture (image upcoming)

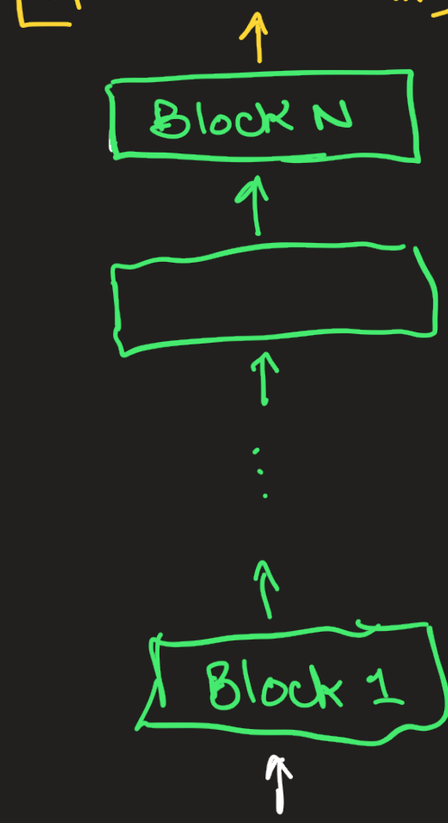
Encoder:

- use multiple layers of multi-head self-attention with residual connections, layer-normalization, and dense nonlinear transforms

Decoder

- use multiple layers of multi-head self-attention, multi-head general attention, with residual connections, layer normalizations, and dense nonlinear transforms.

Encoder:



where each embedding block has its own parameters and computes:

Given input $X = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} \in \mathbb{R}^{m \times d}$,

i) $\Psi = \text{LayerNorm}(X + \text{Mult-head Att}(X, X))$

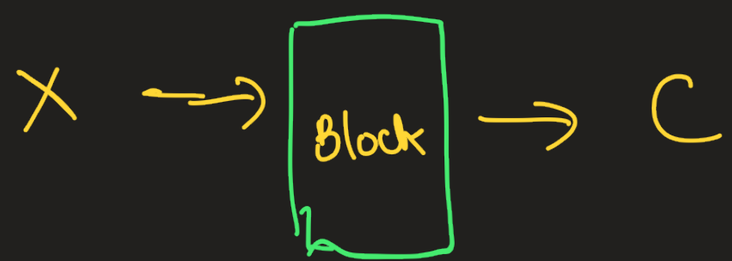
recall $\text{Mult-head Att}(X, X) =$

$[\text{Att}_{h_1}(X, X) | \cdots | \text{Att}_h(X, X)] W_0$

concatenates h different attention outputs together then takes linear combination to learn a final d-dim representation

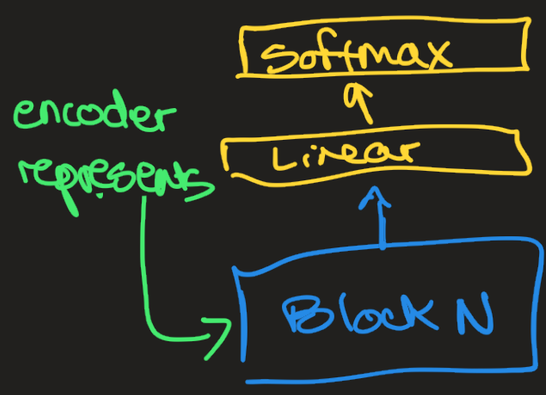
ii) $C = \text{LayerNorm}(\Psi + \text{FF}(\Psi))$

where FF applies the same feedforward dense + ReLU layer to each row of Ψ



Decoder

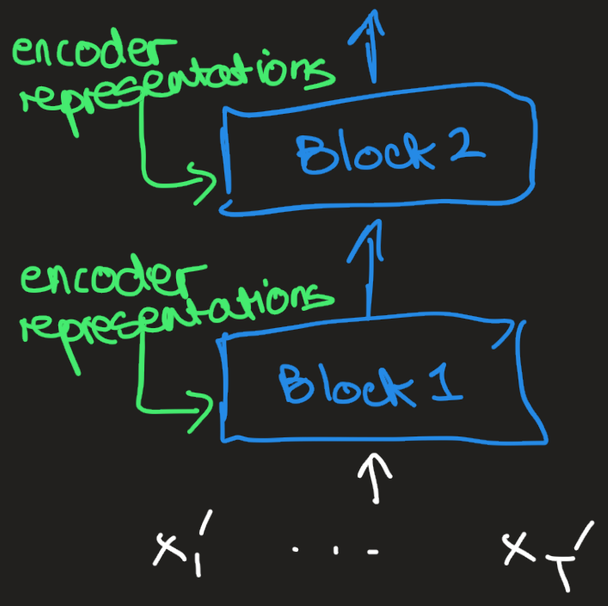
During training, use teacher forcing



each block has its own parameters and computes, given inputs $X' = \begin{bmatrix} x_1' \\ \vdots \\ x_T' \end{bmatrix}$

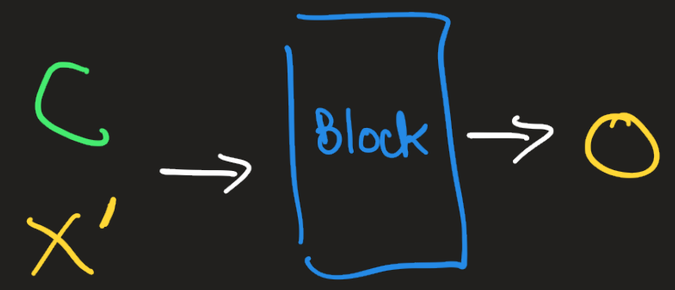
and the final representations from the encoder of the input sequence $C = \begin{bmatrix} c_1 \\ \vdots \\ c_m \end{bmatrix}$:

(i) $Y = \text{LayerNorm}(X' + \text{Multihead Att}(X', X'))$



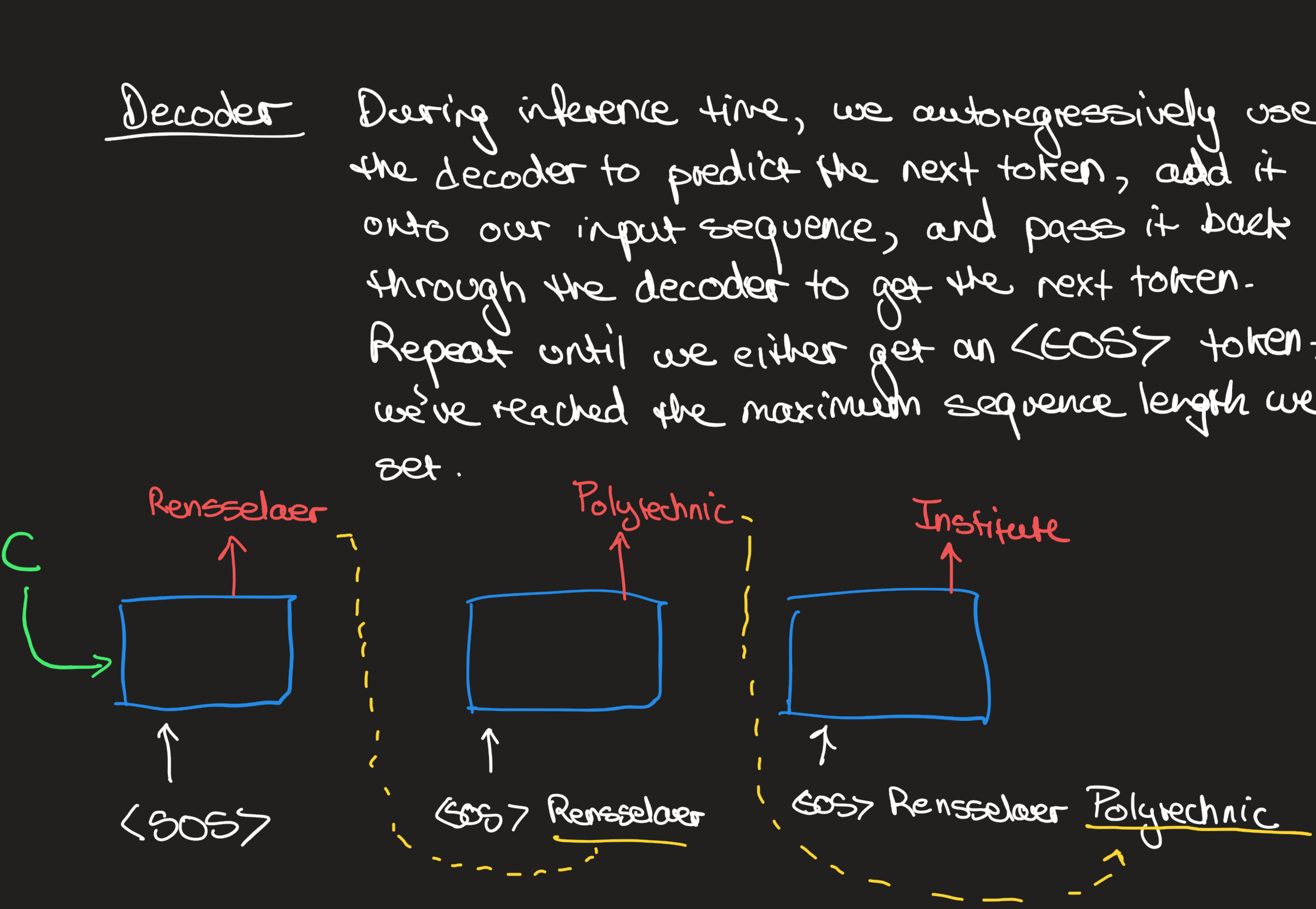
(ii) $Z = \text{LayerNorm}(Y + \text{Multi-head Att}(C, Y))$

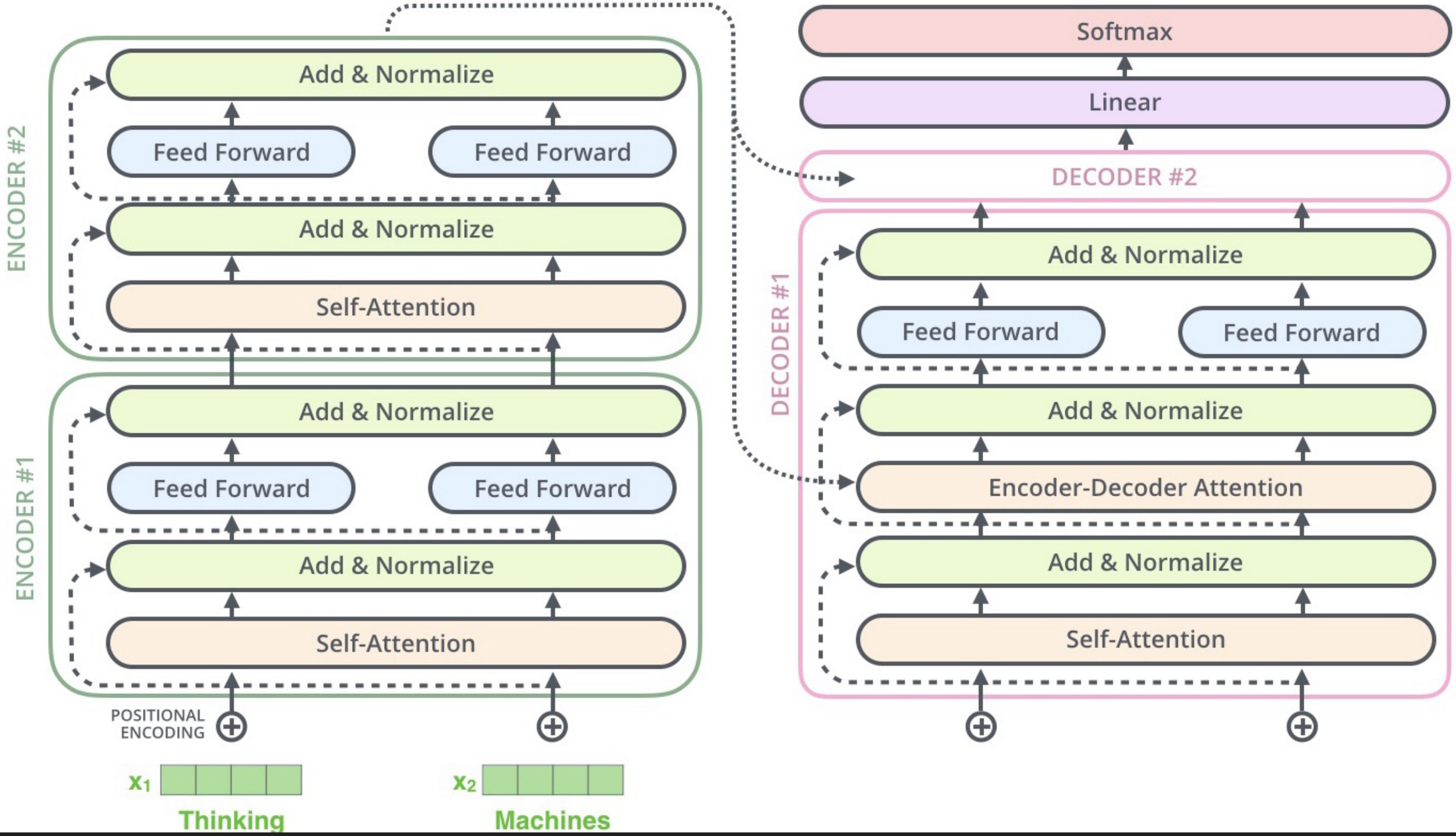
(iii) $O = \text{LayerNorm}(Z + FF(Z))$



In the self-attention layers, the attention is masked: we set $(\alpha_i)_j = 0$ for $j > i$ to ensure we only predict using historical info.

Decoder During inference time, we autoregressively use the decoder to predict the next token, add it onto our input sequence, and pass it back through the decoder to get the next token. Repeat until we either get an <EOS> token, or we've reached the maximum sequence length we set.




Softmax
Linear
DECODER #2
DECODER #1
Add & Normalize
Feed Forward
Feed Forward
Add & Normalize
Encoder-Decoder Attention
Add & Normalize
Self-Attention
ENCODER #2
ENCODER #1
Add & Normalize
Feed Forward
Feed Forward
Add & Normalize
Self-Attention
POSITIONAL ENCODING
x1
x2
Thinking
Machines

Positional Encodings

Issue w/ Transformers as is: they don't properly learn positional relations (e.g. that "the cat ate its food" should have a diff contextual rep from "the food ate its cat")

Soln: replace $X \in \mathbb{R}^{n \times d}$ with $X + P$

where $P \in \mathbb{R}^{n \times d}$ encodes the positions 1 through n

$$P_{i,j} = \begin{cases} \sin(i \omega^{j/d}) & \text{if } j \text{ even} \\ \cos(i \omega^{(j-1)/d}) & \text{if } j \text{ odd} \end{cases}$$

(refer to preceeding diagram to see where positional embeddings are added)