Transformer

Transformer Parameter Count

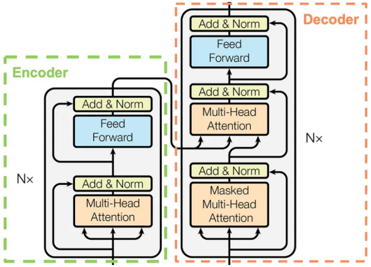
Learning goals

- Understand which components of the Transformer contribute to the parameter count
- Learn how to calculate the total number of the model parameters

WHY DO WE CARE ABOUT THE PARAMETER COUNT?

- Being able to derive the parameter count of a model requires good knowledge of the math behind the model architecture
- So by deriving the parameter count we learn how the model works under the hood
- Since models just stack various buidling blocks on top of eachother, we can reuse the derived parameter count for all sorts of models
- We get an idea about how the parameters scale with increasing model size

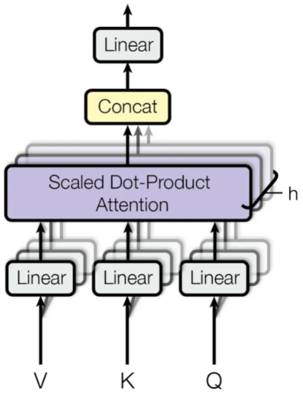
OVERVIEW OVER PARAMETER SOURCES



- **Encoder**
  - Multi-Head Attention
  - 2x Layernorm
  - Feed Forward Network
- **Decoder**
  - 2x Multi-Head Attention
  - 3x Layernorm
  - Feed Forward Network

So all we have to do is to derive the parameter count of each of those components and then sum them up! ▸ Vaswani et al., 2017

MULTI HEAD ATTENTION (1)



- The input token embeddings have shape d_{model} (embedding dimension)
- Project input embeddings into Query, Keys and Values with matrices $\boldsymbol{W_Q}$, $\boldsymbol{W_K}$, $\boldsymbol{W_V}$ for each head (so we have n_{head} (h in the image) of each Matrix)
- Project concatenated output of the Scaled Dot-Product Attention back to d_{model} with $\boldsymbol{W_O}$

To get to the total parameter count we simply have to calculate the number of parameters in all of those Matrices! (for now we also include the bias terms of the linear projections)

MULTI HEAD ATTENTION (2)

- **Project Input Sequence into Query, Key and Value:**
  - Input shape: $seq\_len \times d_{model}$ (seq_len is the number of tokens, where each has the embedding dimension of d_{model})
  - For MHA we project d_{model} to d_{qkv}, with $d_{qkv} = d_{model}/n_{head}$
  - We split up d_{model} into n_{head} heads and need $\boldsymbol{W_Q}$, $\boldsymbol{W_K}$, $\boldsymbol{W_V}$ for each head
  - Each Matrix has shape $d_{model} \times d_{qkv}$

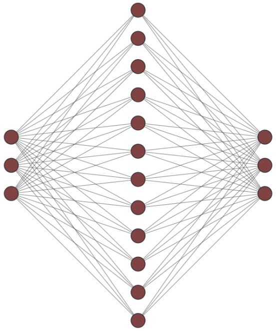
- **Project concatenated output to the MHA output:**
  - We concatenate the n_{head} outputs, where each has shape d_{qkv} and end up with a shape of d_{model} again
  - We then project d_{model} to d_{model} with $\boldsymbol{W_O}$
  - $\boldsymbol{W_O}$ thus has a shape of $d_{model} \times d_{model}$

MULTI HEAD ATTENTION (3)

Now we put both parts together:

$$
\begin{aligned}
N_{attention} &= \underbrace{(d_{model} \times d_{model} + d_{model})}_{W_O} \\
&\quad + \underbrace{(d_{model} \times d_{qkv} + d_{qkv})}_{W_{Q,K,V}} \times \quad \overbrace{n_{heads} \times 3}^{\text{3 matrices for each head}} \\
&= (d_{model} \times d_{model} + d_{model}) + (d_{model} \times d_{model} + d_{model}) \times 3 \\
&= \underbrace{(d_{model}^2 + d_{model}) \times 4}_{\text{The exact formula}} \\
&\approx \underbrace{4 \cdot d_{model}^2}_{\text{Approximation}}
\end{aligned}
$$

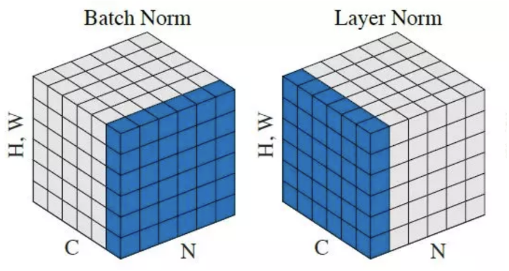
FEED FORWARD NETWORK



- The FFN of the Transformer uses two fully connected layers
- Input and output size is both d_{model}
- The hidden layer size is $4 \times d_{model}$ (*This is a decision by the authors of the paper!)*
- Our weight matrices W_1 and W_2 therefore have shape $d_{model} \times 4 \cdot d_{model}$ and $4 \cdot d_{model} \times d_{model}$ respectively

$$\begin{aligned} N_{FFN} &= (d_{model} \times 4 \cdot d_{model} + 4 \cdot d_{model}) + (4 \cdot d_{model} \times d_{model} + d_{model}) \\ &= 8 \cdot d_{model}^2 + 5 \cdot d_{model} \\ &\approx 8 \cdot d_{model}^2 \end{aligned}$$

LAYER NORM (1)



Instead of calculating mean and standard deviation across the batch we do that across the layer! ▸ Layer Normalization

LAYER NORM (2)

- We estimate μ and σ for each layer (*H denotes the number of hidden neurons in a layer*):

$$\mu = \frac{1}{N}\sum_{i=1}^{H} a_i$$

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{H} (a_i - \mu)^2}$$

- For rescaling the normalizations we need a β and γ parameter for each neuron in the layer
- Since our layer has d_{model} neurons we need $2 \cdot d_{model}$ parameters for the Layer Norm operations

$$N_{Layernorm} = 2 \cdot d_{model}$$

EXACT PARAMETER COUNT

All we have to do now is to sum up the Parameter counts of each component for both the Encoder and Decoder!

$$\begin{aligned} N_{Encoder} &= 1 \cdot N_{attention} + 1 \cdot N_{FFN} + 2 \cdot N_{Layernorm} \\ &= 4 \cdot (d_{model}^2 + d_{model}) + 8 \cdot d_{model}^2 + 5 \cdot d_{model} + 2 \cdot (2 \cdot d_{model}) \\ &= 12 \cdot d_{model}^2 + 13 \cdot d_{model} \end{aligned}$$

$$\begin{aligned} N_{Decoder} &= 2 \cdot N_{attention} + 1 \cdot N_{FFN} + 3 \cdot N_{Layernorm} \\ &= 2 \cdot 4 \cdot (d_{model}^2 + d_{model}) + 8 \cdot d_{model}^2 + 5 \cdot d_{model} + 3 \cdot (2 \cdot d_{model}) \\ &= 16 \cdot d_{model}^2 + 19 \cdot d_{model} \end{aligned}$$

$$\begin{aligned} N_{Total} &= 12 \cdot d_{model}^2 + 13 \cdot d_{model} + 16 \cdot d_{model}^2 + 19 \cdot d_{model} \\ &= 28 \cdot d_{model}^2 + 29 \cdot d_{model} \end{aligned}$$

APPROXIMATE PARAMETER COUNT

Since the parameter count scales quadratically with d_{model} we can approximate it by omitting the layernorm parameters and the bias terms!

$$N_{Encoder} = 1 \cdot N_{attention} + 1 \cdot N_{FFN} = 4 \cdot d_{model}^2 + 8 \cdot d_{model}^2 = 12 \cdot d_{model}^2$$

$$N_{Decoder} = 2 \cdot N_{attention} + 1 \cdot N_{FFN} = 2 \cdot 4 \cdot d_{model}^2 + 8 \cdot d_{model}^2 = 16 \cdot d_{model}^2$$

$$N_{Total} = 12 \cdot d_{model}^2 + 16 \cdot d_{model}^2 = 28 \cdot d_{model}^2$$

This is the approximate parameter count for one layer!

CALCULATE PARAMETER COUNT

Now we can use those formulas to calculate the parameter count of the vanilla transformer

- From the paper we know $n_{Layers} = 6$ for both Encoder and Decoder and $d_{model} = 512$
- Let's first use the exact formula:

$$\begin{aligned} N_{Params} &= 6 \cdot (12 \cdot 512^2 + 13 \cdot 512) + 6 \cdot (16 \cdot 512^2 + 19 \cdot 512) \\ &= 44,138,496 \end{aligned}$$

- And now with the approximation:

$$\begin{aligned} N_{Params} &= 6 \cdot (12 \cdot 512^2) + 6 \cdot (16 \cdot 512^2) \\ &= 44,040,192 \end{aligned}$$

The approximation is really good, which is why we can ignore the bias and layernorm parameters!

EMBEDDING PARAMETERS

We also have to consider the parameters of the embeddings!

- Let M be the maximum sequence length and V the size of the vocabulary
- We have $M \times d_{model}$ positional embedding parameters, which are fixed and not learned
- And $V \times d_{model}$ token embeddings parameters, which are learned
- The original transformer has $V = 37000$ tokens:

$$N_{Embedding} = 37000 \cdot 512 = 18,944,000$$

- Adding those to the previous count yields:

$$N_{Final} = 18,944,000 + 44,040,192 = 62,984,192$$