Implementing Attention

We now provide an in-depth view of how Attention is actually achieved.

Implementing Attention (Attention_Lookup.ipynb)

Where do all the weights come from ?

Ignoring the weights associated with the various embeddings, the weights come from

- Attention
- Feed forward Network

This is for *each* Transformer block

- we will stack n_{layer} such blocks

For Attention, the weights/parameters are in the matrices $\mathbf{W}_Q$, $\mathbf{W}_K$, $\mathbf{W}_V$ and $\mathbf{W}_O$

- all of size $\mathcal{O}\left(d_{\text{model}}^2\right)$, total:
  $$4 * \mathcal{O}\left(d_{\text{model}}^2\right)$$

For the Feed forward network, there are two `Dense` layers

- the first mapping attention output of size d_{model} to size d_{ff}
- the second mapping size d_{ff} to standard output size d_{model}
- total Feed forward weights are $2 * (d_{\text{model}} * d_{\text{ff}})$

Using the standard

$$d_{\text{ff}} = 4 * d_{\text{model}}$$

total Feed forward weights per block

$$2 * (d_{\text{model}} * 4 * d_{\text{model}}) = 8 * \mathcal{O}\left(d_{\text{model}}^2\right)$$

$$
\begin{aligned}
\text{Total parameters} &= \text{Projection matrix parms} &+ \quad &\text{FFN parms} \\
&= 4 * \mathcal{O}\left(d_{\text{model}}^2\right) &+ \quad &8 * \mathcal{O}\left(d_{\text{model}}^2\right) \\
&= 12 * \mathcal{O}\left(d_{\text{model}}^2\right)
\end{aligned}
$$

This is for a single Transformer block

- This gets multiplied by the number n_{layer} stacked blocks..

Notice

- that $\frac{1}{3}$ of the total weights
- come from *linear* projections
    - the matrices associated with Attention
- rather than non-linearities
    - confined to Feed forward network

For GPT-3

- $n_{\text{layer}} = 96$
- $d_{\text{model}} = 12 * 1024$

Total Transformer (non-embedding) weights

$$96 * 12 * (12 * 1024)^2 = 174 \text{ billion}$$

In [2]:
```
print("Done")
```

```
Done
```