北京大学
PEKING UNIVERSITY

Parameters, Memories, and Computations in Transformers

Kun Yuan

Center for Machine Learning Research @ Peking University

北京大学
PEKING UNIVERSITY

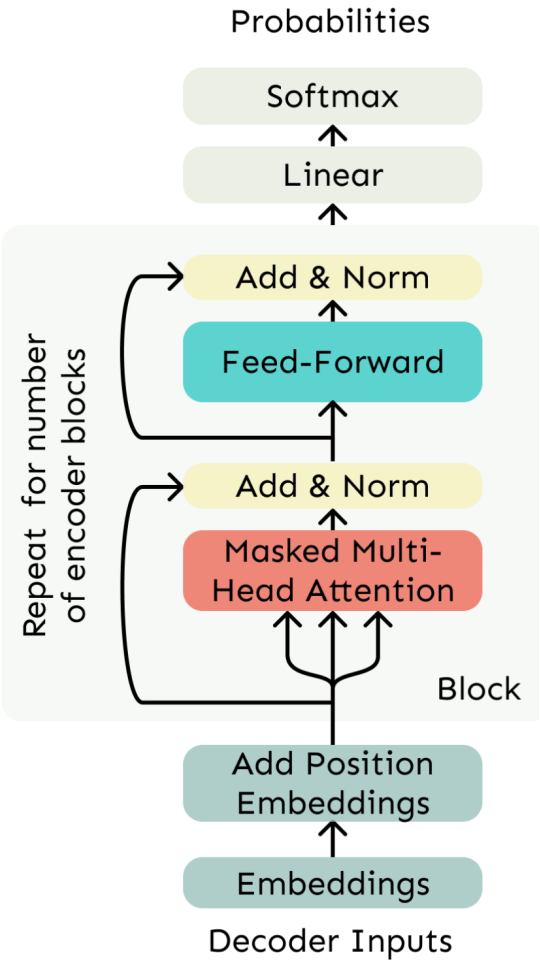
PART 01

Settings and Basics

Decoder-only Transformer

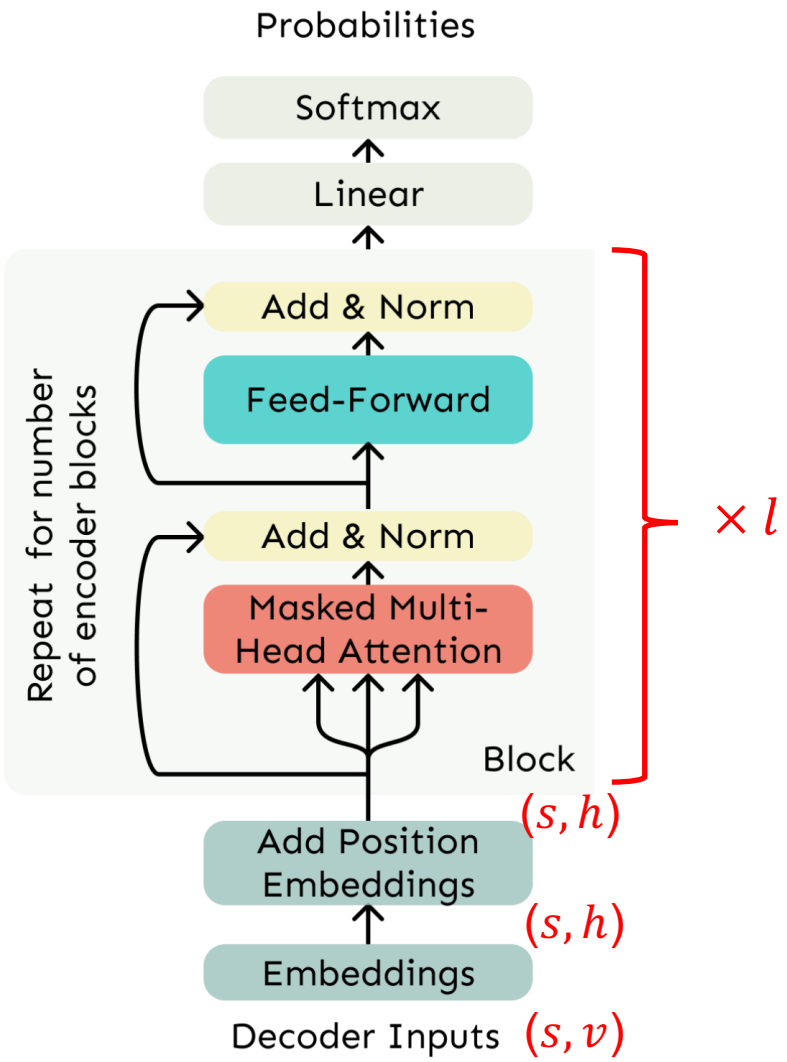
北京大学 PEKING UNIVERSITY

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

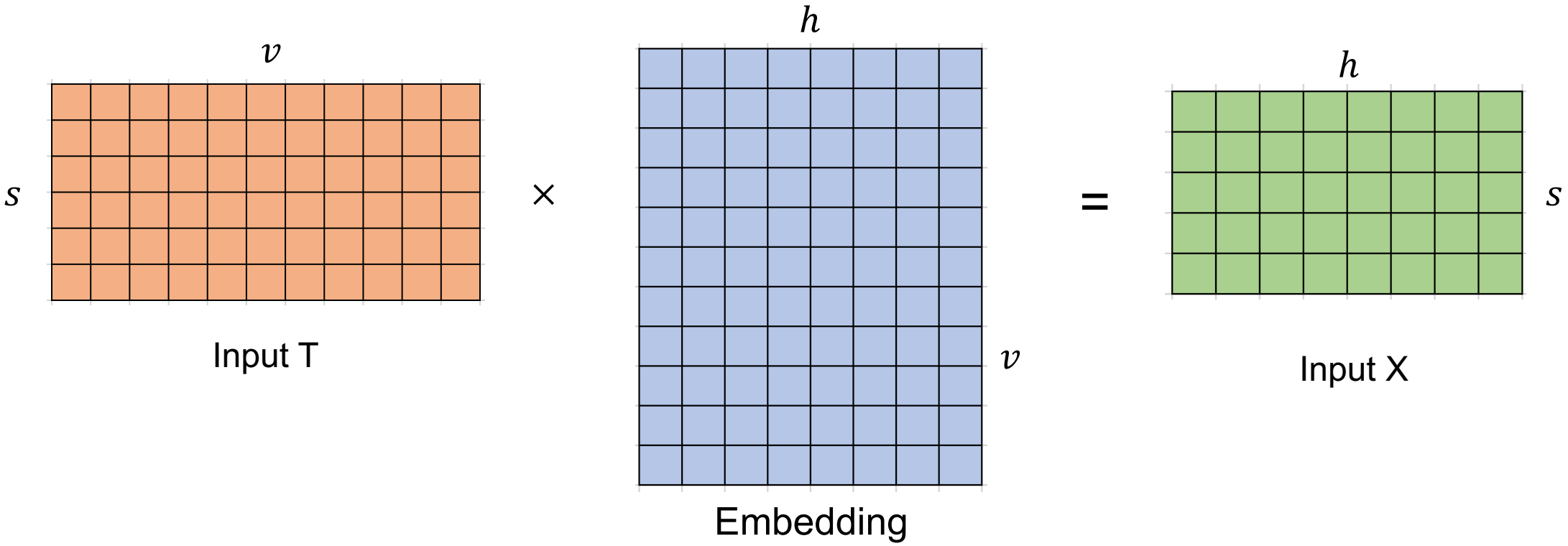


Notations

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

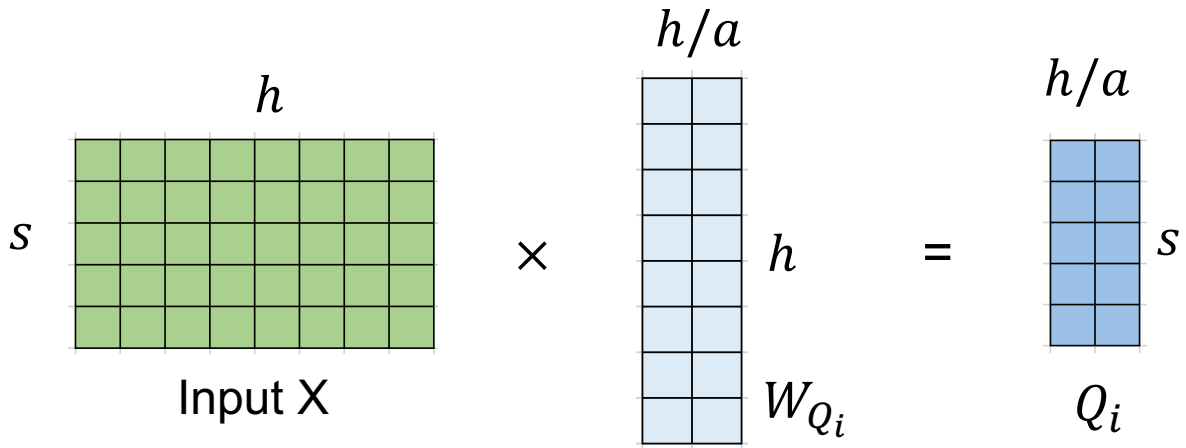


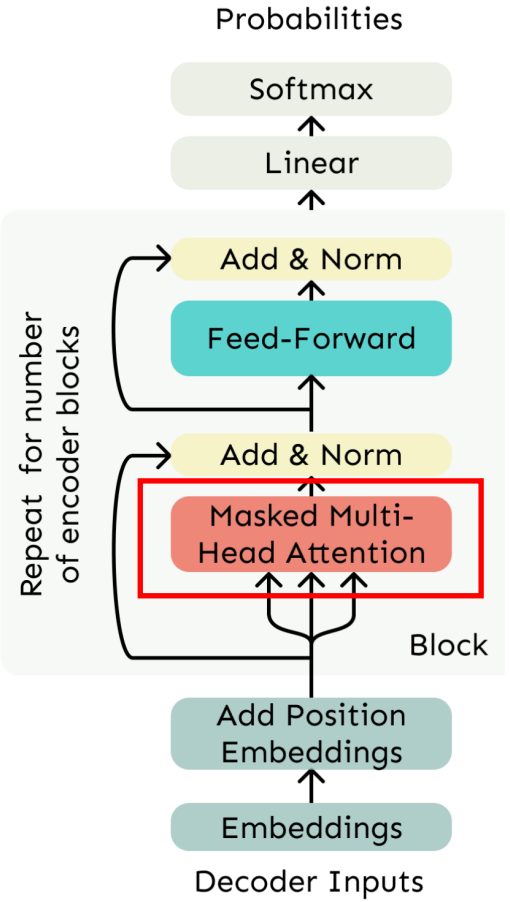
Embedding



Self-attention

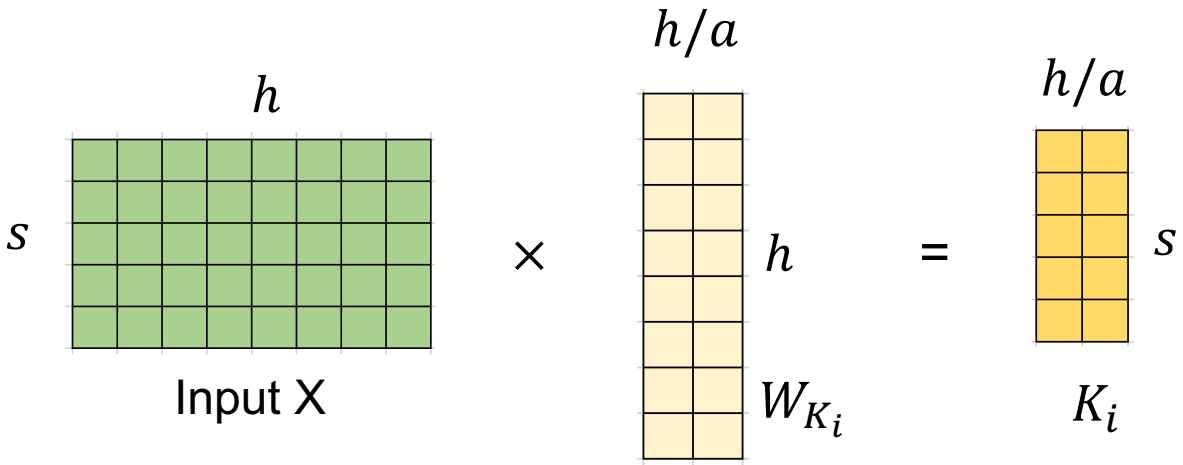
- Number of heads: a
- Dims of each W_{Q_i}, W_{K_i} and W_{V_i}: $h \times \frac{h}{a}$





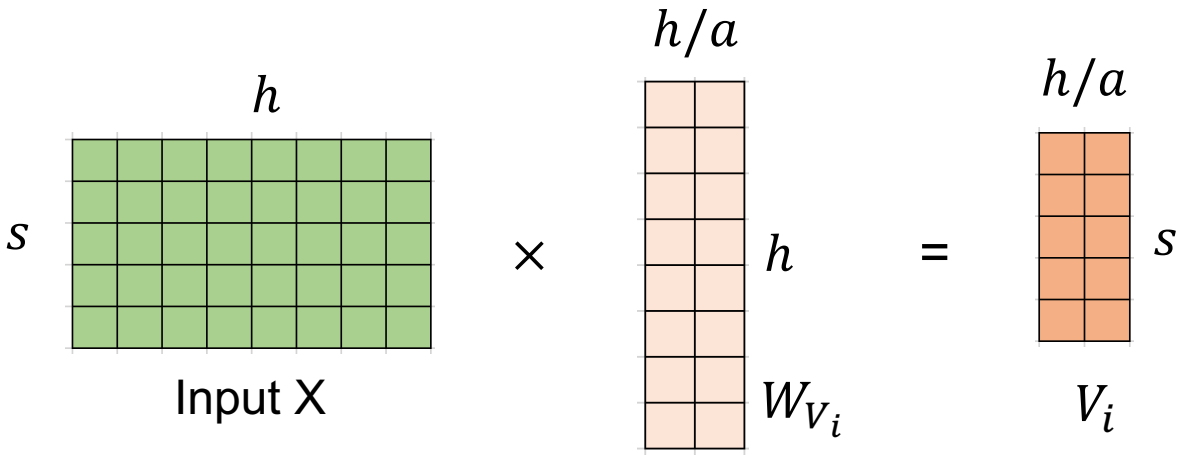
Self-attention

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



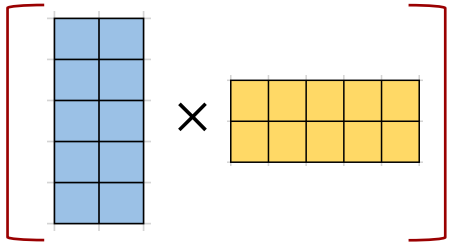
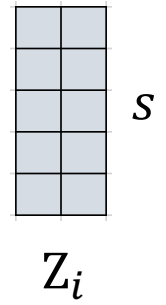
Self-attention

- 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}: $h \times \frac{h}{a}$

$$\text{softmax}(\frac{Q_i K_i^T}{\sqrt{h/a}})V_i$$ = softmax  = 

h/a

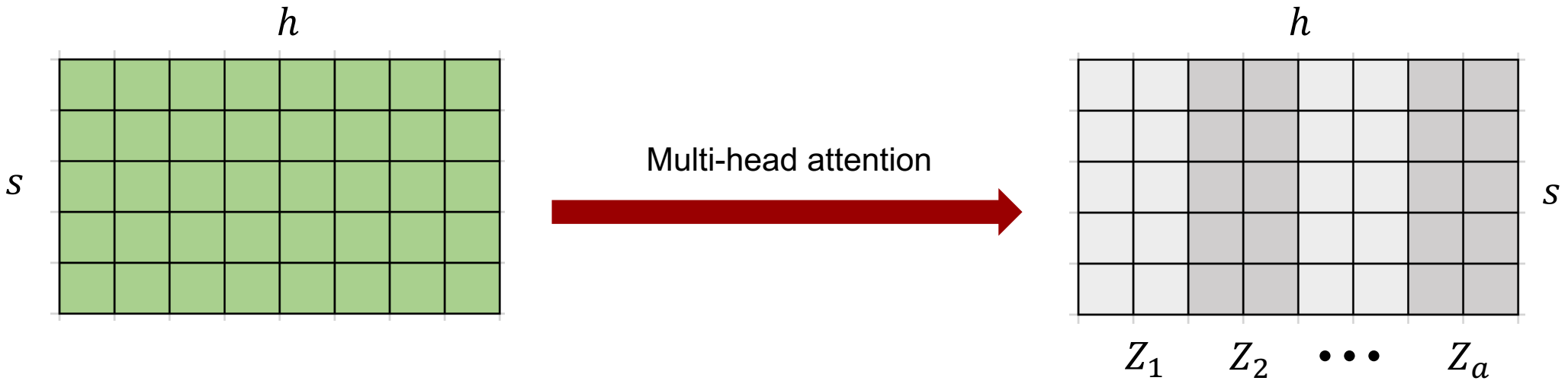
s

Z_i

One-head attention

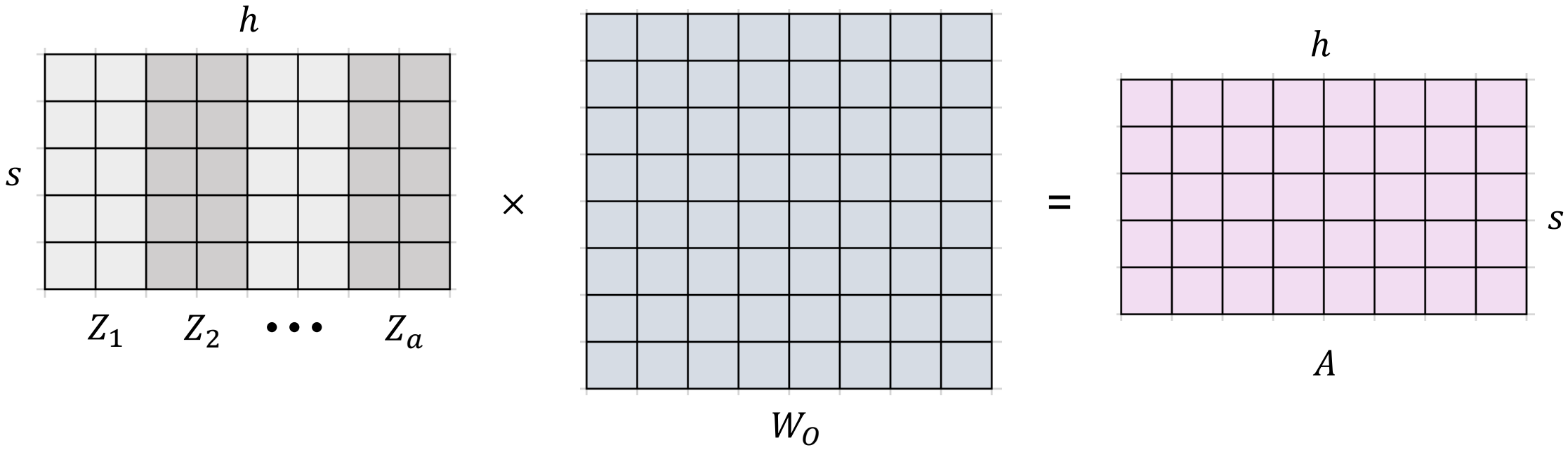
Multi-head attentions

- 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

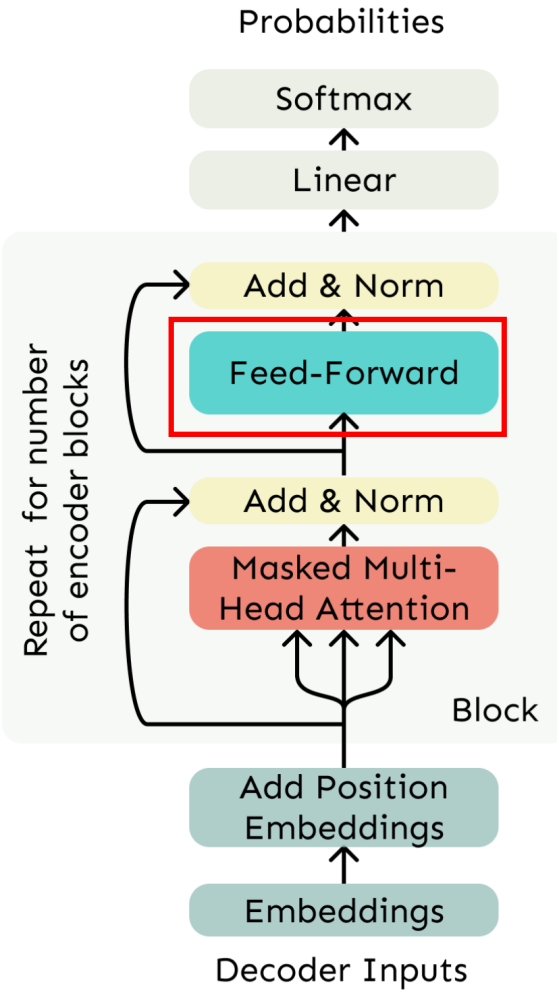
- Number of heads: a
- Dims of each W_O: $h \times h$
- Dims of each W_{Q_i}, W_{K_i} and W_{V_i}: $h \times \frac{h}{a}$



Feed-forward Layer

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

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

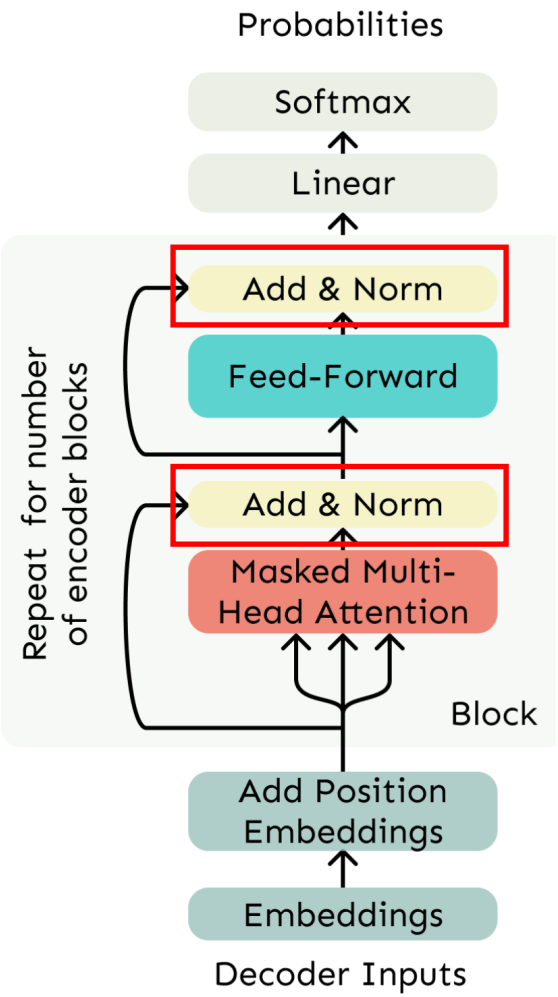


Layer normalization

- Then layer normalization computes:

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

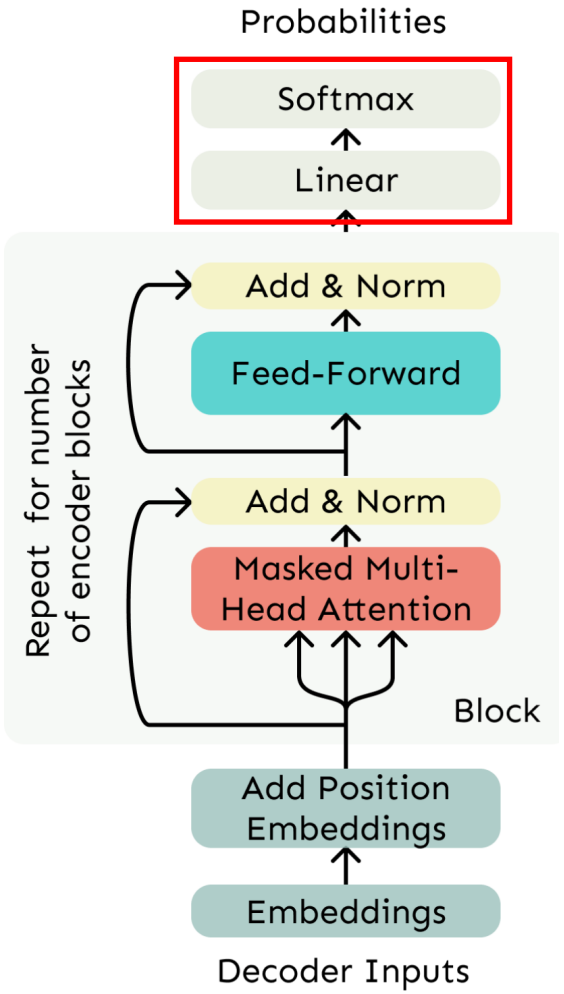
- Dims of γ and β: h



Probability prediction

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

- Dims of W_v: $h \times \text{v}$

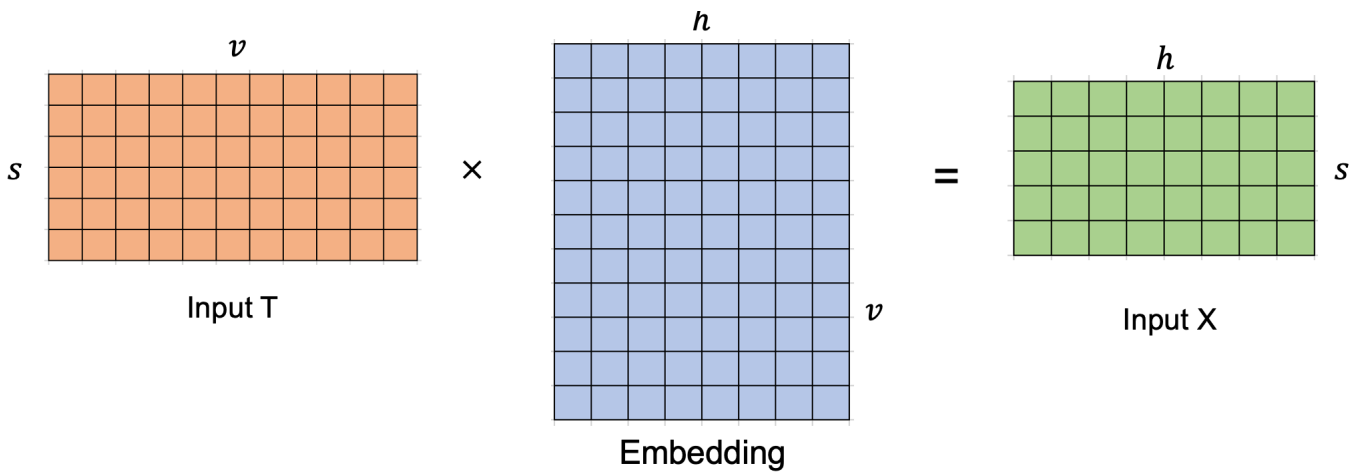


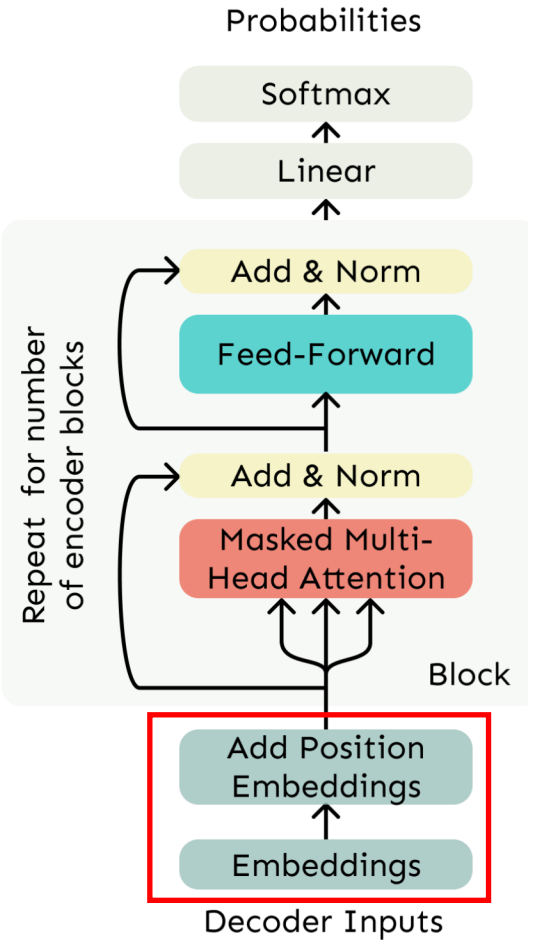
北京大学
PEKING UNIVERSITY

PART 02

Parameters analysis

Embedding



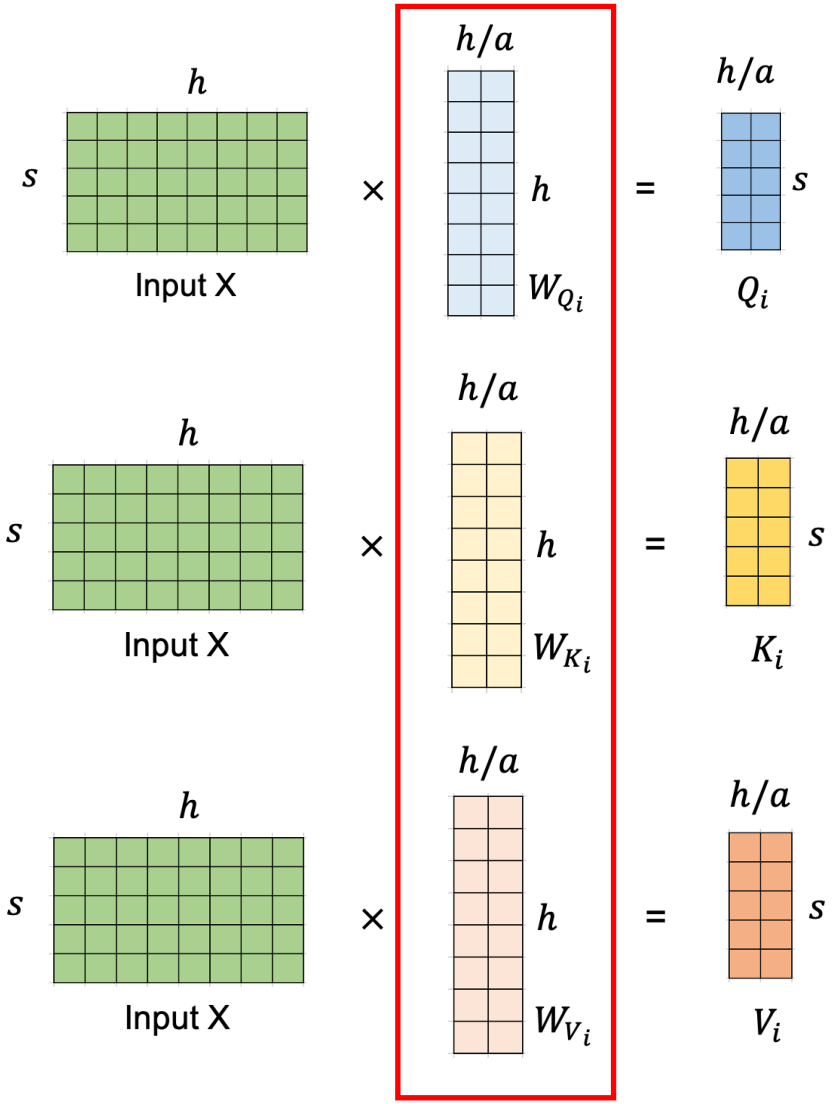


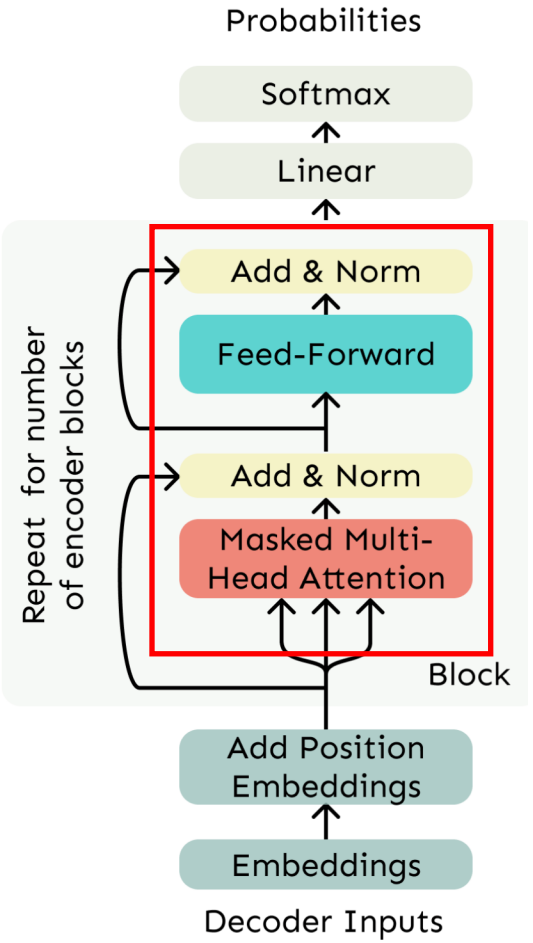
- We need to store the embedding with parameters vh
- Position embedding can be ignored when using RoPE and ALiBi

Multi-head attentions

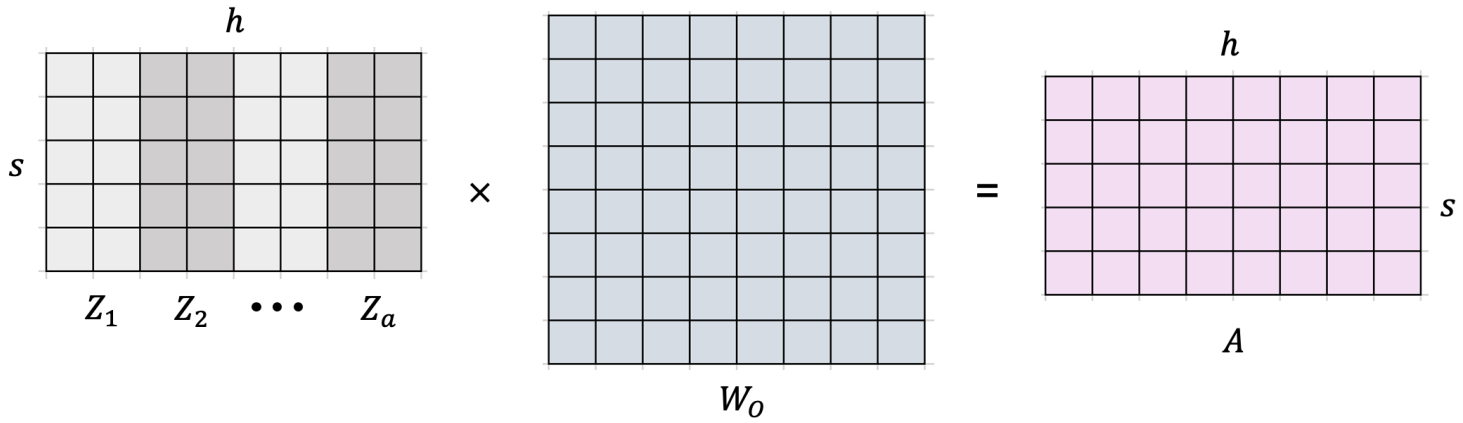
- We need to store W_Q, W_K and W_V

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

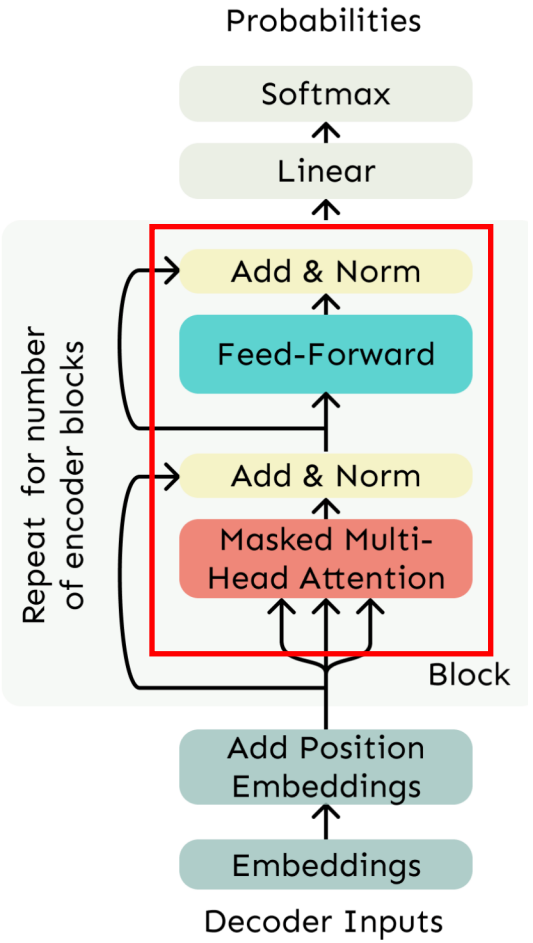




Multi-head attentions



- We need to store W_O: h^2



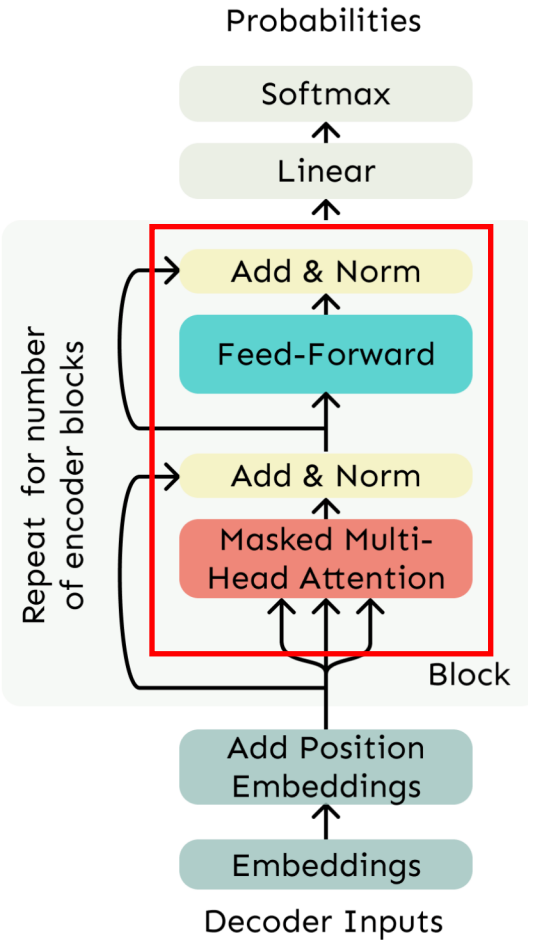
Layer normalization

- Then layer normalization computes:

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

- Dims of γ and β: h

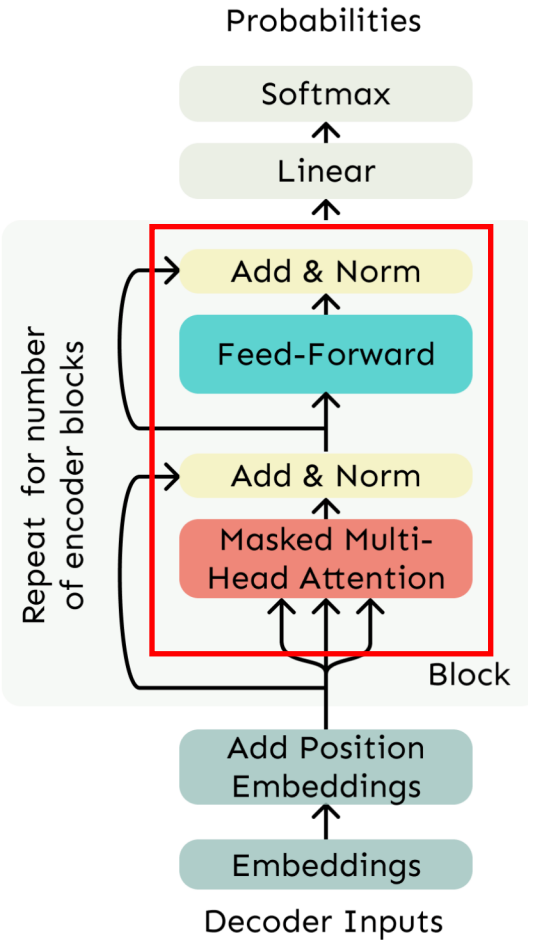
- We should store γ and β. Since their parameters are much smaller than h^2 and vh, we can ignore them



Feed-forward layers

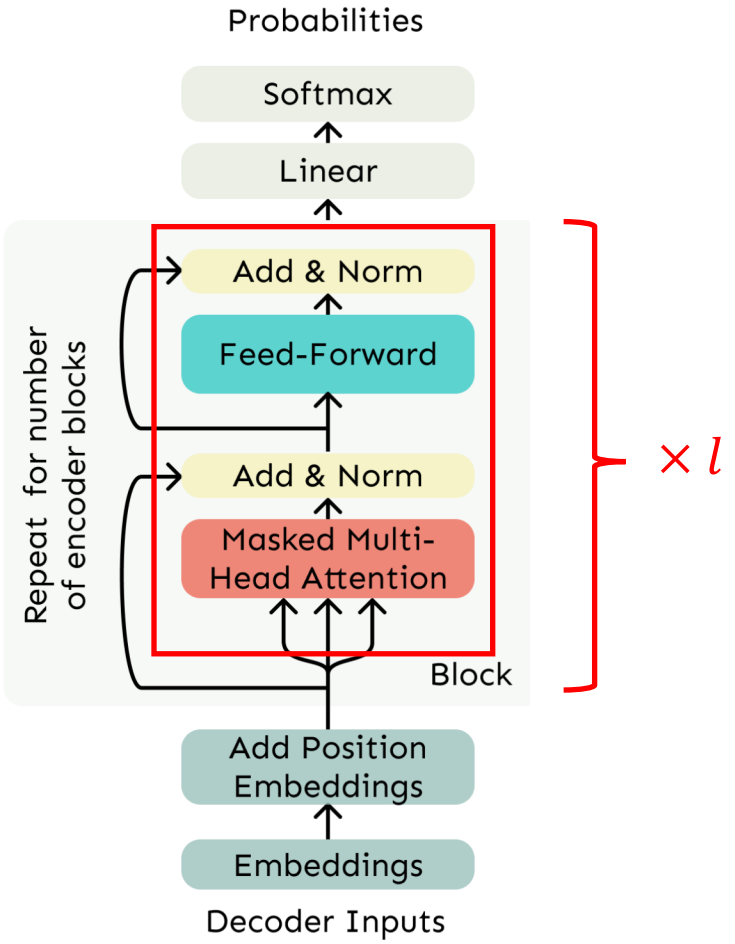
$$X' = \mathrm{ReLU}(A \cdot W_1 + b_1) \cdot W_2 + b_2$$

- 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

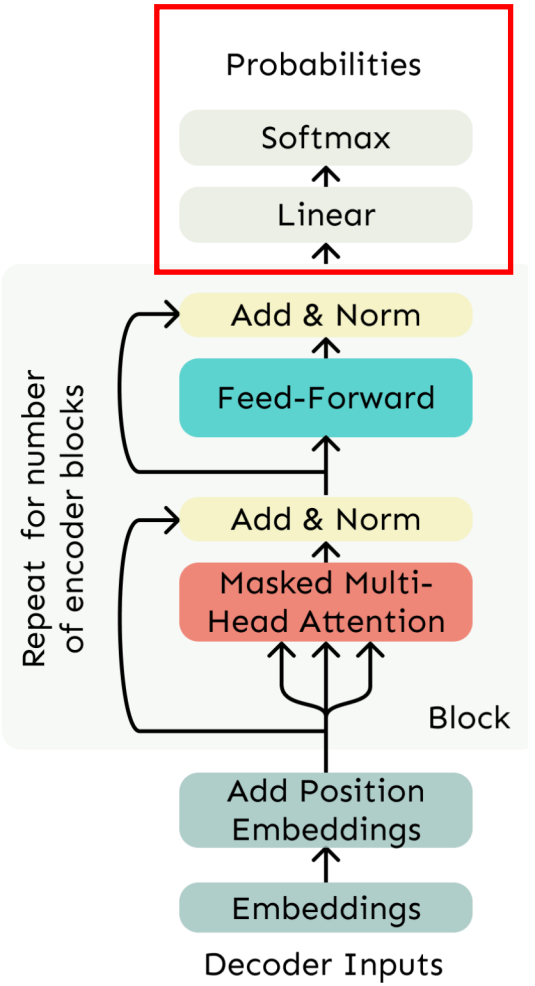
- Multi-head attentions: $4h^2$
- Feed-forward layers : $8h^2$
- l layers of attentions : $(4h^2 + 8h^2) \times l = 12lh^2$



Probability predictions

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

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

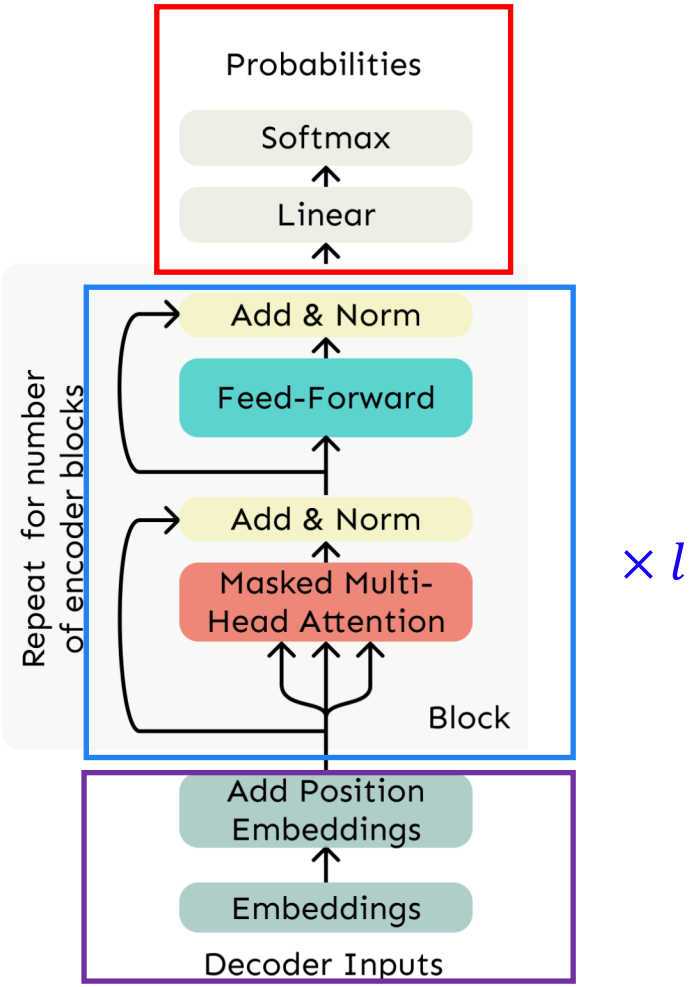


Total parameters

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

Total parameters:

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



Example: LLaMA parameters

北京大学 PEKING UNIVERSITY

- 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

北京大学
PEKING UNIVERSITY

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

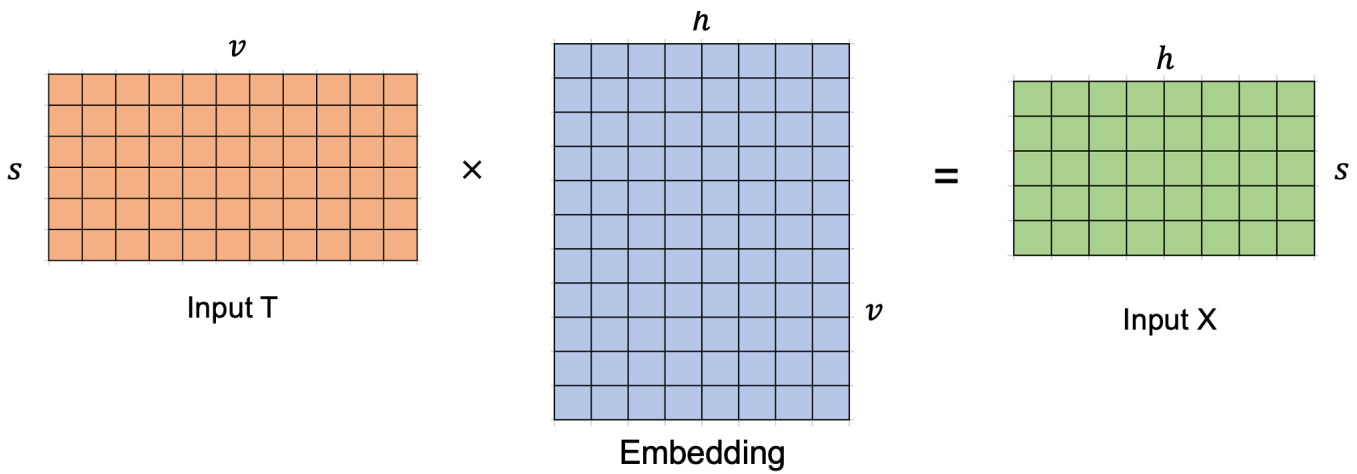
mnp additions

mnp multiplications

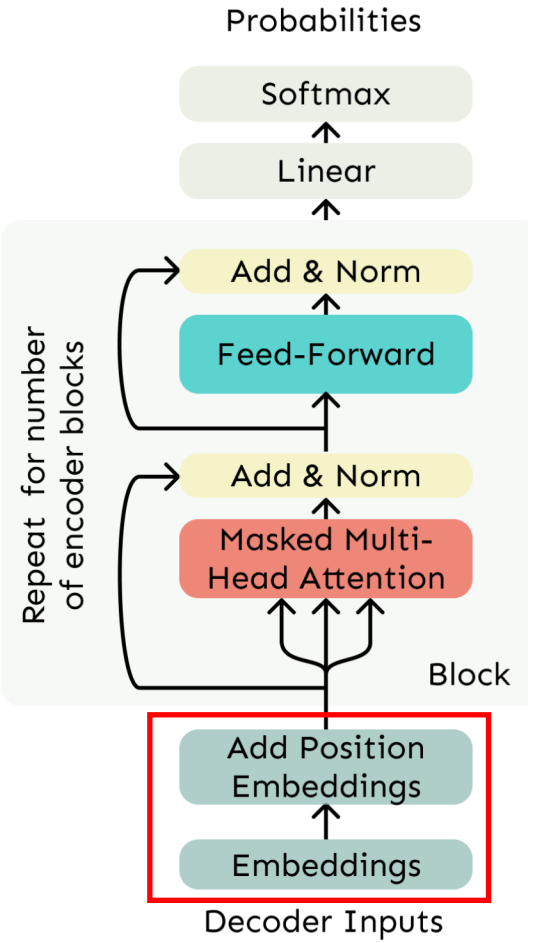
} $2mnp$ FLOPs

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

Embedding



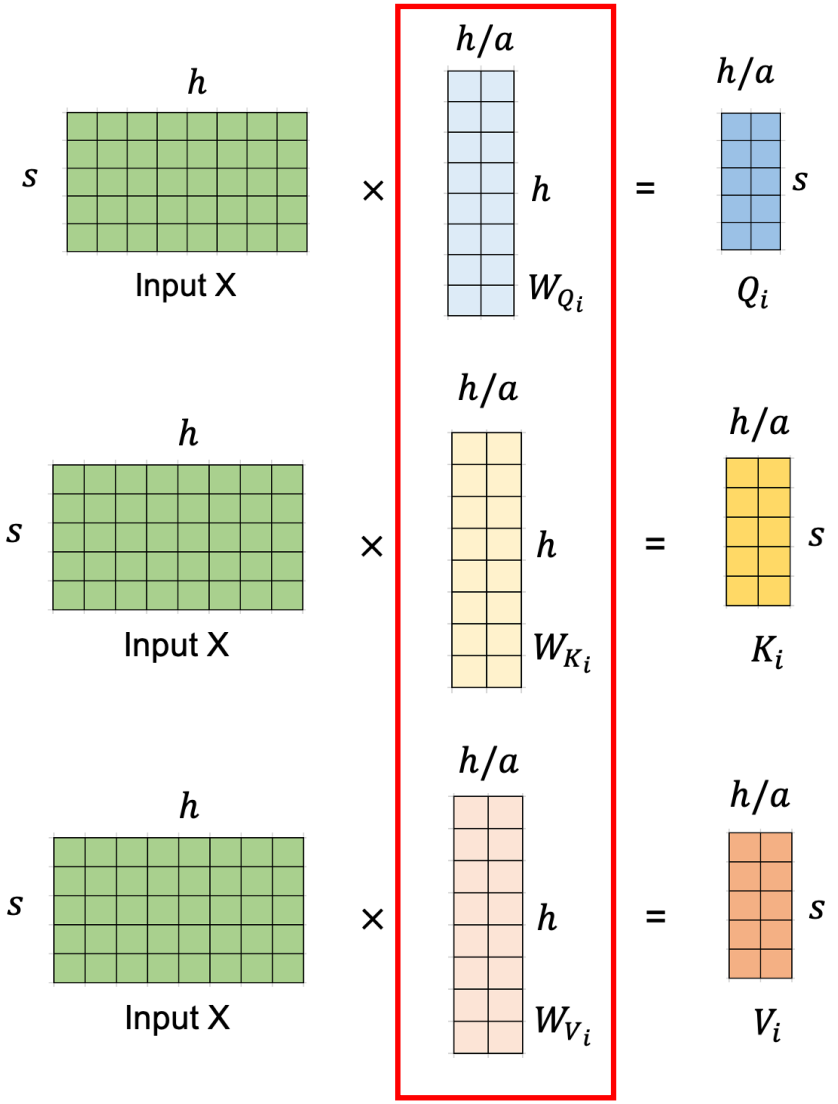
- Word embedding: $2svh$

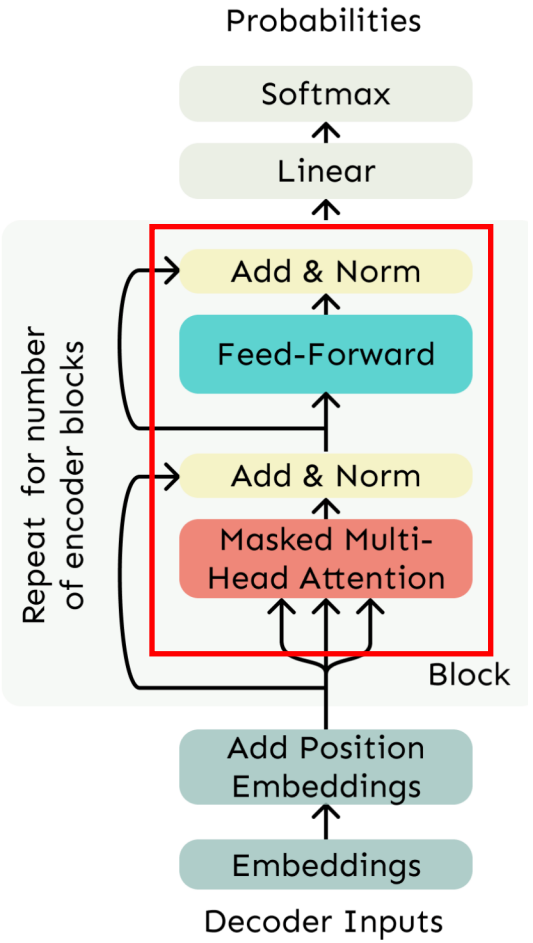


Multi-head attentions

- Multi-head attentions

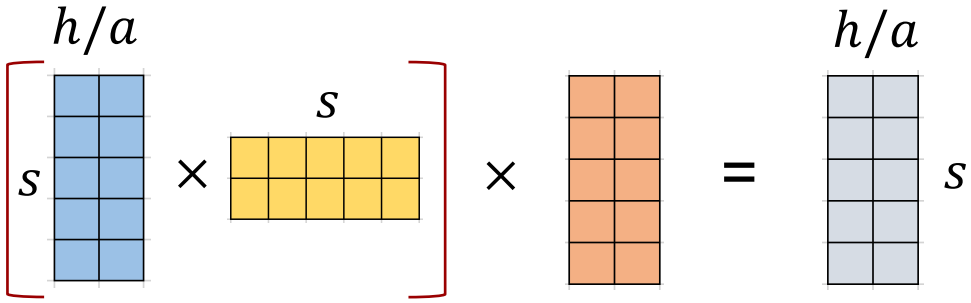
$$6(sh^2/a) \times a = 6sh^2$$





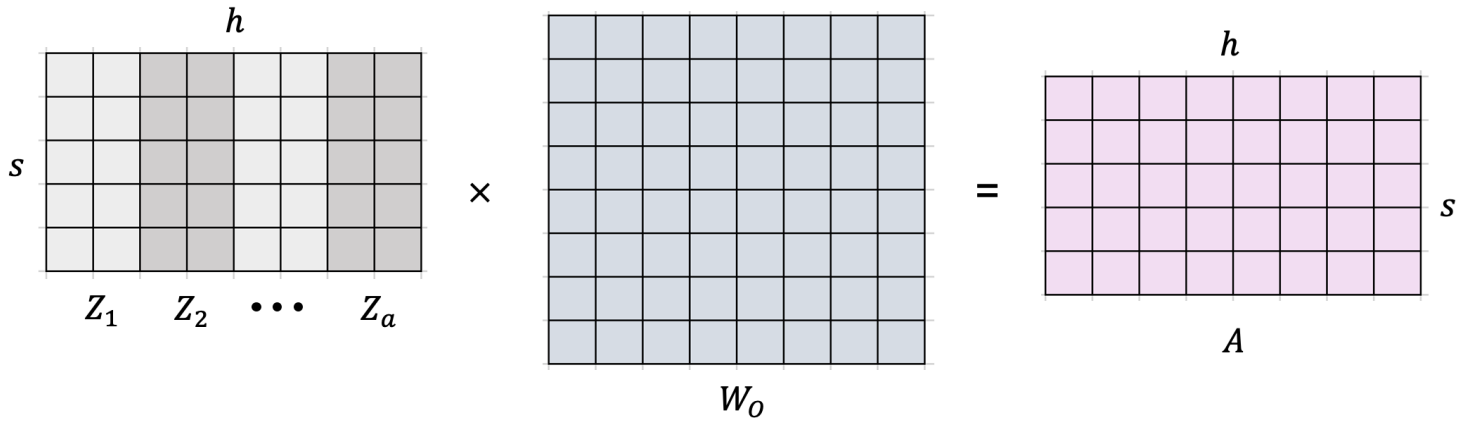
Multi-head attentions

$$\text{softmax}(\frac{Q_i K_i^T}{\sqrt{h/a}})V_i = \text{softmax}\left[s \begin{array}{|c|c|}\hline \\ \\ \\ \\ \\ \hline \end{array} \times \begin{array}{|c|c|c|c|c|}\hline \\ \\ \\ \\ \\ \hline \end{array} \right] \times \begin{array}{|c|c|}\hline \\ \\ \\ \\ \\ \hline \end{array} = \begin{array}{|c|c|}\hline \\ \\ \\ \\ \\ \hline \end{array} \; s$$

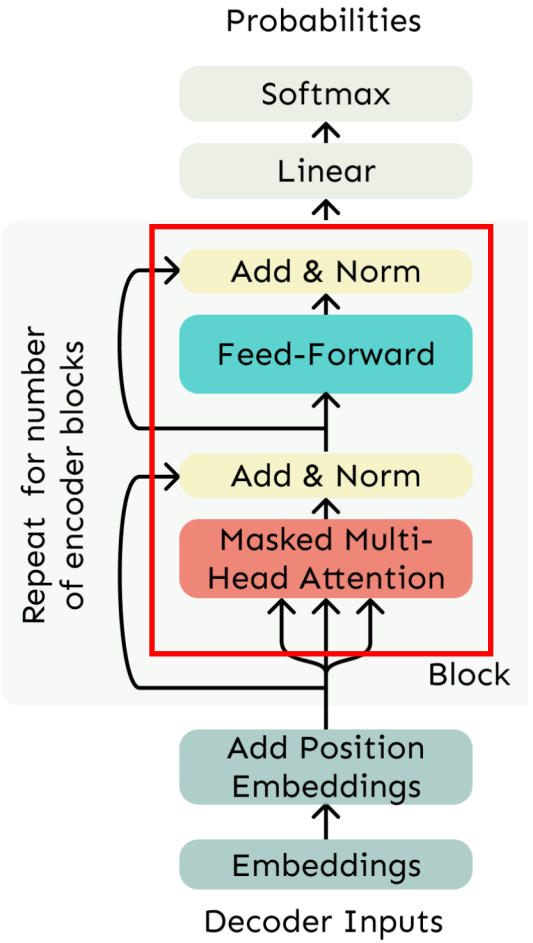


$$(2s^2h/a + 2s^2h/a) \times a = 4s^2h$$

Multi-head attentions



$2sh^2$ Flops

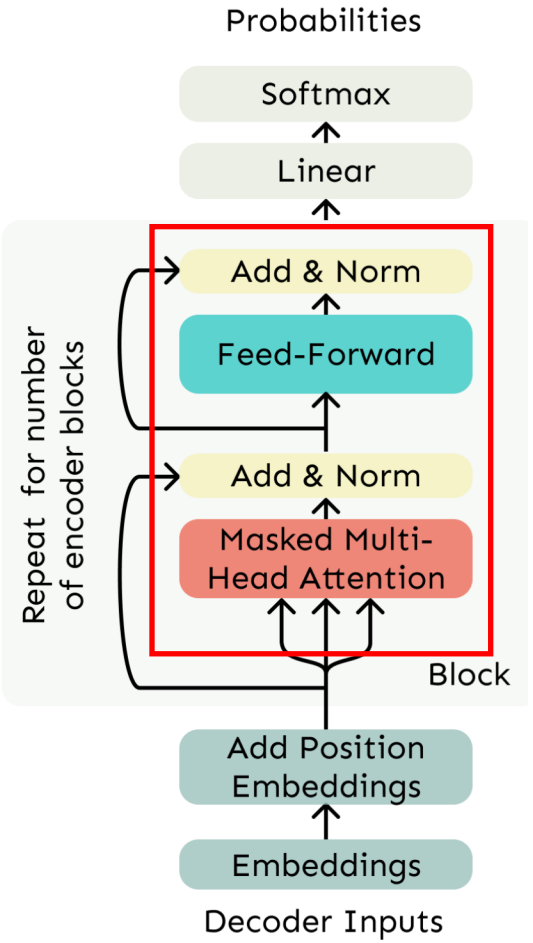


Feed-forward layers

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

- 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: $8sh^2$

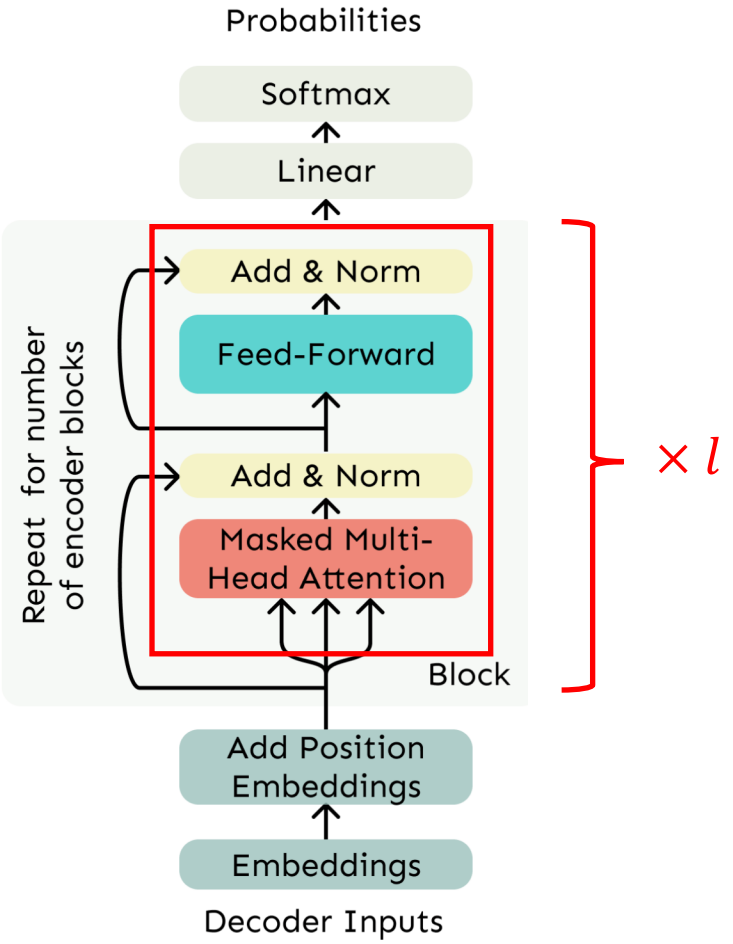
$16sh^2$



Transformer block

- Multi-head attentions: $8sh^2 + 4s^2\mathrm{h}$
- Feed-forward layers : $16sh^2$
- l layers of attentions :

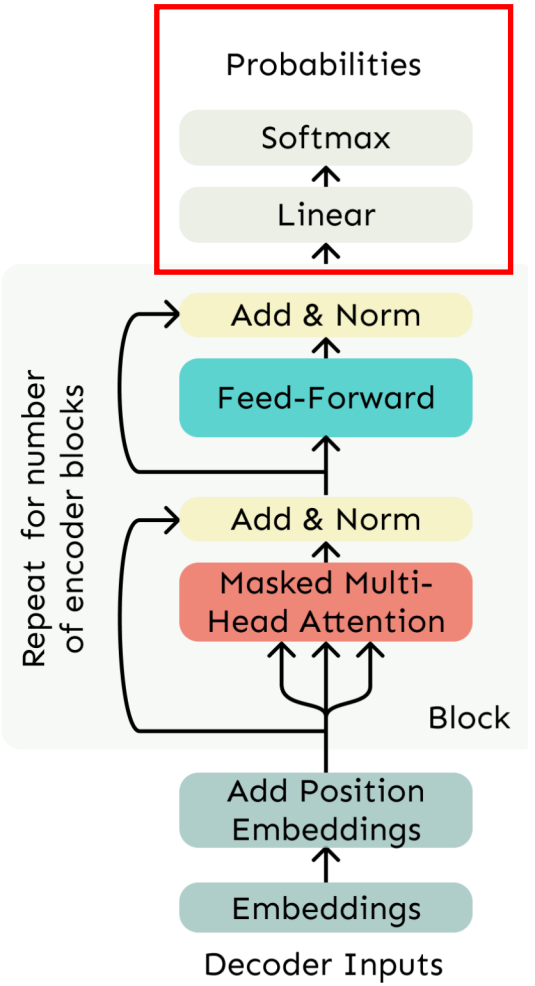
$$(8sh^2 + 16sh^2 + 4s^2h\,)\times l = 24slh^2 + 4s^2lh$$



Probability predictions

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

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



Total forward FLOPs

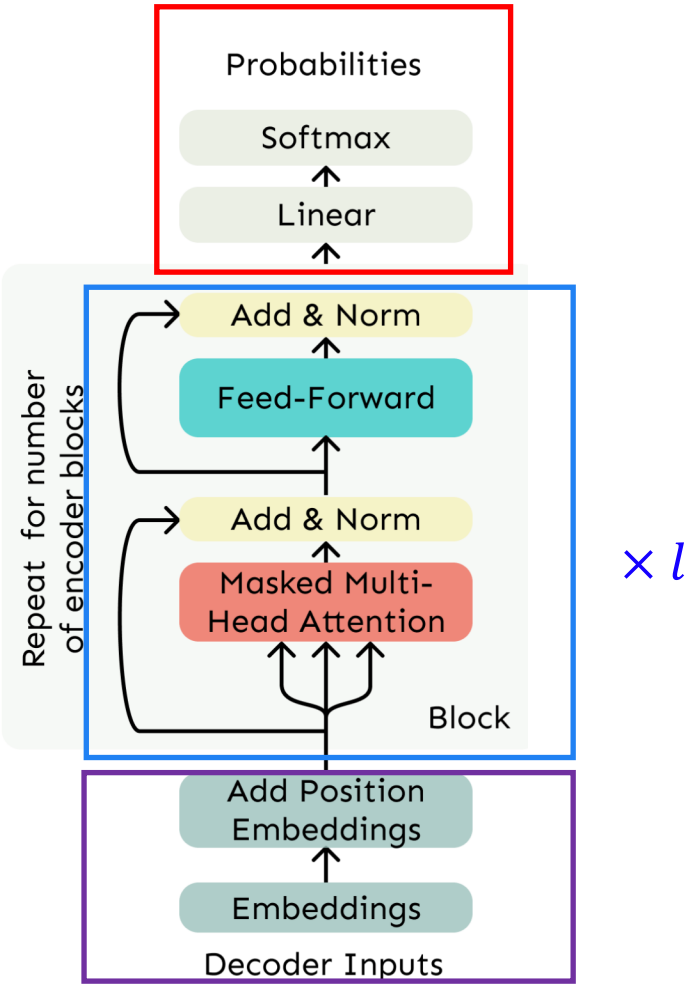
- Embeddings: $2svh$
- Attention blocks: $24lsh^2 + 4s^2lh$
- Probability predictions: $2svh$

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 $\text{FLOPs} = 6TP$

Example: GPT3-175B

北京大学 PEKING UNIVERSITY

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

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

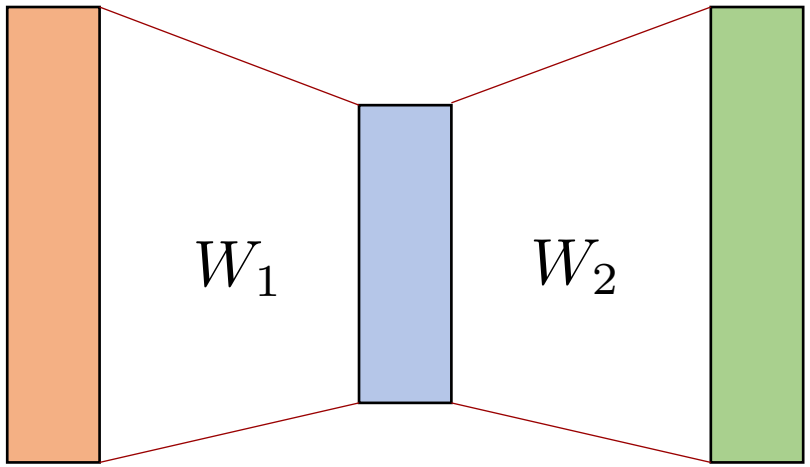
Model	Total train compute (PF-days)	Total train compute (flops)	Params (M)	Training tokens (billions)	Flops per param per token	Mult for bwd pass	flops per active param per token	Frac of params active for each token
T5-Small	2.08E+00	1.80E+20	60	1,000	3	3	1	0.5
T5-Base	7.64E+00	6.60E+20	220	1,000	3	3	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

北京大学
PEKING UNIVERSITY

PART 04

Memory analysis

Warmup: Linear neural network



dims: d (p,d) p (q,p) q

$$h = W_1 x$$

$$z = \sigma(h)$$

$$\hat{y} = W_2 z$$

$$f = L(\hat{y})$$

Forward

$$\frac{\partial f}{\partial W_1} = \frac{\partial f}{\partial h} x^T$$

$$\frac{\partial f}{\partial h} = \frac{\partial f}{\partial z} \odot \nabla\sigma(h)$$

$$\frac{\partial f}{\partial W_2} = \frac{\partial L}{\partial \hat{y}} z^T, \quad \frac{\partial f}{\partial z} = W_2 \frac{\partial L}{\partial \hat{y}}$$

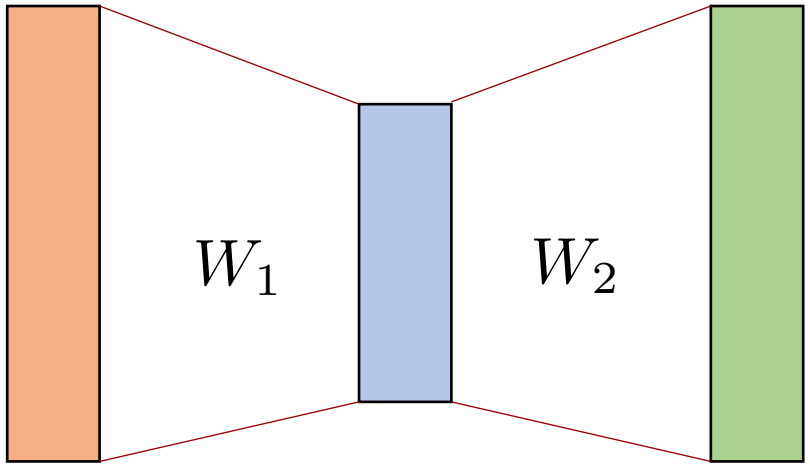
$$\frac{\partial f}{\partial \hat{y}} = \nabla L(\hat{y})$$

Backward

Store h, z and $\hat{y}$

Store ∇W_1 and ∇W_2

Warmup: Linear neural network with batch size



dims: d (p,d) p (q,p) q

$$h_b = W_1\, x_b$$

$$z_b = \sigma(h_b)$$

$$\hat{y}_b = W_2\, z_b$$

$$f = \frac{1}{B}\sum_{b=1}^{B} L(\hat{y}_b)$$

Forward

$$\frac{\partial f}{\partial W_1} = \frac{1}{B}\sum_{b=1}^{B} \frac{\partial f}{\partial h_b} x_b^T,$$

$$\frac{\partial f}{\partial h_b} = \frac{\partial f}{\partial z_b} \odot \nabla\sigma(h_b)$$

$$\frac{\partial f}{\partial W_2} = \frac{1}{B}\sum_{b=1}^{B} \frac{\partial L}{\partial \hat{y}_b} z_b^T, \quad \frac{\partial f}{\partial z_b} = W_2 \frac{\partial f}{\partial \hat{y}_b}$$

$$\frac{\partial f}{\partial \hat{y}_b} = \frac{\partial L}{\partial \hat{y}_b}$$

Backward

Store $\{\mathbf{h_b}, \mathbf{z_b}, \hat{\boldsymbol{y}}_b\}_{b=1}^{B}$

Store $\nabla \mathbf{W_1}$ and $\nabla \mathbf{W_2}$

Memory decomposition

北京大学 PEKING UNIVERSITY

Memory = Model + Gradient + Optimizer states + Activations

- Given a model with P parameters, gradient will consume P parameters, and Optimizer states will consume 2P parameters; **4P parameters in total.**
- When using FP32 to store parameters, each parameter takes **4** Bytes
- When using FP16 or BF16 to store parameters, each parameter takes **2** Bytes

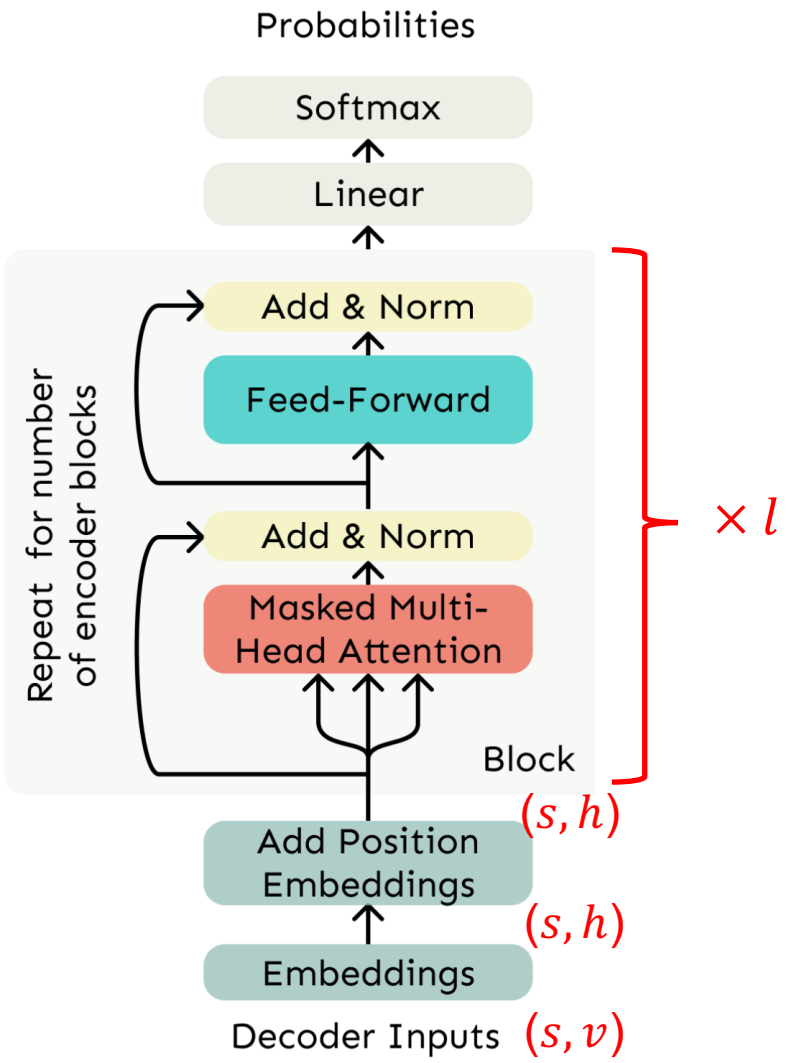
Decoder-only Transformers

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

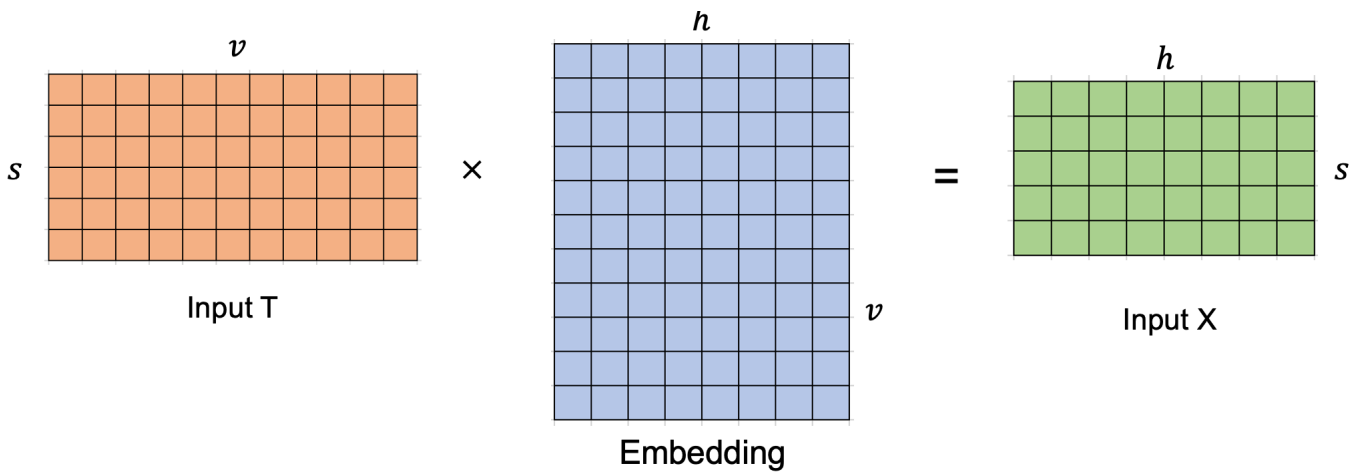
Parameters $P = 12\ell h^2 + 2vh$

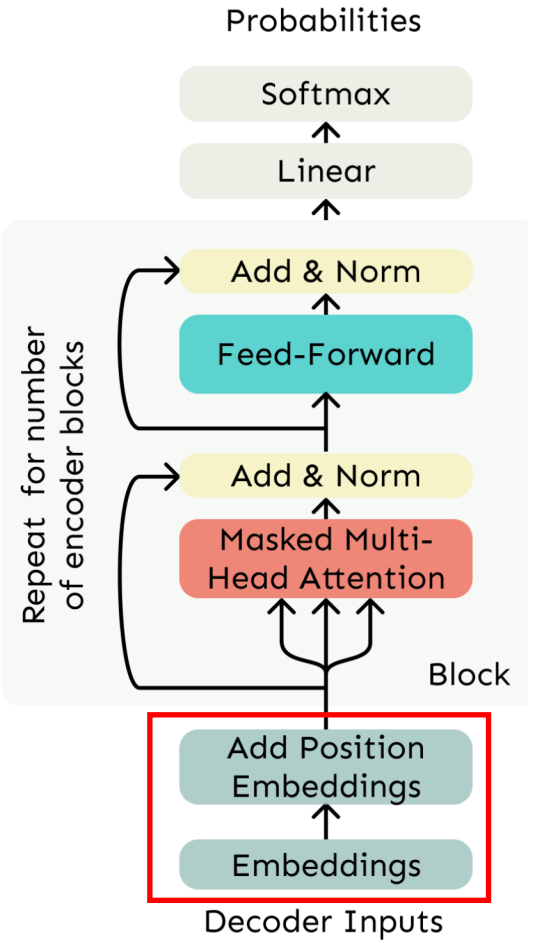
Model + Gradient + Optimizer states = 4P * 4 (Bytes)

Next we estimate activations



Embedding



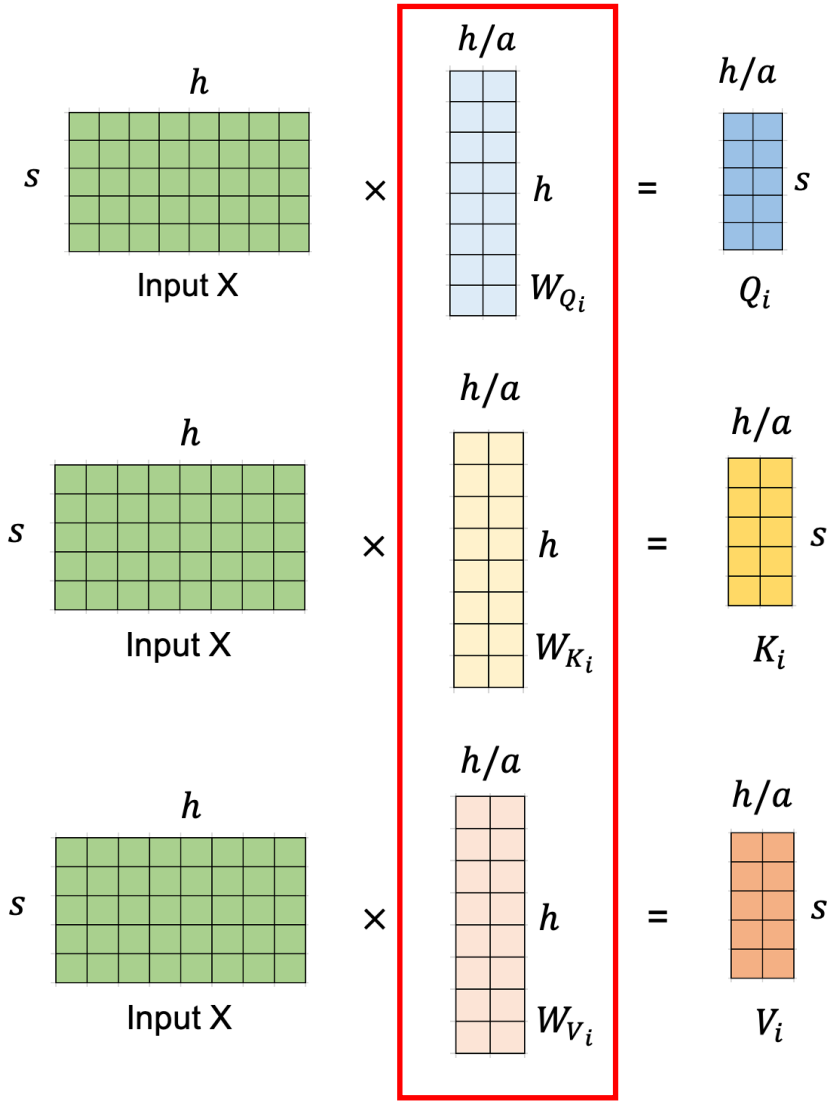


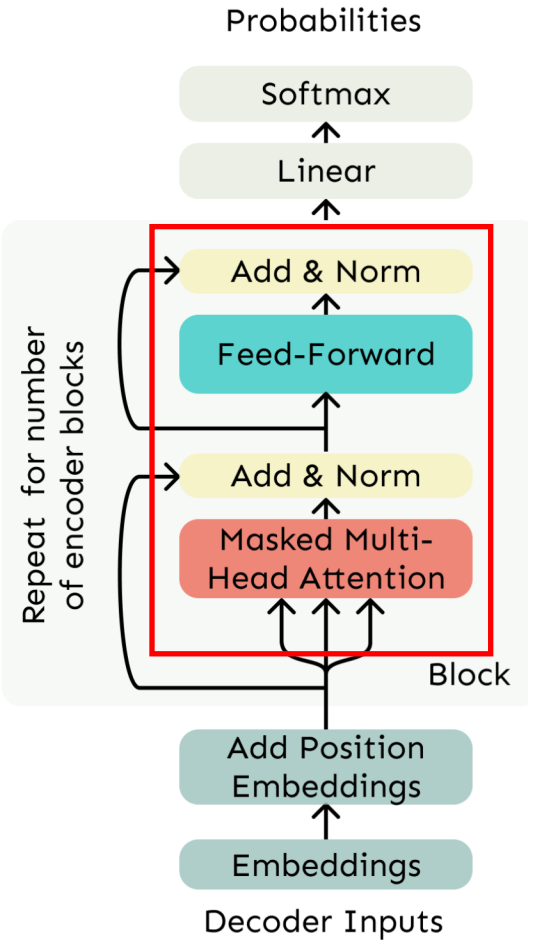
- We need to store the embedding activations with parameters sh
- Position embedding can be ignored when using RoPE and ALiBi

Multi-head attentions

- We need to store Q_i, K_i and V_i

$$3(sh/a) \times a = 3sh$$





Self-attention

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

h/a

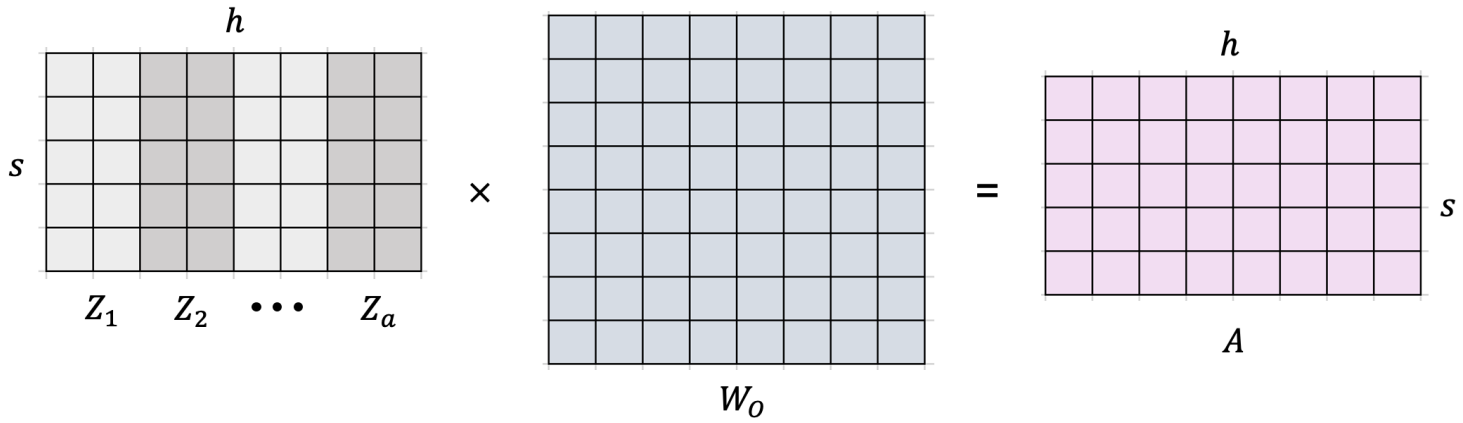
Z_i

One-head attention

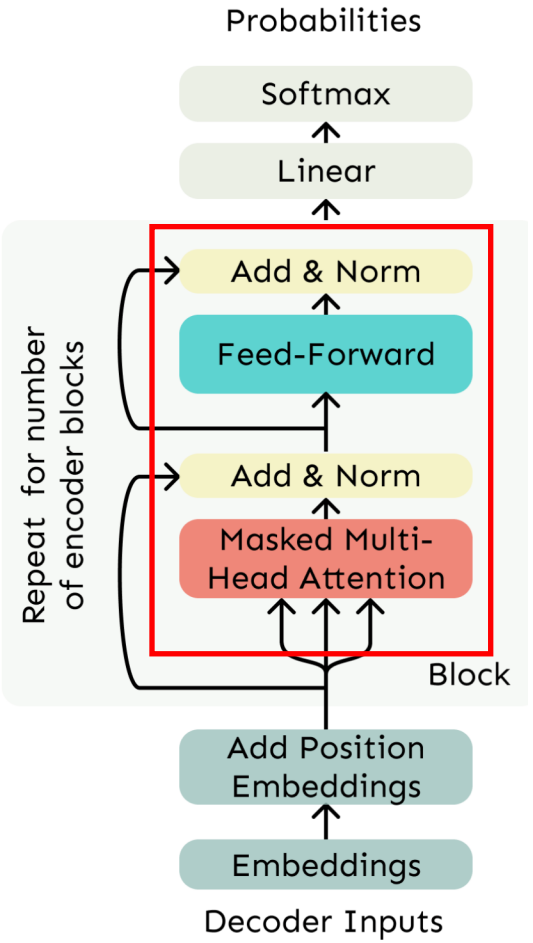
- Store $Q_i K_i^T$ with s^2 parameters;
- Store softmax($Q_i K_i^T$) with s^2 parameters
- Store $\text{softmax}(\frac{Q_i K_i^T}{\sqrt{h/a}})V_i$ with sh/a parameters

} Store $2s^2a + sh$ activations

Multi-head attentions



- We need to store A with sh parameters



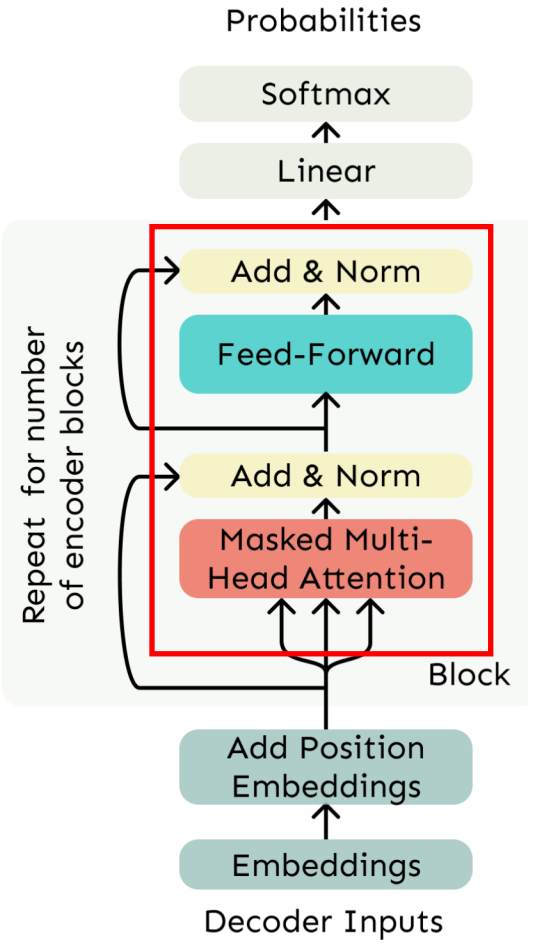
Layer normalization

- Then layer normalization computes:

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

- Dims of γ and β: h

- We can ignore the layer normalization



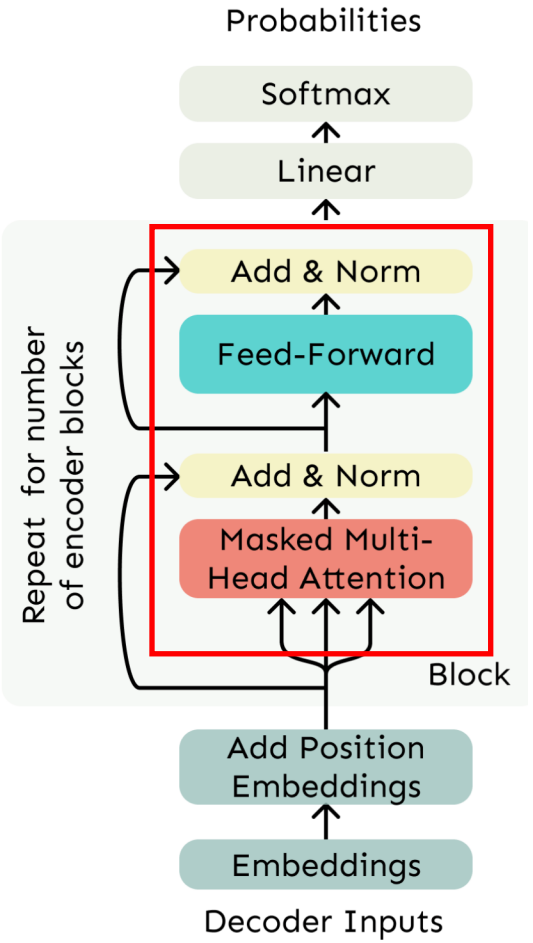
Feed-forward layers

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

- Store AW_1 with 4sh parameters
- Store ReLU(AW_1) with 4sh parameters
- Store ReLU(AW_1)$\cdot W_2$ with sh parameters

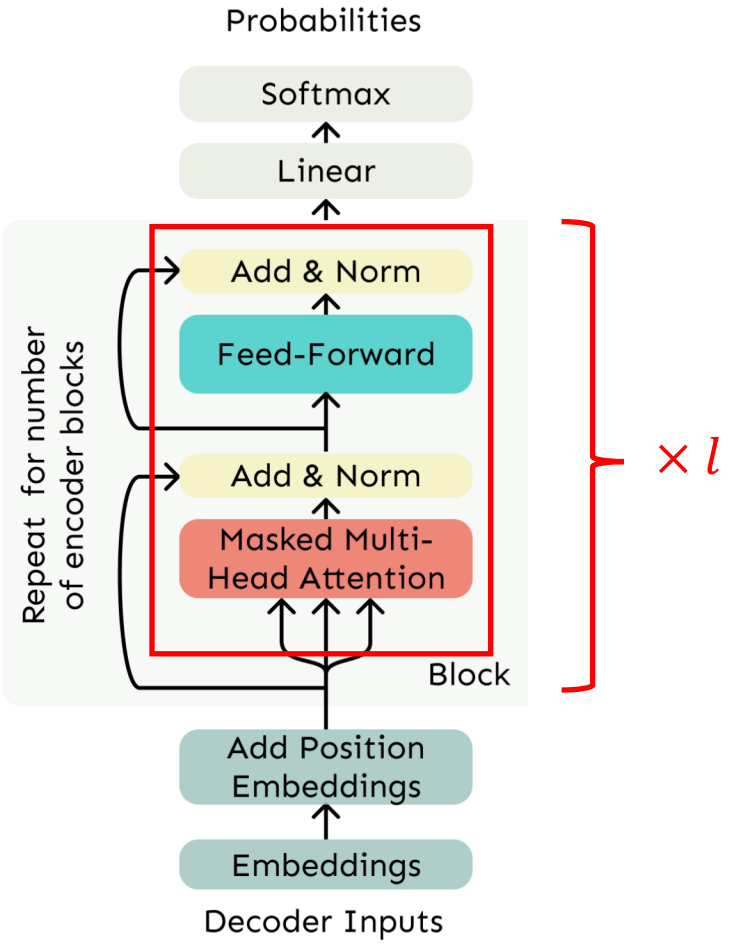
9sh

- The activations of b_1 and b_2 can be ignored



Transformer block

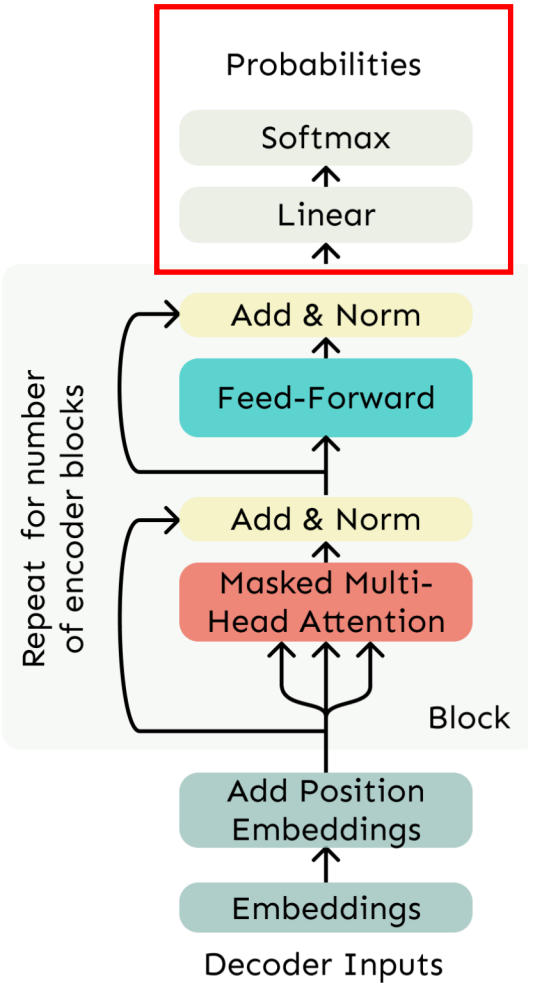
- Multi-head attentions: $5sh + 2s^2a$
- Feed-forward layers : $9sh$
- l layers of attentions : $(2s^2a + 14sh) \times l$



Probability predictions

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

- Store XW_v with sv parameters
- Store Softmax(XW_v) with sv parameters



Total activations

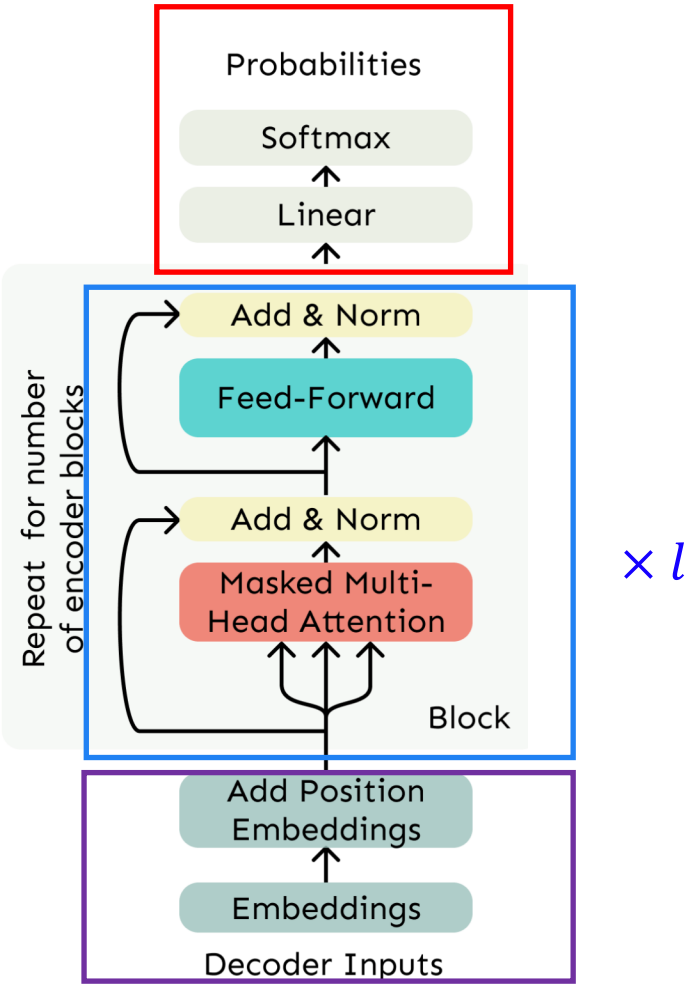
- Embedding activations: sh
- Self-attention activations: $(2s^2a + 14sh) \times l$
- Probability activations: $2sv$

Total activation parameters with batch-size 1:

$$(2s^2a + 14sh) \times l + 2sv + sh$$

Ignoring 2sv and sh and using batch-size b:

$$(2s^2a + 14sh) \times l \times b$$



Total memory

Memory = **Model + Gradient + Optimizer states** + **Activations**

$$\left(48\, l\, h^2 + bl\left(2s^2 a + 14sh\right)\right) \times 4 \text{ Bytes}$$

- When hidden state h is large, the model parameters dominate the memory
- When batch-size b or sequence length s is large, the activation dominates the memory

Examples

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

GPT3 has 175B parameters; its model consumes $4 \times 175 \times 10^9$ Bytes = 700 GB

Its gradient takes 700 GB parameters; Optimizer states take 1.4 TB

GPT has sequence length s = 2048. When b=1, its activation takes 444 GB, 63% of the model

When b=128, its activation is 81 times of the model size

Summary

Memory = **Model + Gradient + Optimizer states** + **Activations**

$$\left(48\, l\, h^2 + bl\left(2s^2 a + 14sh\right)\right) \times 4 \text{ Bytes}$$

- When hidden state h is large, the model parameters dominate the memory
- When batch-size b or sequence length s is large, the activation dominates the memory
- The activation-incurred memory **cannot be ignored**