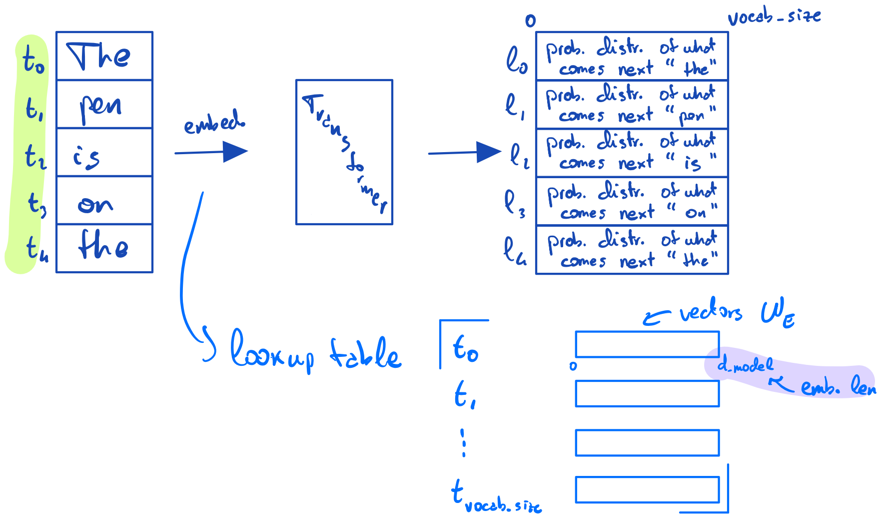
Tranformer Notes

Softmax $x_i \to \dfrac{e^{x_i}}{\sum e^{x_j}}$ ← makes everything positive

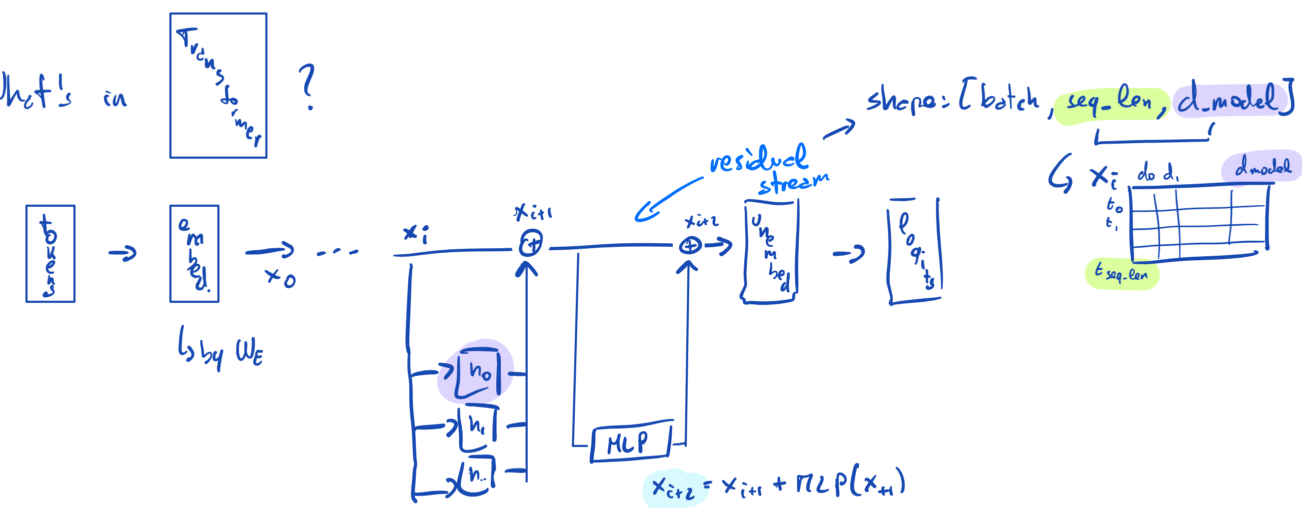
↖ everything add to 1

where $x = [\,|\,|\,|\,|\,|\,]$ ↖ vector

Causal attention: moves forward and predicts token t_n based only on tokens $t_0, \dots, t_{n-1}$



What's in Transformer ?

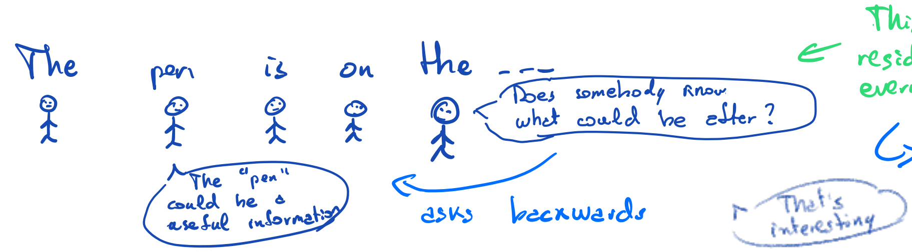


$x_{i+2} = x_{i+1} + MLP(x_{i+1})$

$x_{i+1} = x_i + \sum_{h \in H_i} h(x_i)$

a layer / block / residual blocks

- Residual stream: where model "stores" information (and remembers)
- Attention ?
  - ↳ Moves info from prior positions to current tokens



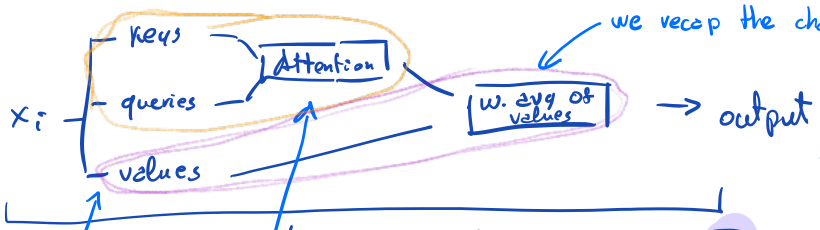
← This that compose the residual stream. It's where everybody communicate!

↳ QK circuit

→ OV circuit

How does it works?

we recap the chat above (trough w. avg)



some linear transf. $\forall$ seq. position

↳ single attention head $\boxed{h_0}$

How much query position pays attention to each key position?
(i.e. the chat above between tokens)

OK! We collected some info and connection between tokens: now let's share them into the residual stream!

- MLP?

↳ simple NN  with (usually) a hidden layer 4x bigger than d_model

↑ GeLU in here

I belive

In math? $MLP := f(x^T W^{in}) W^{out} = \sum_{i=1}^{4x\ d\_model} f(x^T W^{in}_{:,i}) W^{out}_{i,:}$

→ GeLU

→ residual stream

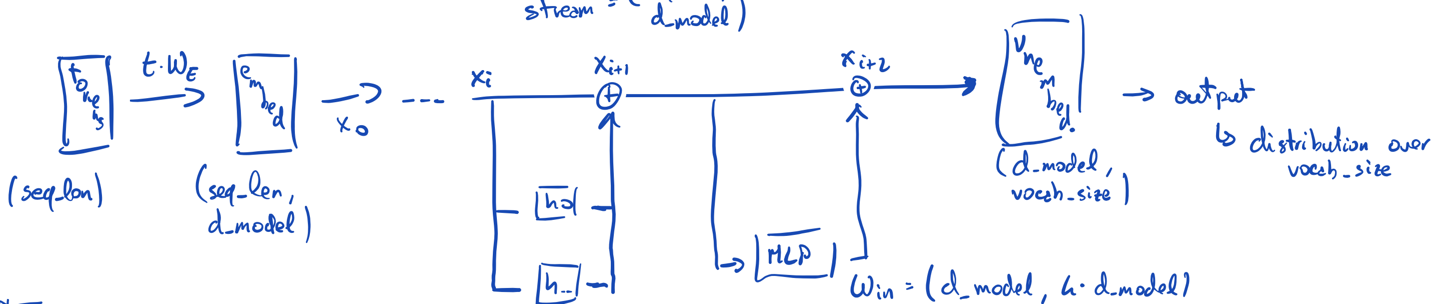
linear transformations (the neurons)

Is this detected feature relevant? (aka; the subject is a cat, seems important? Probably yes, let me write it to the residual stream)

By training a transformer we train layers to understand connection and what's important in a sentence!

I really like to look at matrix dimensions (shape) so:

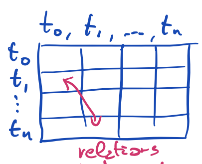
residual stream = (seq_len, d_model)



$W_{in} = (d\_model, 4 \cdot d\_model)$
$W_{out} = (4 \cdot d\_model, d\_model)$

Q,K,V = (seq_len, n_heads, d_model/n_heads)

Attentions = (n_heads, seq_len, seq_len)



For GPT-2:

d_model: 768

vocab_size: 50257

n_heads: 12

n_layers: 12

d_heads = d_model / n_heads = 64

d_mlp = 4·d_model = 3072

Implementation:

Steps:

[batch, seq_len, d_model] [batch, seq_len, n_heads, d_head]

1. Linear map residual stream to queries, keys and values

   Math:

   [n_heads, d_model, d_head]

   queries: $x^T W_Q + bias_Q$

   keyes: $x W_K^T + bias_K$

   values: $x^T W_V + bias_V$

   Pseudo-code:

   ```
   queries = einsum(
           x, W_Q,
           "b s d_model, n_heads d_model d_head -> b s n_heads d_heads"
   ) + bias_Q
   keyes: same as above but with W_Q and bias_Q
   values: same as above but with W_Q and bias_Q
   ```

   [batch, n_heads, s_Q, s_K] [batch, $s_{\{Q,K\}}$, n_heads, d_heads]

2. Get attention scores using queries and keys

   Math:

   $$\text{scores} = x^T W_Q W^T x$$

   Pseudo-code:

   ```
   scores = einsum(
       queries, keys,
       "b s_Q n_heads d_heads, b s_K n_heads d_heads -> b n_heads s_Q s_K"
   )
   ```

   [s_Q, s_K] where Upper Triangular filled w/ 0

3. Scale it (by $\sqrt{d\_head}$), mask and apply softmax (along keys dimension)

   get a prob. distr. for each query

   Math:

   $$A = \text{softmax}\left(\frac{\text{mask}(\text{scores})}{\sqrt{d\_head}}\right)$$

   [batch, n_heads, s_Q, s_K]

   Pseudo-code:

   ```
   A = softmax(mask(scores) / sqrt(d_head), dim = -1)
   ```

   [batch, n_heads, s_Q, s_K] [batch, $s_V = s_K$, n_head, d_head]

4. Apply linear map from source A to destination tokens $x W_V$ getting the w. avg of values

   [batch, s_V, n_head, d_head]

   Math:

   $$z = A \cdot x W_V$$

   Pseudo-code:

   ```
   z = einsum(
       A, values,
       "b n_heads s_Q s_K, b s_K n_heads d_head -> b s_V n_head d_head"
   )
   ```

   [batch, s, d_model]

5. Get the output OV summing over heads (or concatenation of heads) + $bias_V$

   Math:

   [n_heads, d_heads, d_model] [b, s, n_heads, d_heads]

   $$OV = z \cdot W_O + bias_V$$

Pseudo - code

OV = einsum (

z, W_O

"b s_v n_heads d_heads , n_heads d_heads d_model -> b s_v d_model"

implicit concat or position sum

) + $bias_v$

Note that there are two circuits: QK and OV circuits