



Parameters, Memories, and Computations in Transformers

Kun Yuan

Center for Machine Learning Research @ Peking University

Mar. 19, 2024

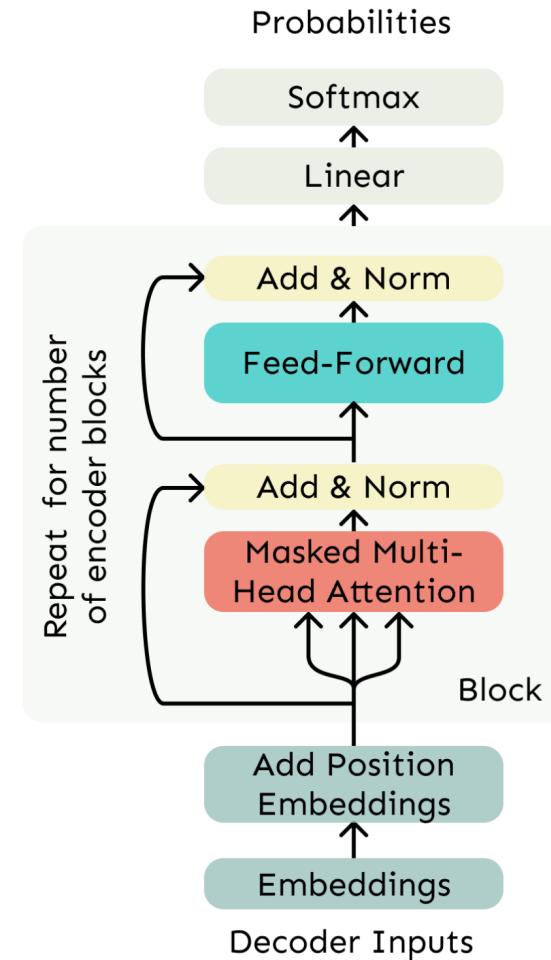


PART 01

Settings and Basics

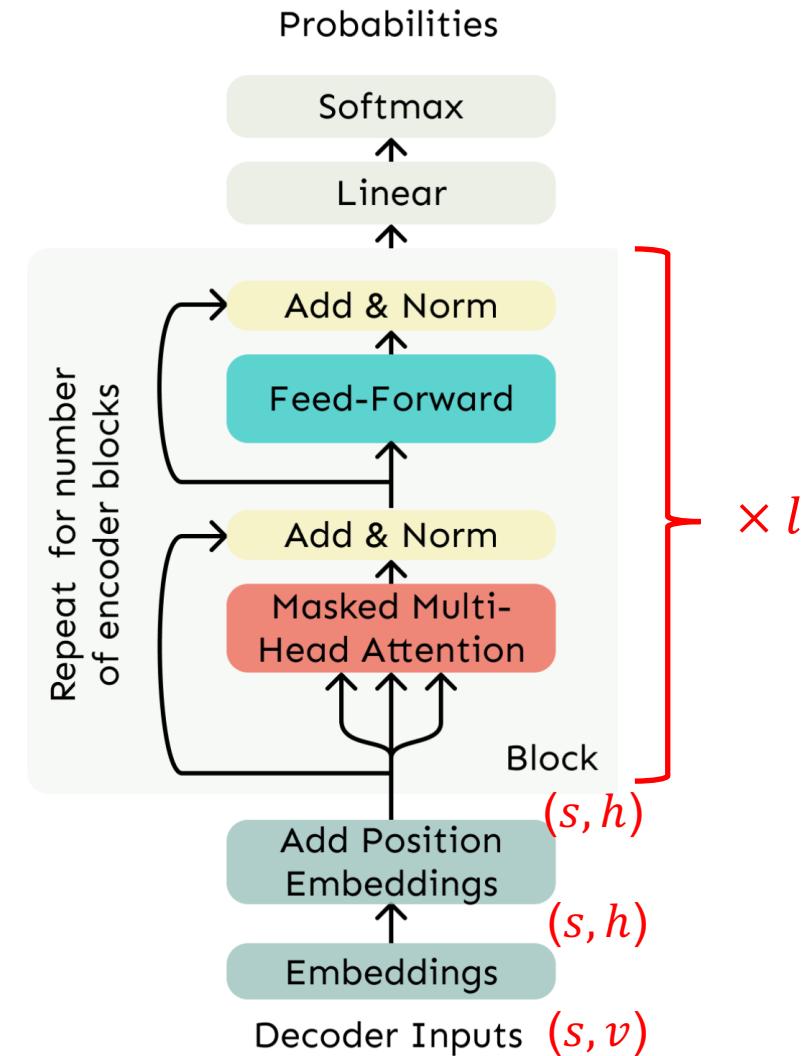
Decoder-only Transformer

- GPT is based on the **decoder-only** transformer
- We will analyze the parameters, memories, and computation costs for decoder-only transformer

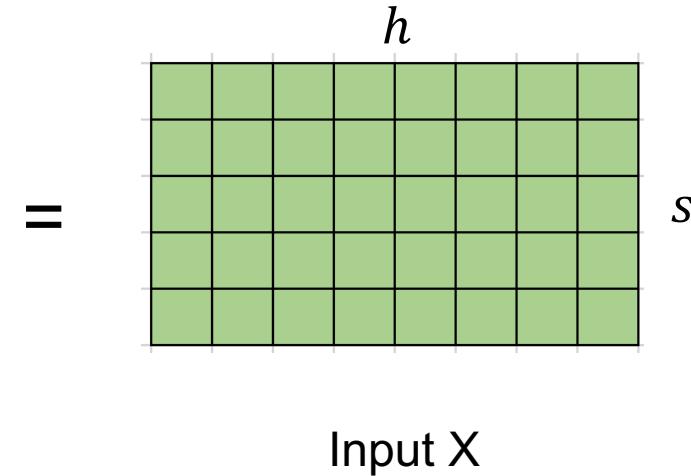
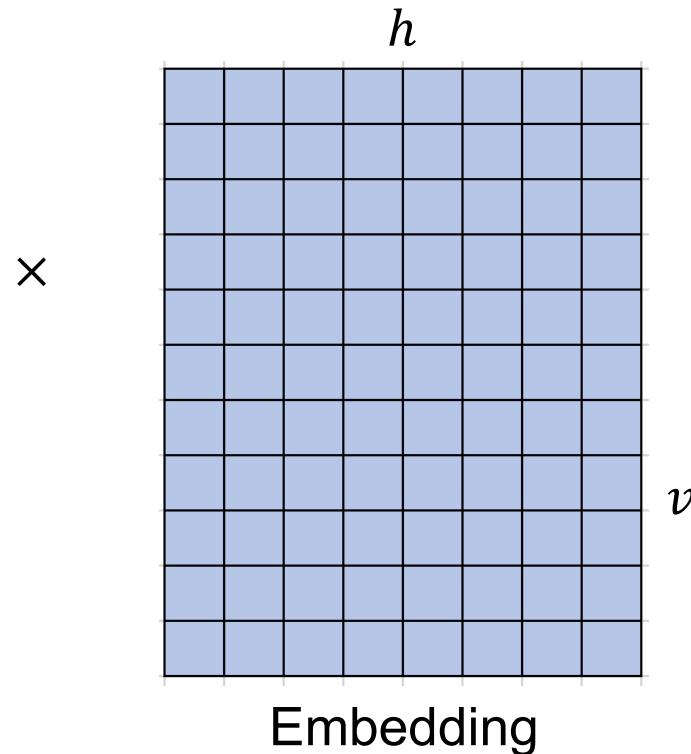
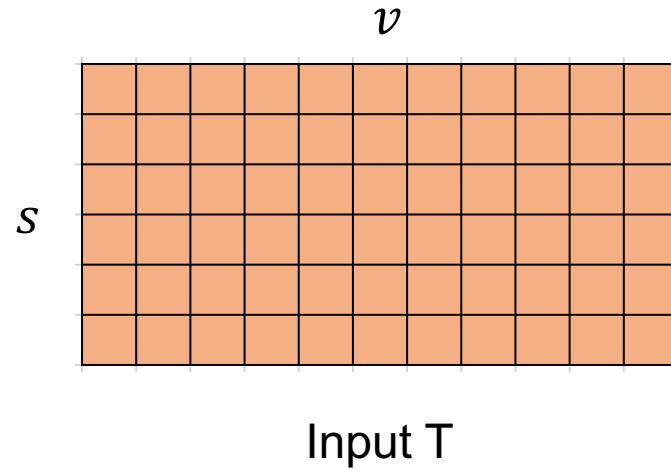


Notations

- Number of the transformer layers: l
- Sequence length: s
- Vocabulary size: v
- Embedding representation dims: h

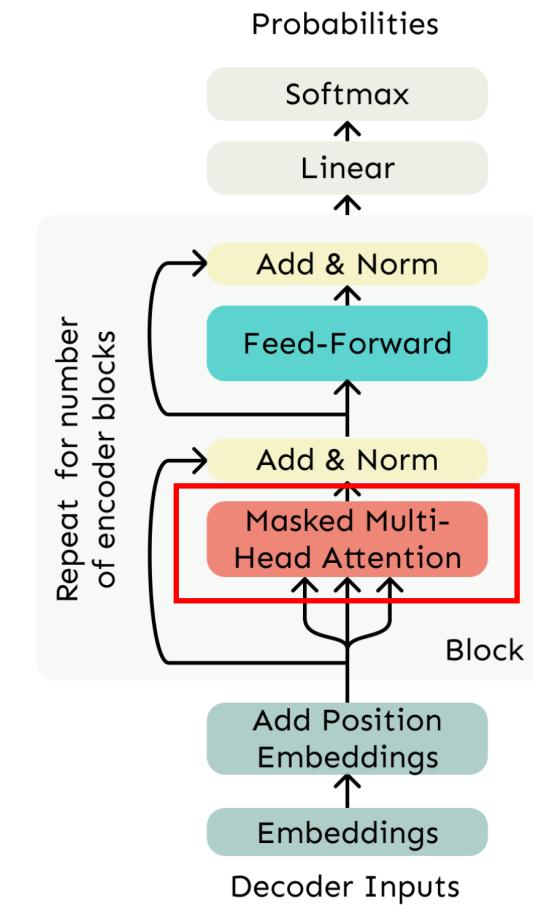
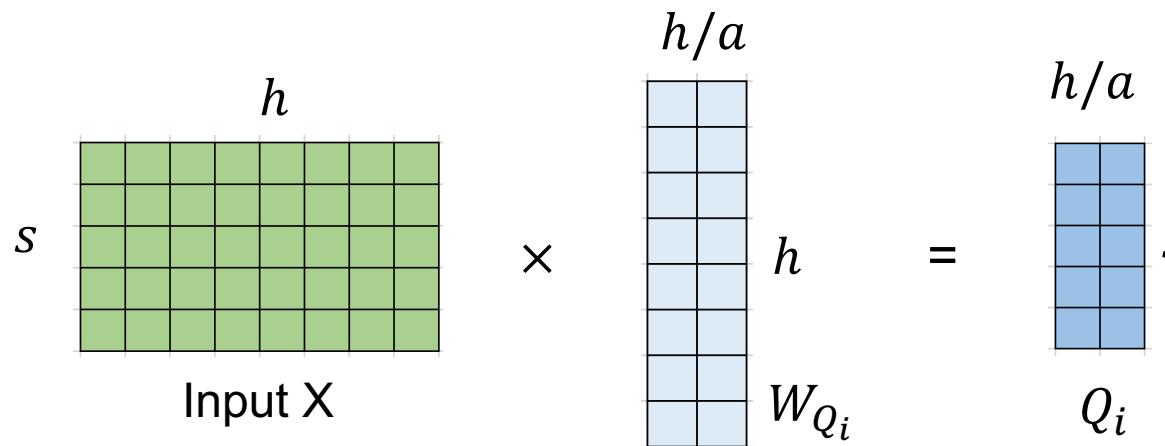


Embedding



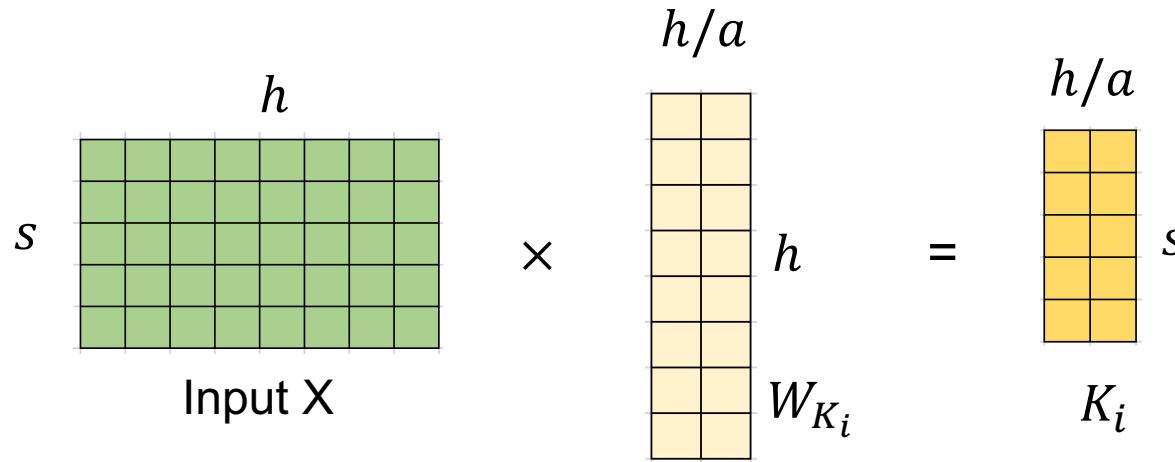
Self-attention

- Number of heads: a
- Dims of each W_{Q_i} , W_{K_i} and W_{V_i} : $h \times \frac{h}{a}$



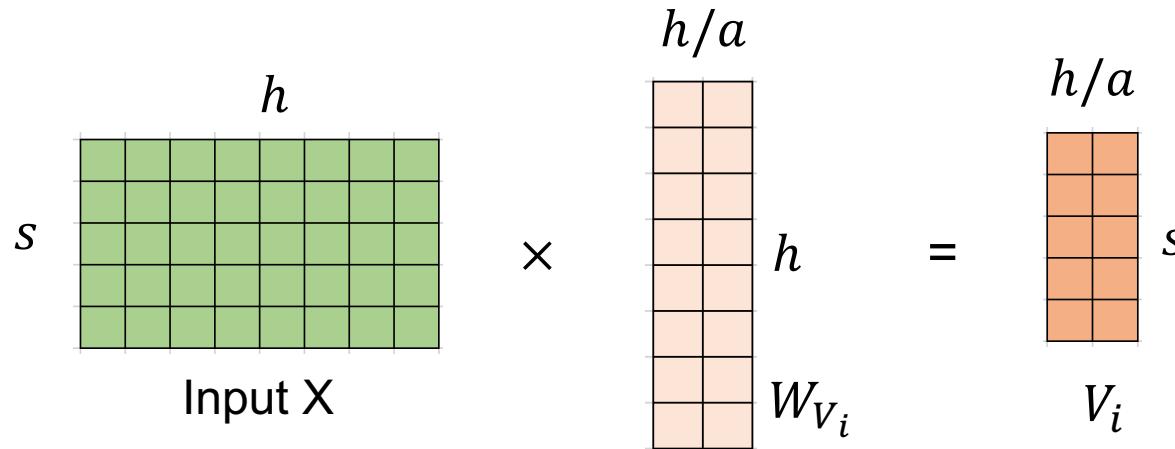
Self-attention

- Number of heads: a
- Dims of each W_{Q_i} , W_{K_i} and W_{V_i} : $h \times \frac{h}{a}$



Self-attention

- Number of heads: a
- Dims of each W_{Q_i} , W_{K_i} and W_{V_i} : $h \times \frac{h}{a}$



Self-attention

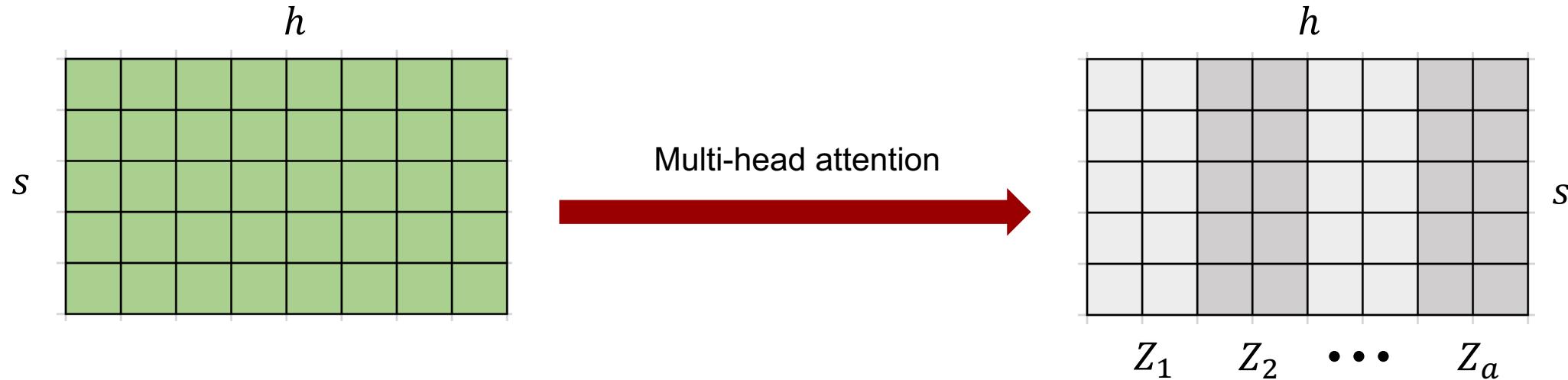
- Number of heads: a
- Dims of each W_{Q_i} , W_{K_i} and W_{V_i} : $h \times \frac{h}{a}$

$$\text{softmax}\left(\frac{Q_i K_i^T}{\sqrt{h/a}}\right) V_i = \text{softmax} \left[\begin{array}{c|c|c} \text{blue grid} & \times & \text{yellow grid} \\ \hline \end{array} \right] \times \begin{array}{c|c|c} \text{orange grid} & \times & \text{light blue grid} \\ \hline \end{array} = \begin{matrix} h/a \\ s \\ Z_i \end{matrix}$$

One-head attention

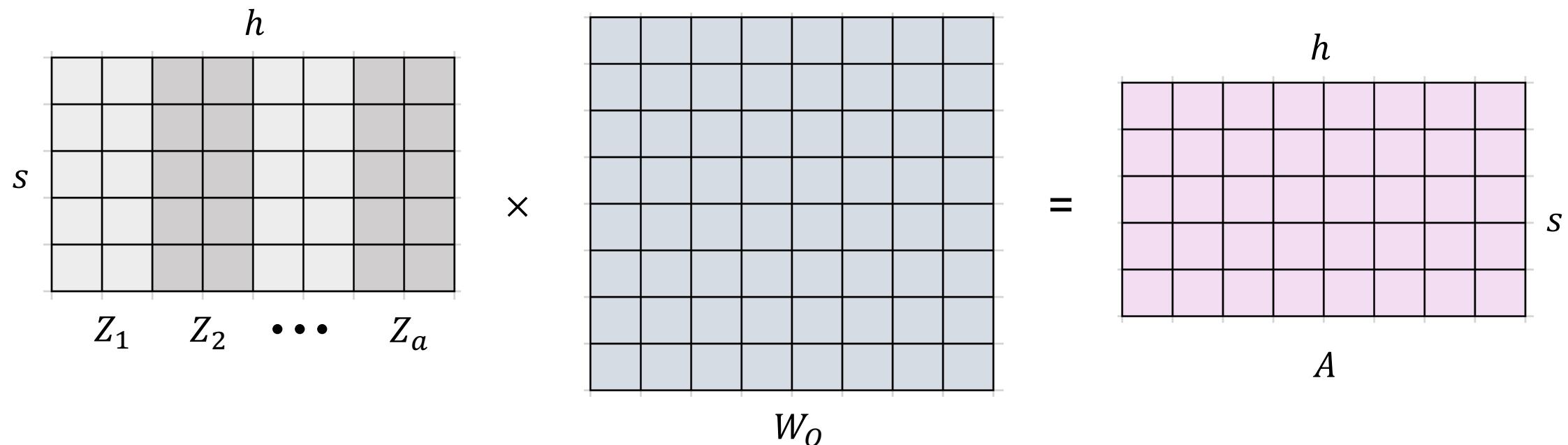
Multi-head attentions

- Number of heads: a
- Dims of each W_{Q_i} , W_{K_i} and W_{V_i} : $h \times \frac{h}{a}$



Multi-head attentions

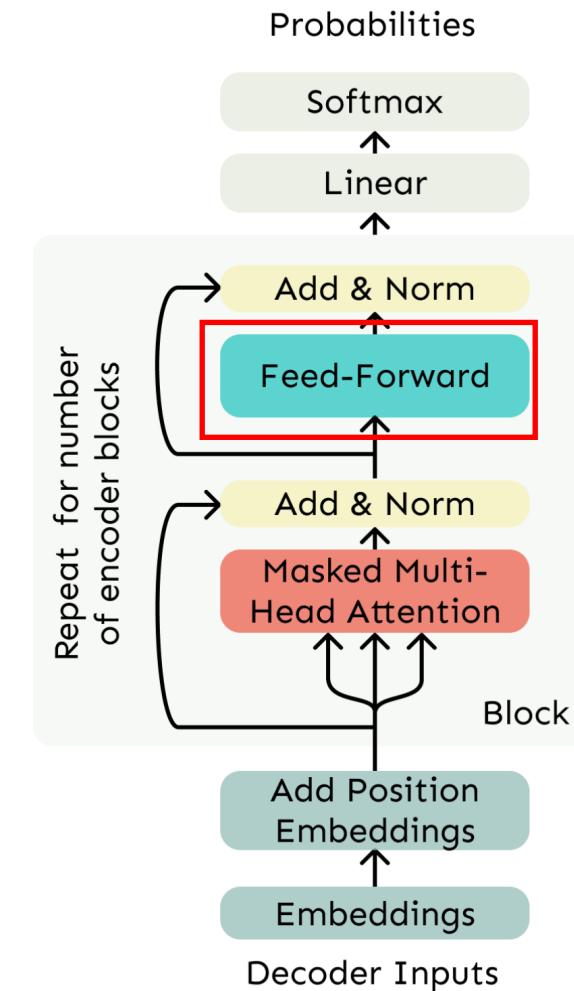
- Number of heads: a
- Dims of each W_O : $h \times h$
- Dims of each W_{Q_i} , W_{K_i} and W_{V_i} : $h \times \frac{h}{a}$



Feed-forward Layer

$$X' = \text{ReLU}(A \cdot W_1 + b_1) \cdot W_2 + b_2$$

- Dims of W_1 : $h \times 4h$
- Dims of each W_2 : $4h \times h$

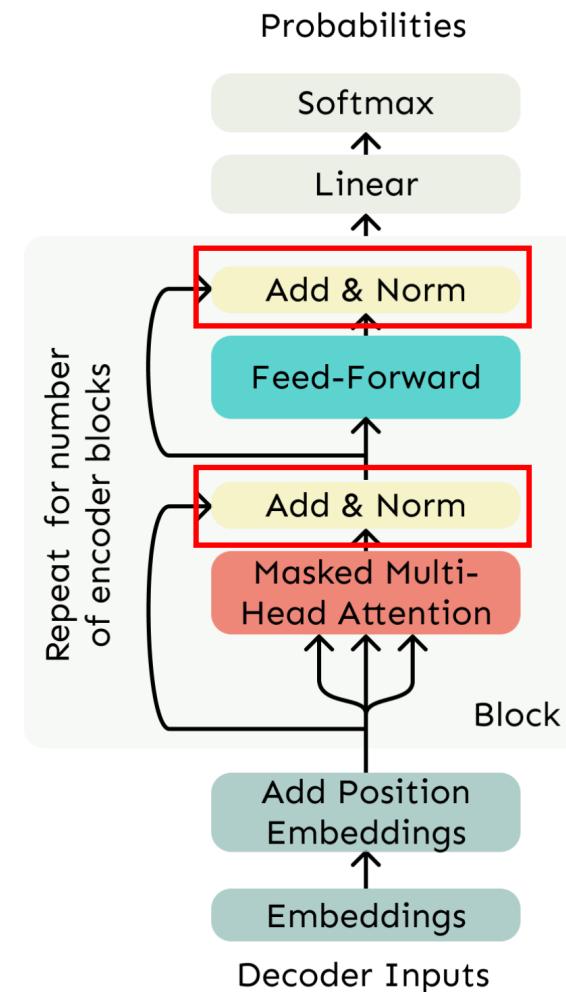


Layer normalization

- Then layer normalization computes:

$$\text{output} = \frac{x - \mu}{\sqrt{\sigma + \epsilon}} * \gamma + \beta$$

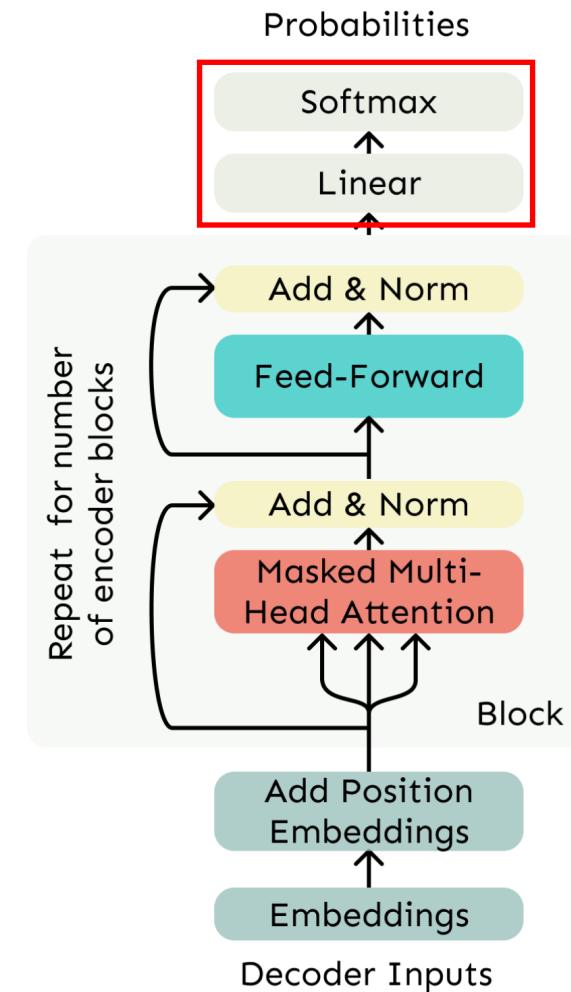
- Dims of γ and β : h



Probability prediction

$$p = \text{Softmax}(X \cdot W_v + b_v)$$

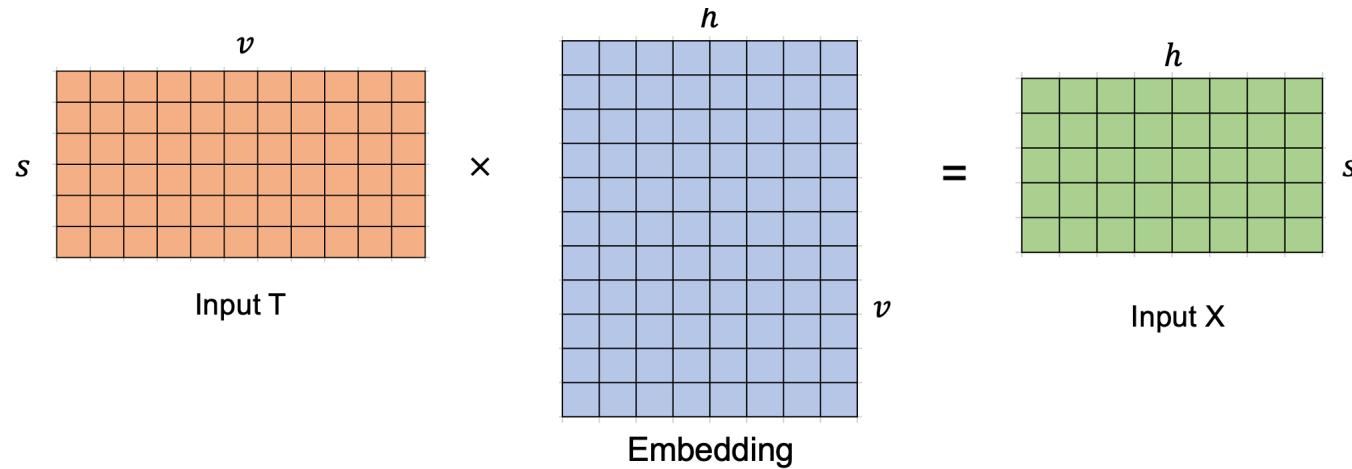
- Dims of W_v : $h \times v$



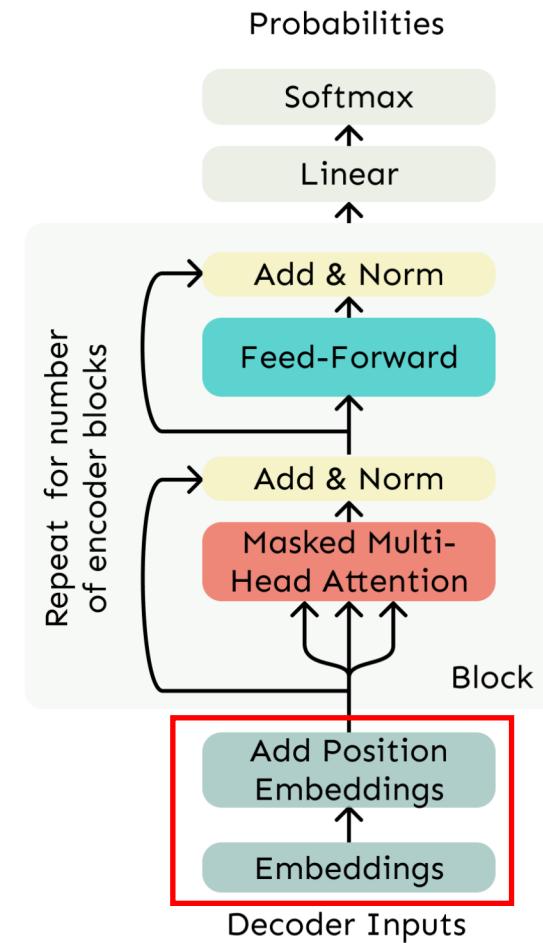
PART 02

Parameters analysis

Embedding



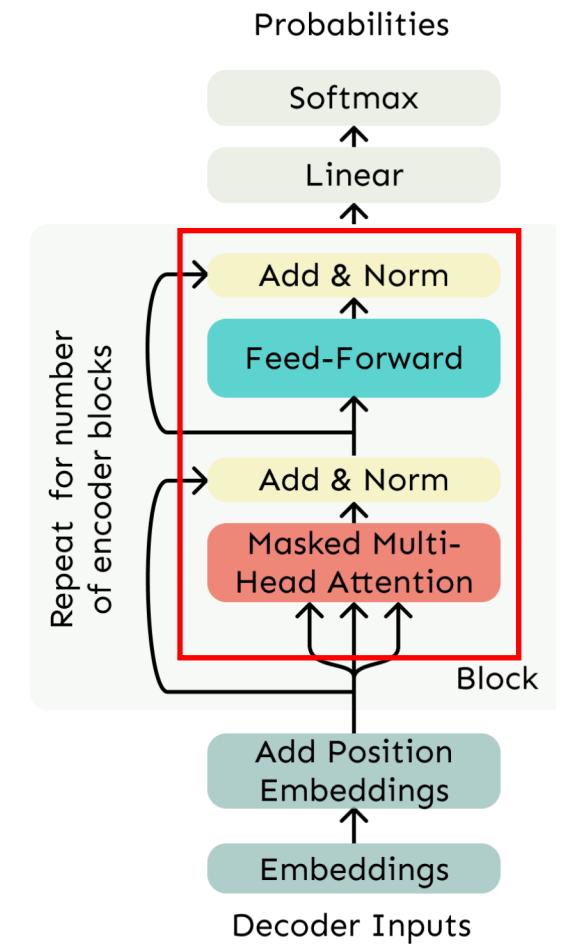
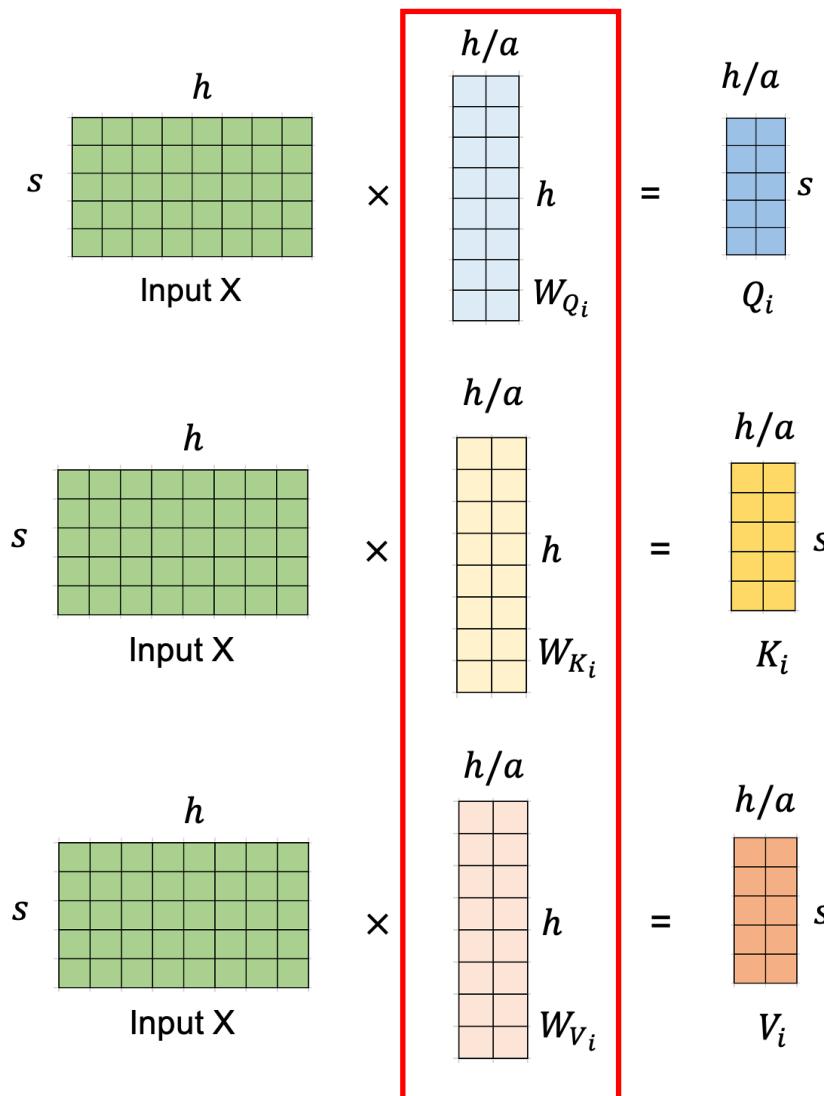
- We need to store the embedding with parameters vh
- Position embedding can be ignored when using RoPE and ALiBi



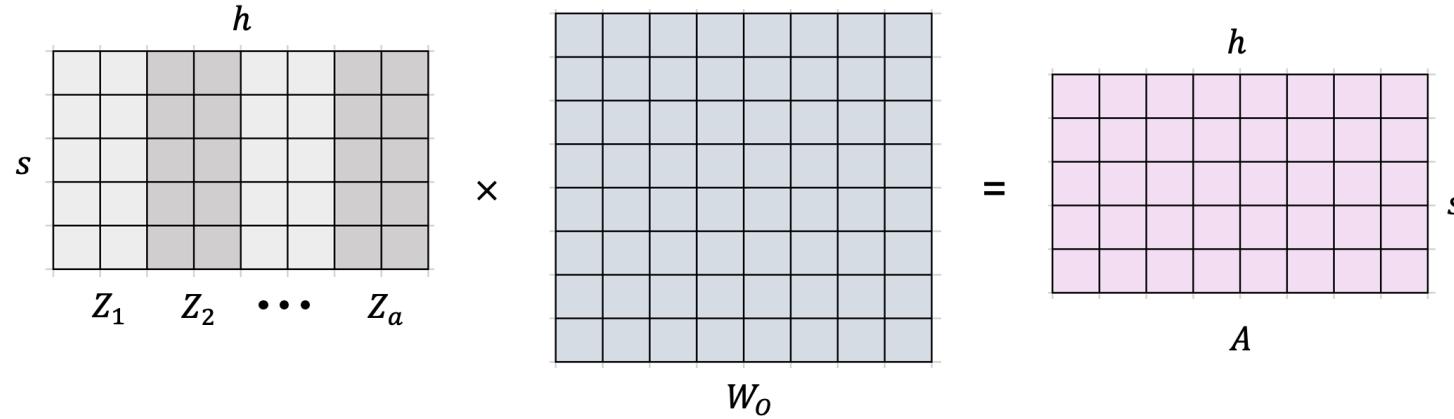
Multi-head attentions

- We need to store W_Q , W_K and W_V

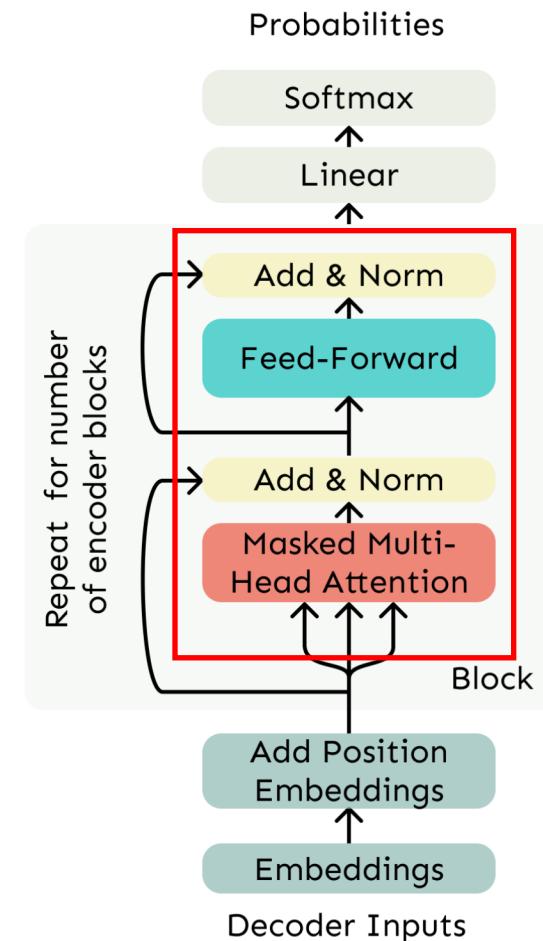
$$3(h^2/a) \times a = 3h^2$$



Multi-head attentions



- We need to store W_O : h^2

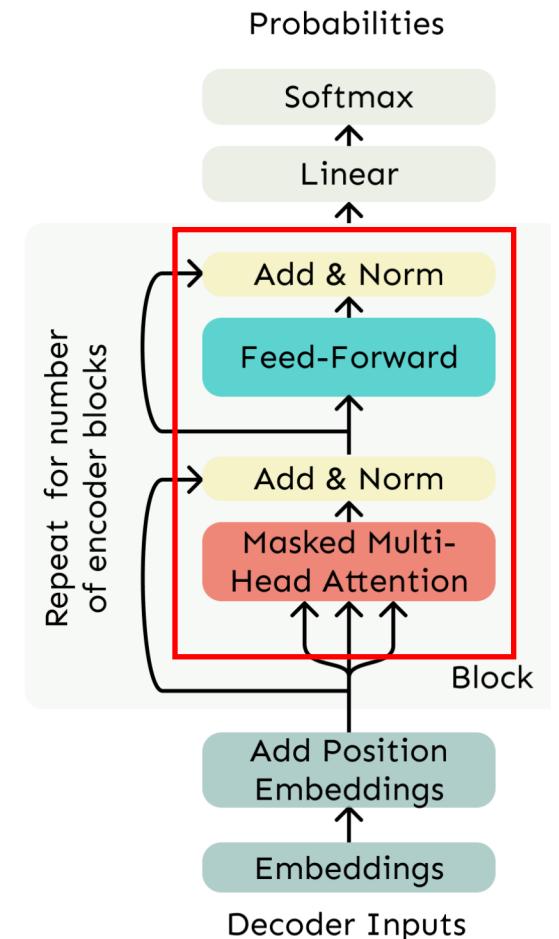


Layer normalization

- Then layer normalization computes:

$$\text{output} = \frac{x - \mu}{\sqrt{\sigma + \epsilon}} * \gamma + \beta$$

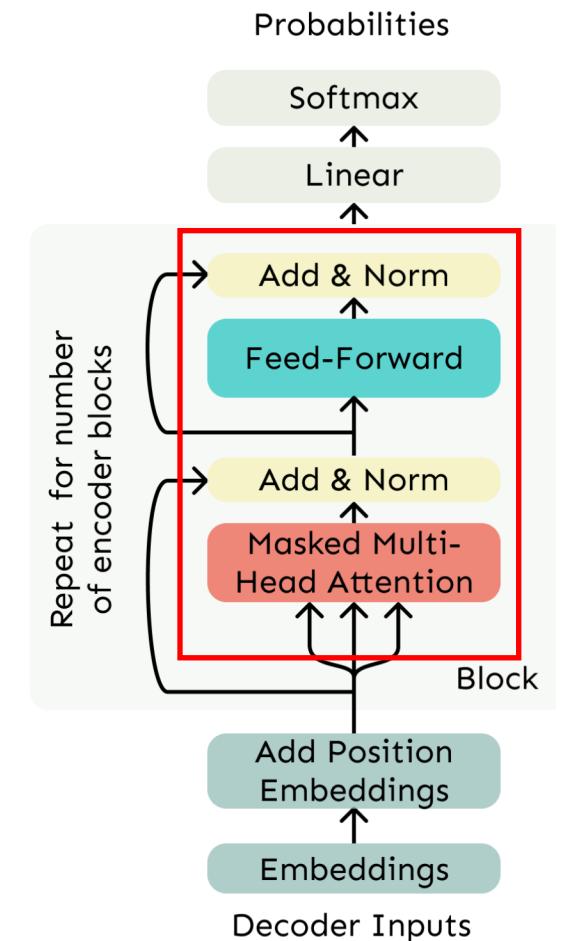
- Dims of γ and β : h
- We should store γ and β . Since their parameters are much smaller than h^2 and vh , we can ignore them



Feed-forward layers

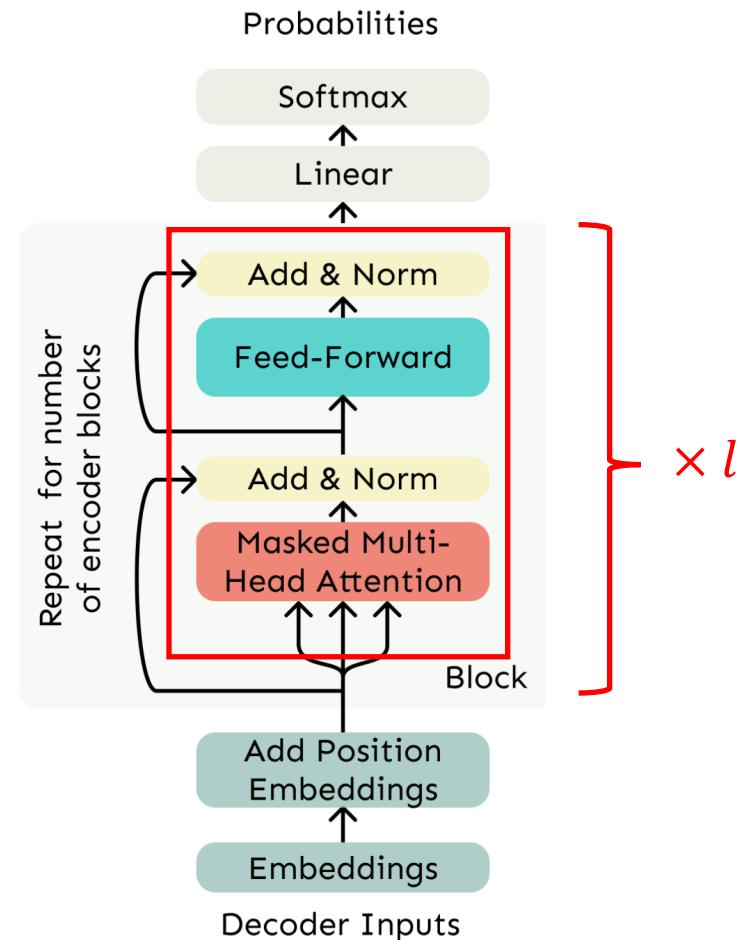
$$X' = \text{ReLU}(A \cdot W_1 + b_1) \cdot W_2 + b_2$$

- Dims of W_1 : $h \times 4h$
- Dims of each W_2 : $4h \times h$
- We need to store W_1 and W_2 : $8h^2$
- The storage of b_1 and b_2 can be ignored



Transformer block

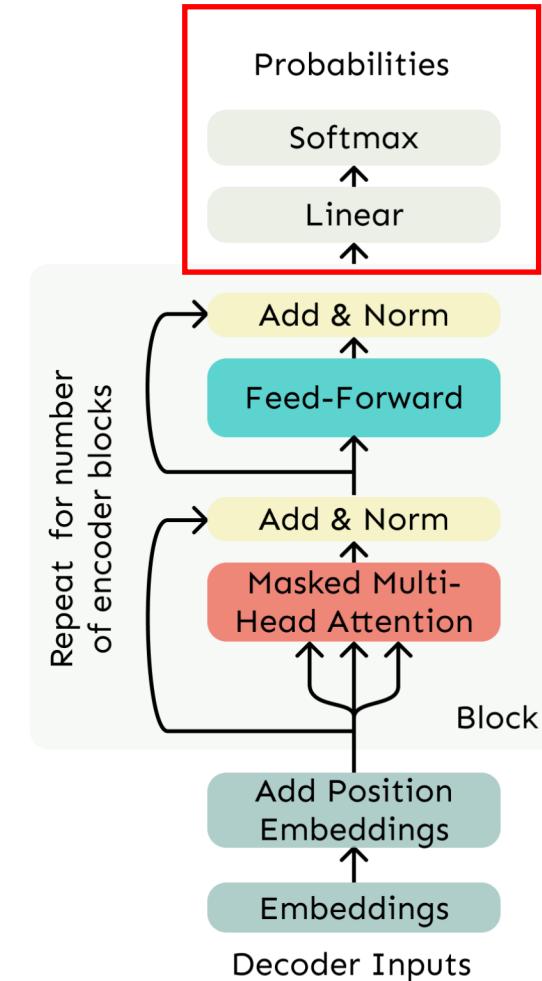
- Multi-head attentions: $4h^2$
- Feed-forward layers : $8h^2$
- l layers of attentions : $(4h^2 + 8h^2) \times l = 12lh^2$



Probability predictions

$$p = \text{Softmax}(X \cdot W_v + b_v)$$

- Dims of W_v : $h \times v$
- We need to store W_v : hv parameters
- b_v can be ignored

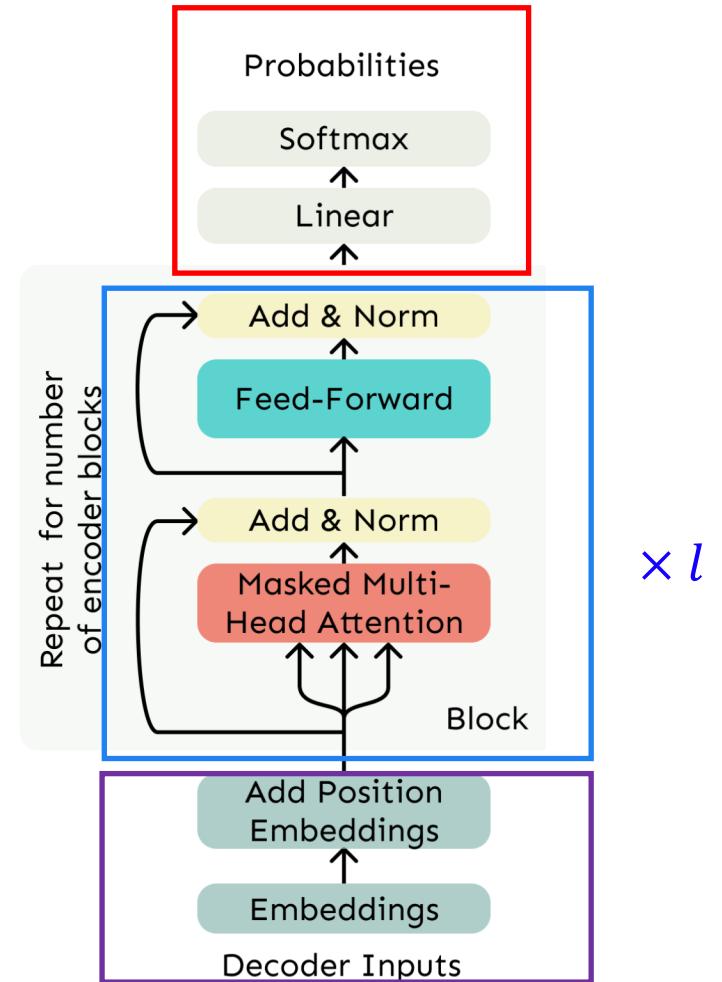


Total parameters

- Embeddings: vh
- Attention blocks: $12lh^2$
- Probability predictions: vh

Total parameters:

$$12\ell h^2 + 2vh$$



Example: LLaMA parameters

- Now we compare our theoretical evaluations with LLaMA model
- $12\ell h^2 + 2vh$ is a very accurate estimation

实际参数量	Embedding h	Attention层数l	Vocab大小v	预估参数量
6.7B	4096	32	32000	6,704,594,944
13.0B	5120	40	32000	12,910,592,000
32.5B	6656	60	32000	32,323,665,920
65.2B	8192	80	32000	64,948,797,440



PART 03

Computations analysis

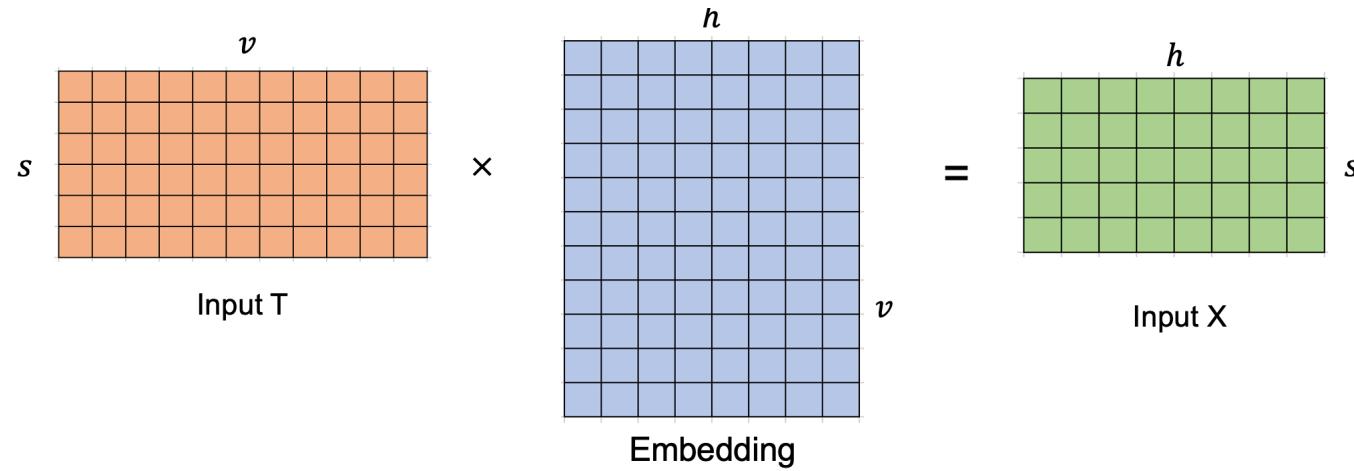
Flops

- FLOPs: Floating point operations; gauges the total amount of computations
- Given matrices $A \in \mathbb{R}^{m \times n}, B \in \mathbb{R}^{n \times p}$, to compute AB , we need

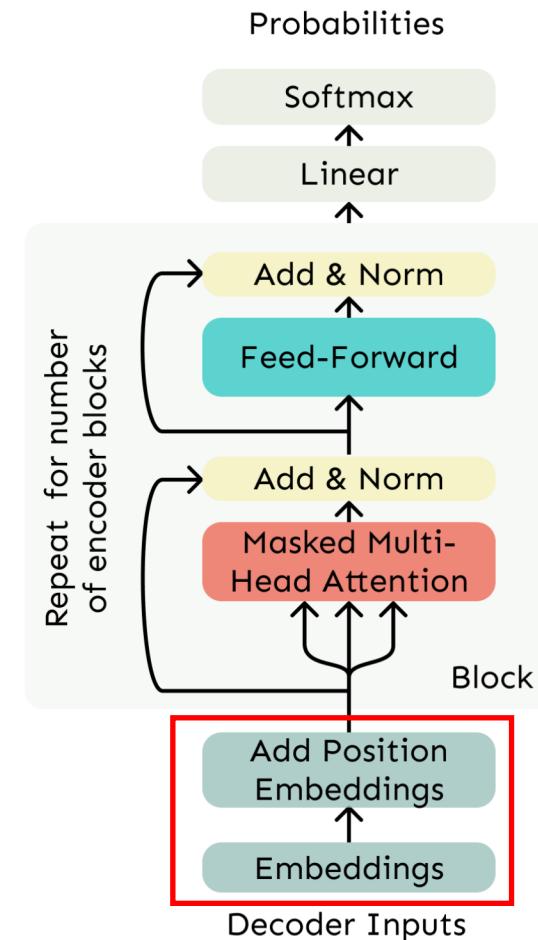
$$\left. \begin{array}{l} mnp \text{ additions} \\ mnp \text{ multiplications} \end{array} \right\} 2mnp \text{ FLOPs}$$

- In transformers, we only count computations raised by matrix operations and ignore vector operations since the later is trivial

Embedding



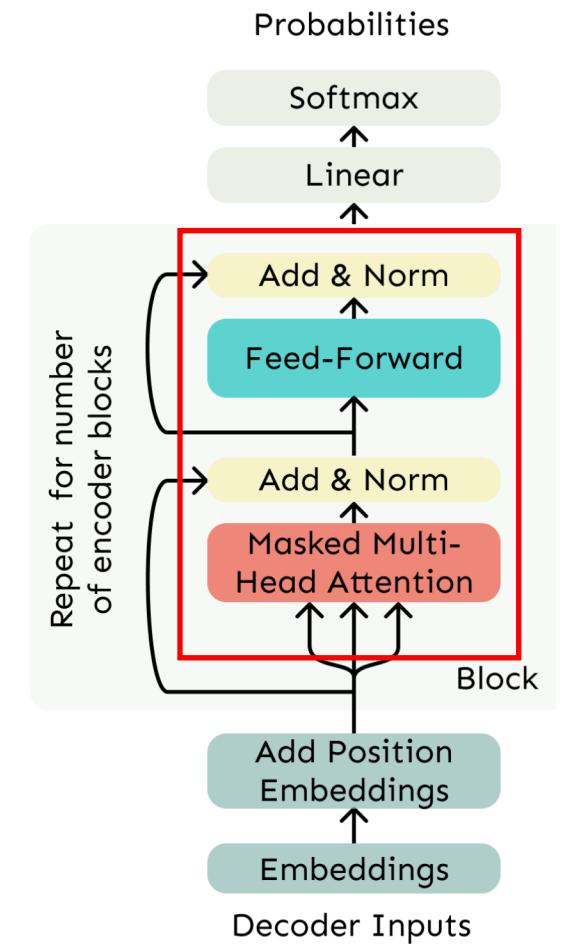
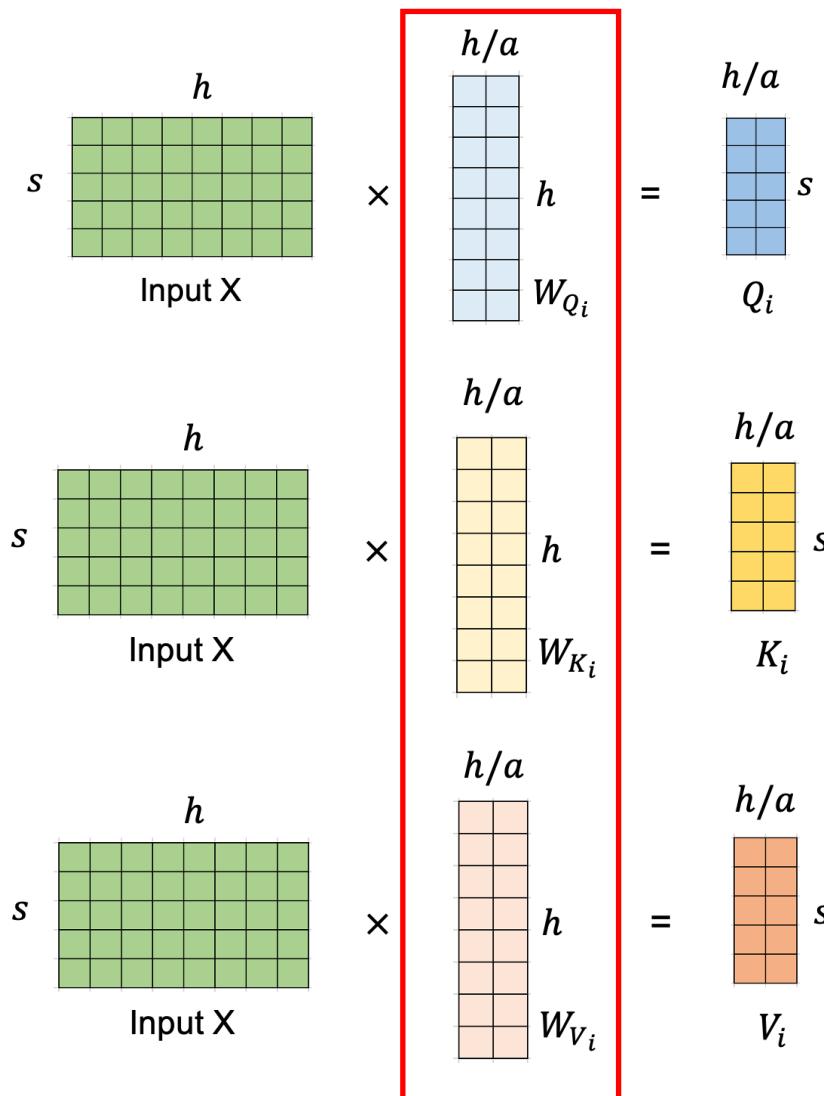
- Word embedding: $2svh$



Multi-head attentions

- Multi-head attentions

$$6(sh^2/a) \times a = 6sh^2$$



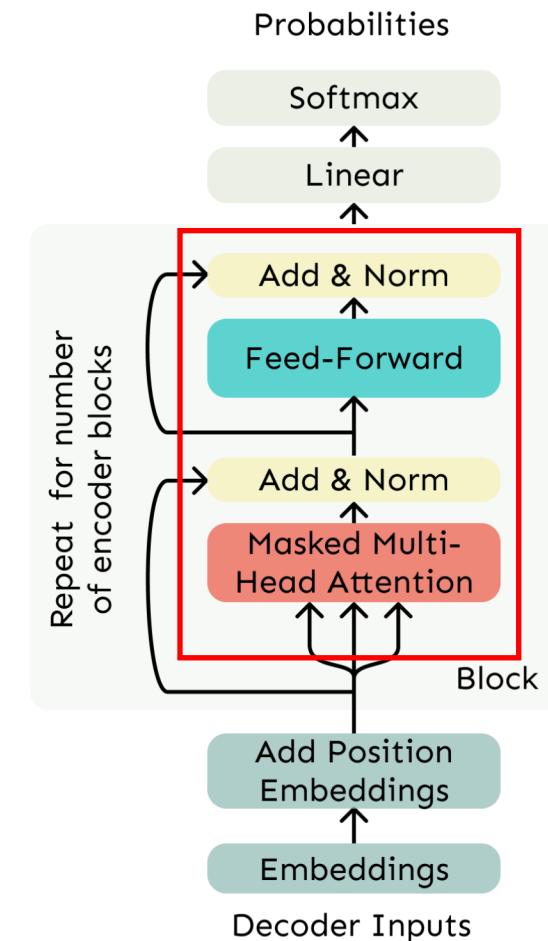
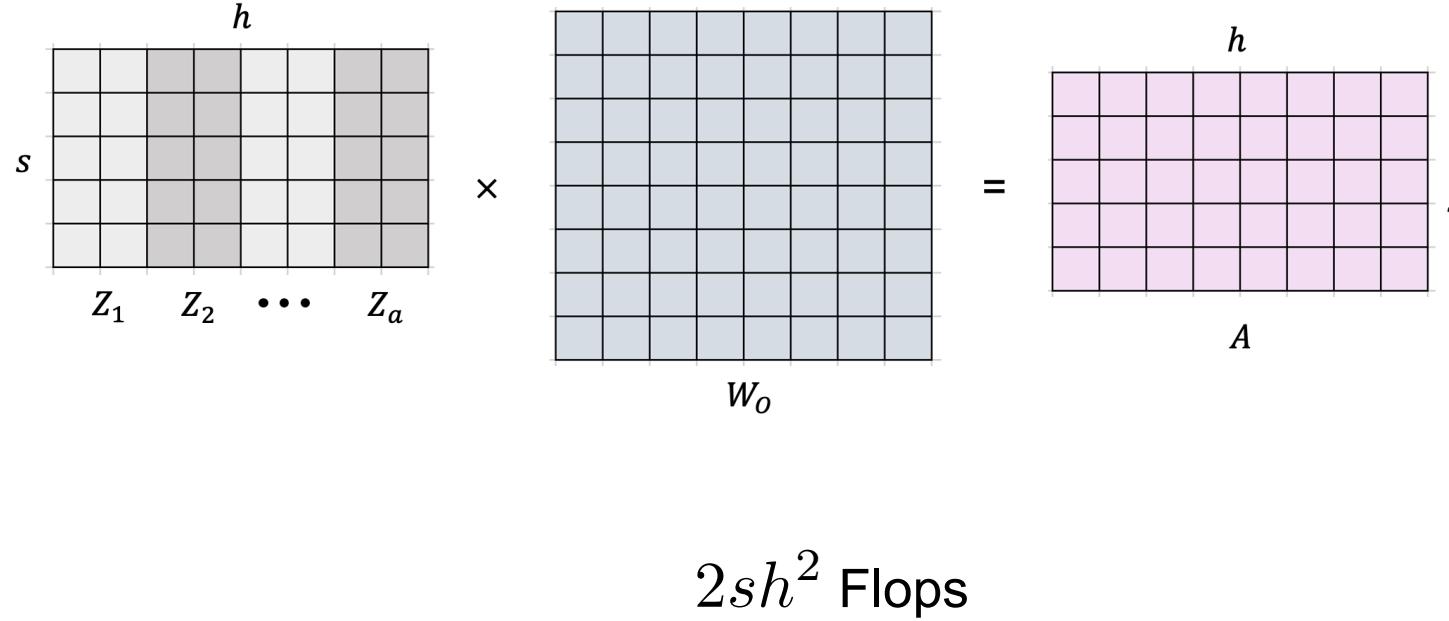
Multi-head attentions

$$\text{softmax}\left(\frac{Q_i K_i^T}{\sqrt{h/a}}\right) V_i = \text{softmax} \left[s \begin{matrix} h/a \\ \times \end{matrix} \begin{matrix} s \\ \times \end{matrix} \right] \times \begin{matrix} h/a \\ s \\ Z_i \end{matrix}$$

The diagram illustrates the computation of a single multi-head attention output Z_i . It shows the input vector V_i being multiplied by a weight matrix s (dimensions $h/a \times s$) and then by another weight matrix s (dimensions $s \times h/a$). The result is then multiplied by a scaling factor s and the softmax function to produce the final output Z_i .

$$(2s^2h/a + 2s^2h/a) \times a = 4s^2h$$

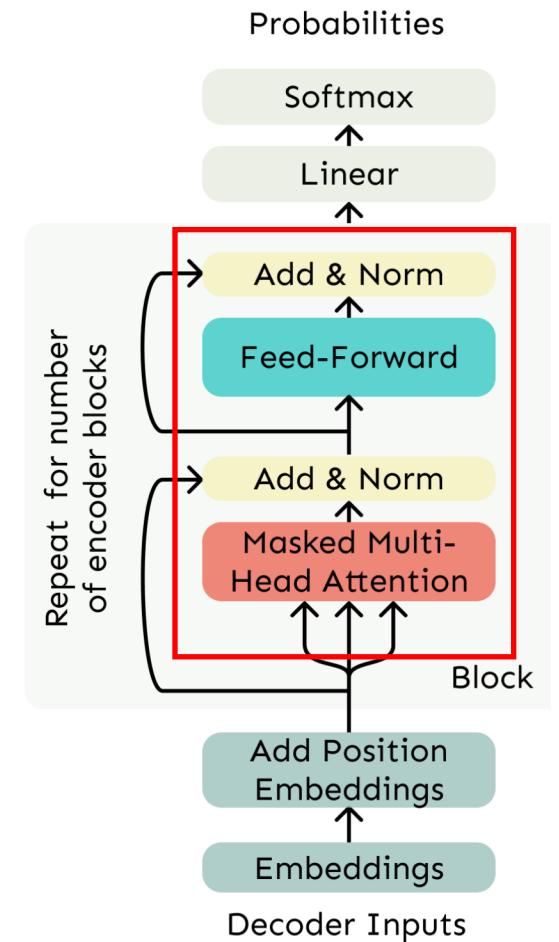
Multi-head attentions



Feed-forward layers

$$X' = \text{ReLU}(A \cdot W_1 + b_1) \cdot W_2 + b_2$$

- Dims of W_1 : $h \times 4h$
 - Dims of each W_2 : $4h \times h$
 - $AW_1 + b_1$ needs: $8sh^2$
 - $A'W_2 + b_2$ needs: $8sh^2$
- $16sh^2$



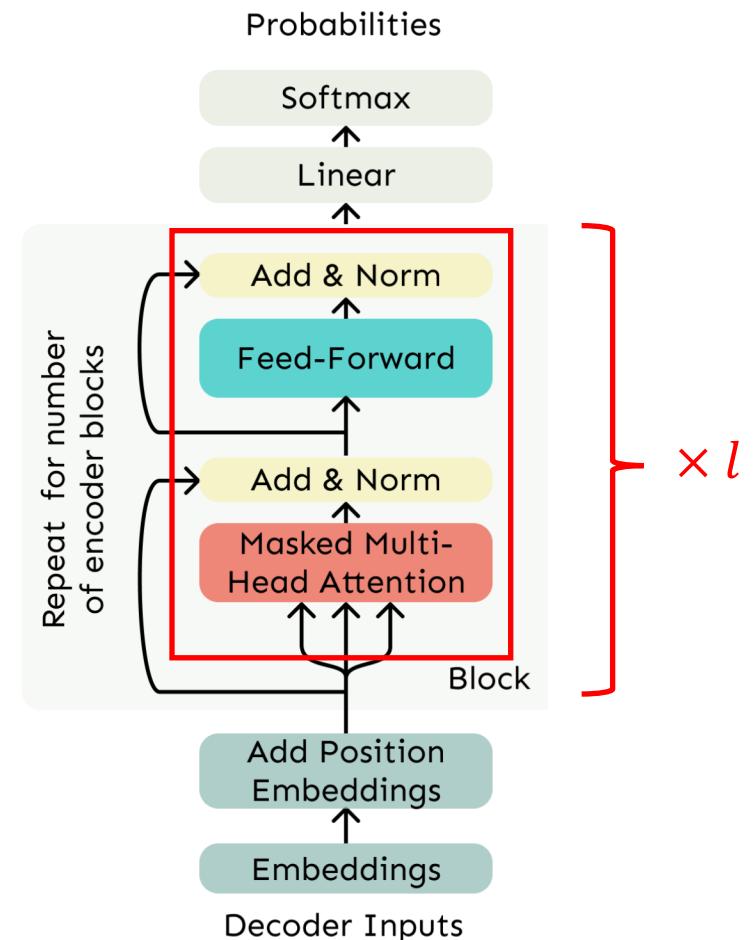
Transformer block

- Multi-head attentions: $8sh^2 + 4s^2h$

- Feed-forward layers : $16sh^2$

- l layers of attentions :

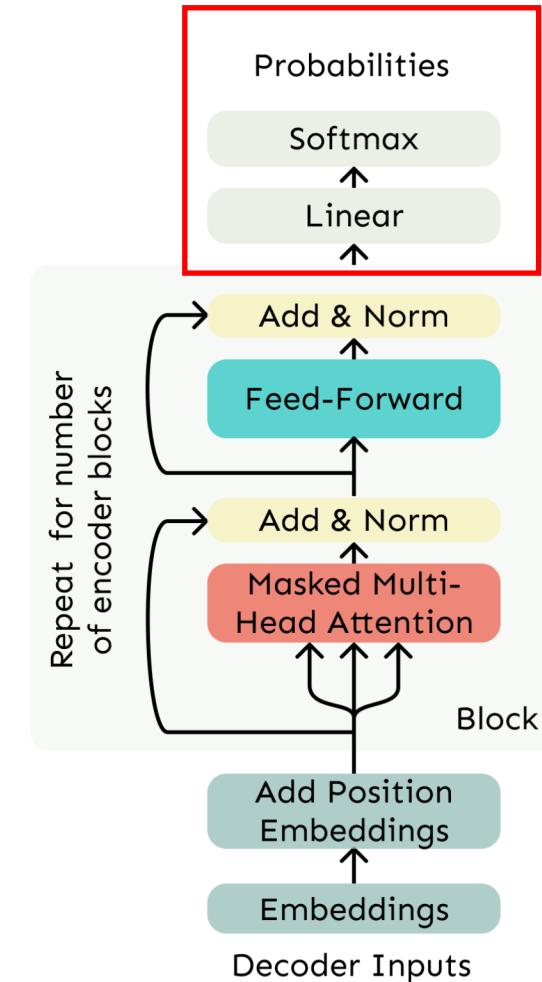
$$(8sh^2 + 16sh^2 + 4s^2h) \times l = 24slh^2 + 4s^2lh$$



Probability predictions

$$p = \text{Softmax}(X \cdot W_v + b_v)$$

- Dims of W_v : $h \times v$
- We need to : $2shv$ FLOPs



Total forward FLOPs

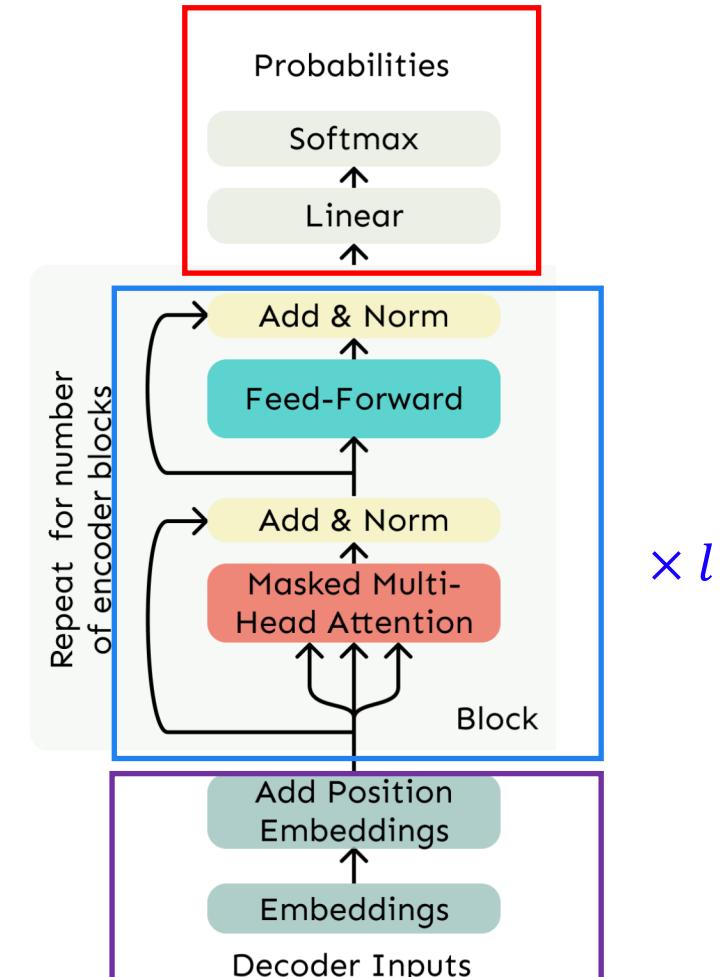
- Embeddings: $2svh$
- Attention blocks: $24lsh^2 + 4s^2h$
- Probability predictions: $2svh$

Total forward FLOPs:

$$\ell(24sh^2 + 4s^2h) + 4svh$$

When using batch-size b , the total forward FLOPs:

$$b\ell(24sh^2 + 4s^2h) + 4bsvh$$



Total forward-backward FLOPs

The backward computations are **twice amount** of the forward computations

Total forward-backward FLOPs

$$\left(b\ell(24sh^2 + 4s^2h) + 4bsvh \right) \times 3 = 3 \left(b\ell(24sh^2 + 4s^2h) + 4bsvh \right)$$

When h^2 dominates, the above FLOPs can be simplified as $72bs\ell h^2$

When h^2 dominates, the parameters can be simplified as $P = 12\ell h^2$

Since $T = bs$ is the number of tokens, we thus have FLOPs = $6TP$

Example: GPT3-175B

GPT-175B: 175B parameters, 300B tokens

$$6 \times 174600 \times 10^6 \times 300 \times 10^9 = 3.1428 \times 10^{23} flops$$

Model	Total train compute (PF-days)	Total train compute (flops)	Params (M)	Training tokens (billions)	Flops per param per token	Mult for bwd pass	flops per active param per token	params active for each token
T5-Small	2.08E+00	1.80E+20	60	1,000	3	3	1	0.5
T5-Base	7.64E+00	6.60E+20	220	1,000	3	3	1	0.5
T5-Large	2.67E+01	2.31E+21	770	1,000	3	3	1	0.5
T5-3B	1.04E+02	9.00E+21	3,000	1,000	3	3	1	0.5
T5-11B	3.82E+02	3.30E+22	11,000	1,000	3	3	1	0.5
BERT-Base	1.89E+00	1.64E+20	109	250	6	3	2	1.0
BERT-Large	6.16E+00	5.33E+20	355	250	6	3	2	1.0
RoBERTa-Base	1.74E+01	1.50E+21	125	2,000	6	3	2	1.0
RoBERTa-Large	4.93E+01	4.26E+21	355	2,000	6	3	2	1.0
GPT-3 Small	2.60E+00	2.25E+20	125	300	6	3	2	1.0
GPT-3 Medium	7.42E+00	6.41E+20	356	300	6	3	2	1.0
GPT-3 Large	1.58E+01	1.37E+21	760	300	6	3	2	1.0
GPT-3 XL	2.75E+01	2.38E+21	1,320	300	6	3	2	1.0
GPT-3 2.7B	5.52E+01	4.77E+21	2,650	300	6	3	2	1.0
GPT-3 6.7B	1.39E+02	1.20E+22	6,660	300	6	3	2	1.0
GPT-3 13B	2.68E+02	2.31E+22	12,850	300	6	3	2	1.0
GPT-3 175B	3.64E+03	3.14E+23	174,600	300	6	3	2	1.0