人工智能编译器与大模型系统训练营 2024冬

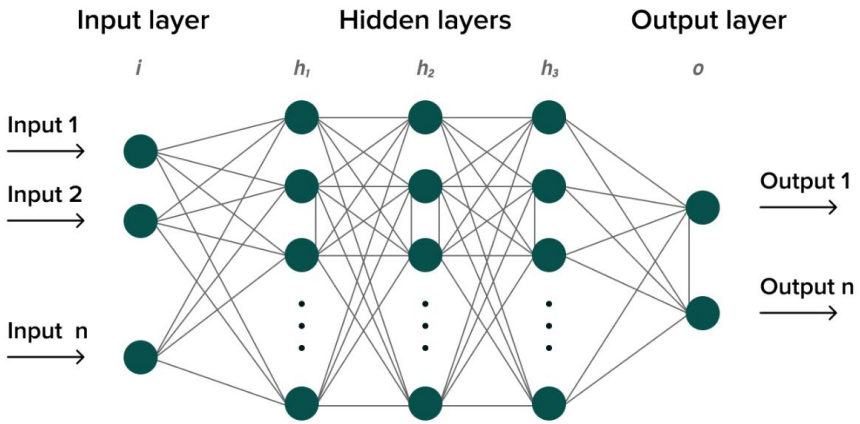
大模型推理系统02：大模型原理与结构

讲师　　潘泽众

前情回顾：人工神经网络

输入是一个向量，输出也是一个向量

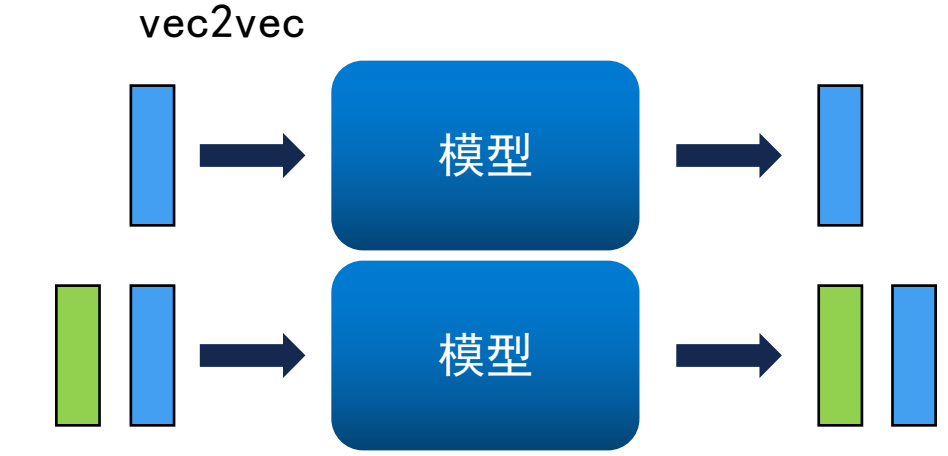
0. 02	0. 3	. . . . .				

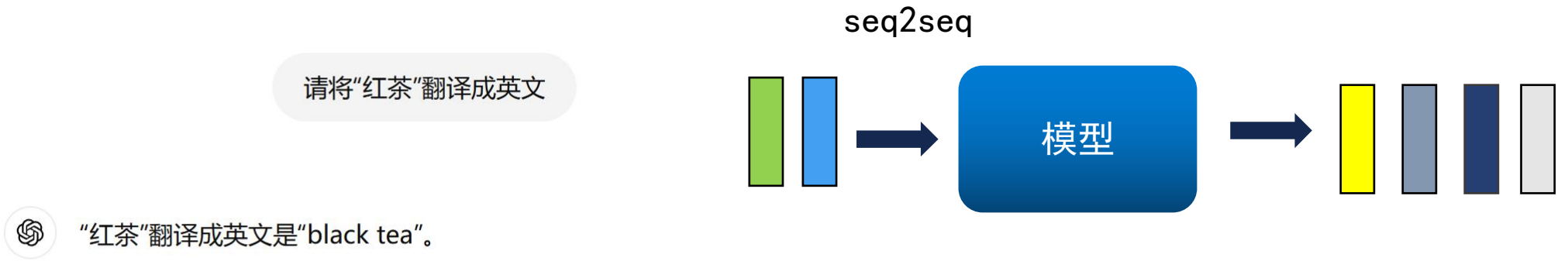


1	0	0	0	0	0	0	0	0	0

传统神经网络的局限

红 = Red
茶 = Tea
红茶 = Black Tea





- 输入是任意个数的向量，输出是任意（可与输入不同）个数的向量
- 向量之间推理是相互影响的

文本输入→张量输入

“中国的首都是北京”

“中” “国” “的” “首” “都” “是” “北” “京”

“中国” “的” “首都” “是” “北京”

- BPE（Byte Pair Encoding）：一种常用于文本处理的分词算法

1. 初始化：将文本中每个字符当作一个独立的token。
2. 统计频率：计算文本中每个字符对的出现频率。
3. 合并字符对：找到频率最高的字符对，并将其合并为一个新的token。
4. 更新词汇表：将新生成的token添加到词汇表中，并在文本中替换所有该字符对出现的地方。
5. 重复步骤2-4：反复执行上述步骤，直到达到预定的token数量或不能再找到新的字符对。

```
A B A B A B A C

'A B' -> 3
'B A' -> 3
'A C' -> 1
```

将新的token AB 添加到词汇表中。

文本输入→张量输入

标记器（Tokenizer）

Tokenizer

- Encode：根据词表，将文本转换为token-id序列
- Decode：将token-id还原成文本

```
import transformers
tokenzier = transformers.AutoTokenizer.from_pretrained(model_path)
token_ids = tokenzier.encode("Once upon a time")
tokens = tokenzier.convert_ids_to_tokens(token_ids)
print(token_ids)
print(tokens)
```

```
[1, 9038, 2501, 263, 931]
['<s>', '_Once', '_upon', '_a', '_time']
```

vocab_size: 32000

```
" efter"
" eg"
" egalitarian"
" eget"
" egg"
" eggs"
" ego"
" egreg"
" egregious"
" egy"
" eh"
" ehem"
" ehemal"
" ehemaligen"
" ei"
" eig"
" eigen"
" eigenen"
" eigenvalues"
" eigh"
" eight"
" eighteen"
```

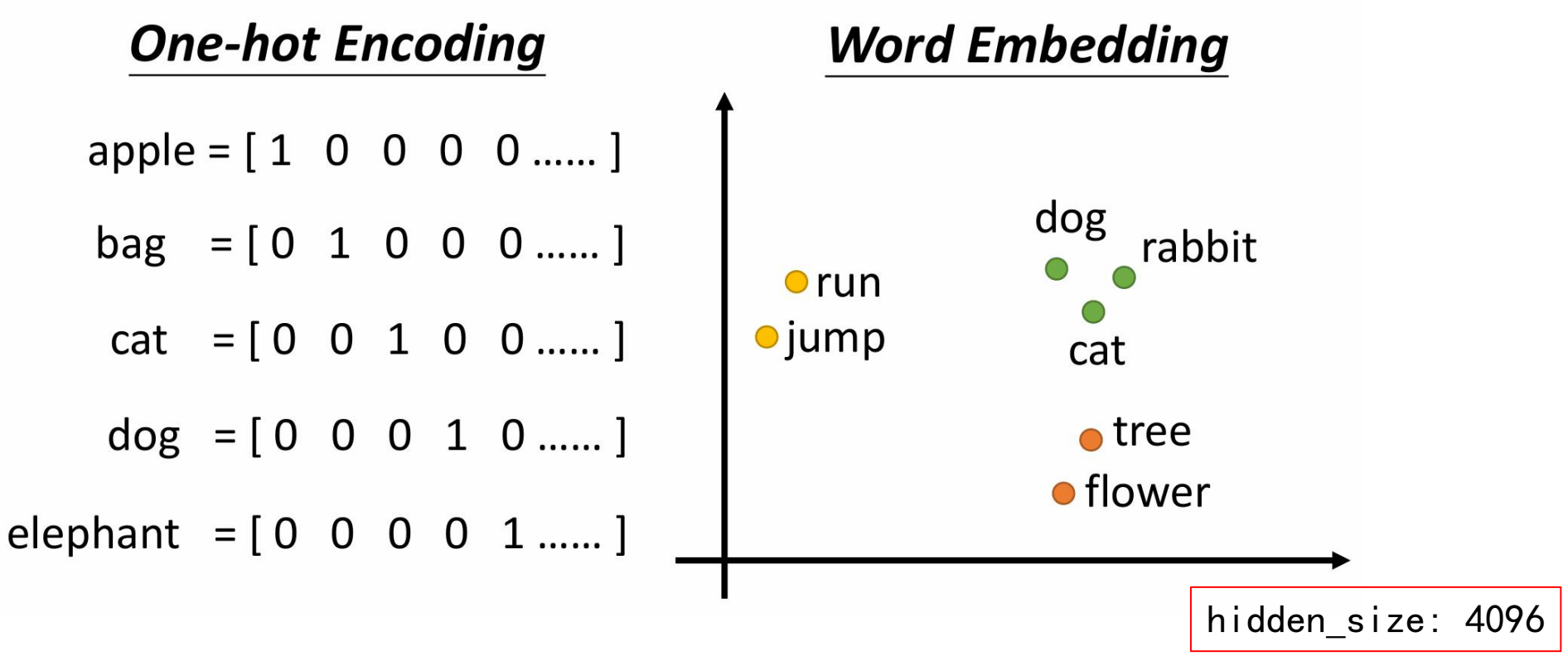
文本输入→张量输入

文本嵌入（Embedding）

将标记（tokens）转换为向量（张量）

Input Embedding

（计算方式为索引，而非矩阵乘）



基于Transformer的开源大语言模型——LLaMA

LLaMA（Large Language Model Meta AI）是由Meta开发的大型语言模型家族。可以用来执行文本分类、文本生成、翻译、AI对话等NLP任务。

Meta

- 基于Transformer架构
- 开源、可商用
- 效果好、结构简单
- 2024年LLaMA-3公布

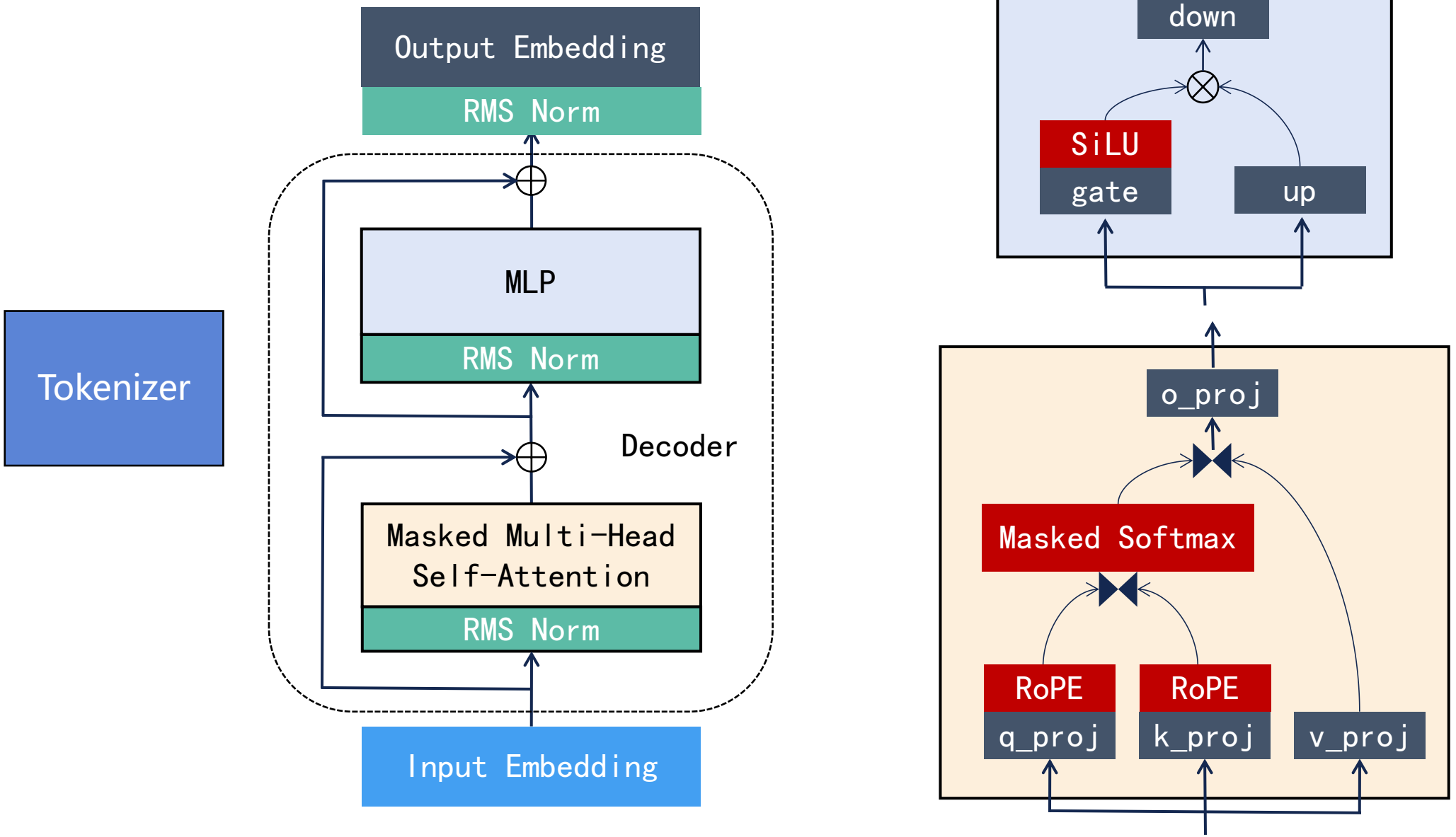

[Submitted on 12 Jun 2017 (v1), last revised 2 Aug 2023 (this version, v7)]

Attention Is All You Need

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin

官方代码：https://github.com/meta-llama/llama3
Transformers库：https://github.com/huggingface/transformers/tree/main/src/transformers/models/llama
官方模型下载（需要申请）：https://huggingface.co/meta-llama
TinyLlama：https://huggingface.co/TinyLlama/TinyLlama-1.1B-Chat-v0.4

大模型（LLaMA）模型结构



层归一化（RMS Norm）

LLaMA模型对每一层的输入都会做一次RMS Norm操作

- **每一层的输入输出最后一维都是hidden_size**

$$\bar{a}_i = \frac{a_i}{RMS(a)} g_i$$

$$RMS(a) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} {a_i}^2}$$

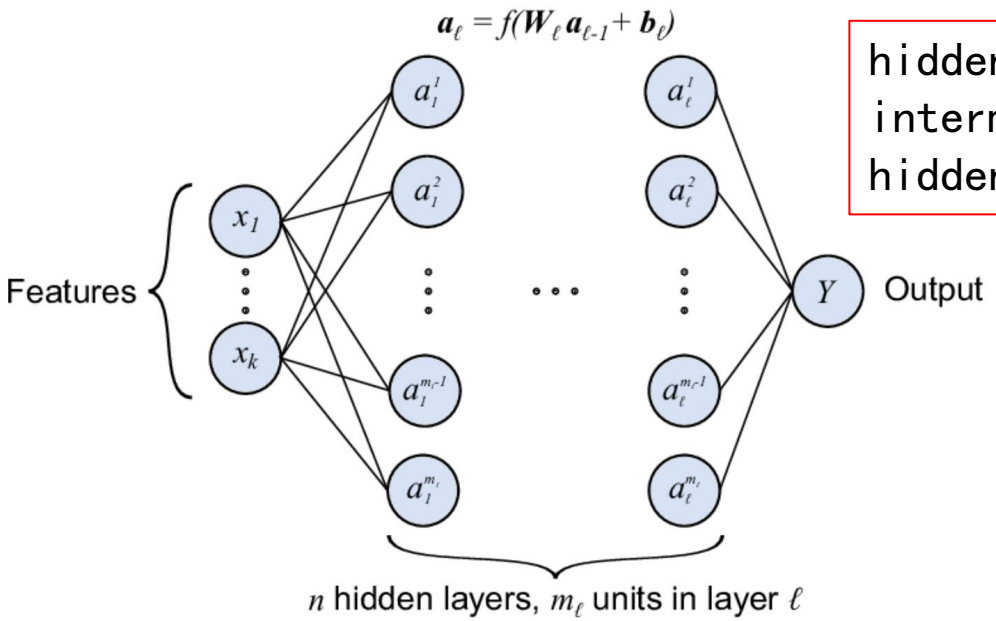
（g为可训练的模型参数）

```
hidden_size: 4096
rms_norm_eps: 1e-05
```

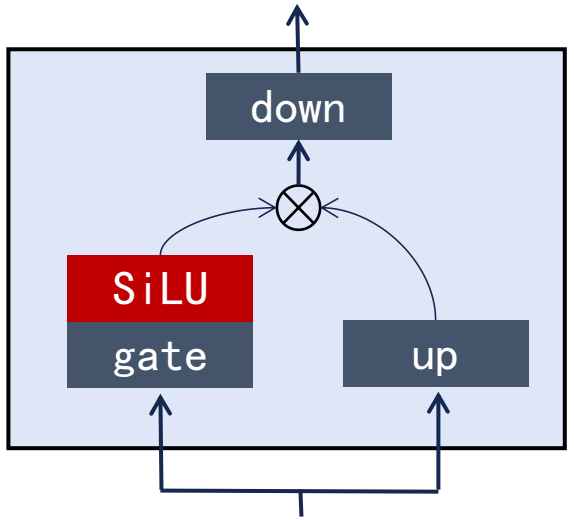
实践中会给分母加上一个小数eps避免分母为0

层归一化可以提升训练的效果，提高模型计算的稳定性

多层感知器（MLP）层

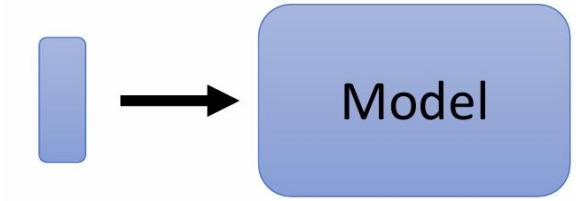


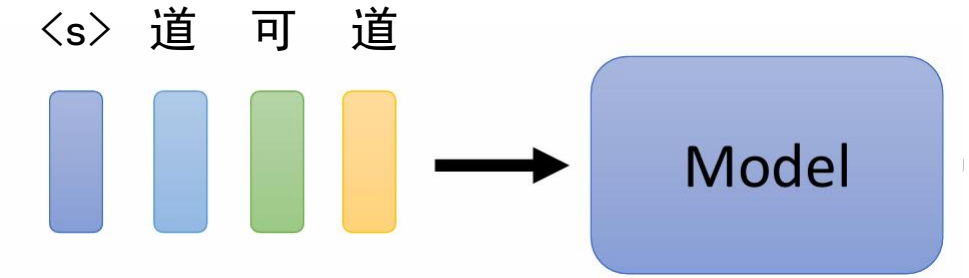

```
hidden_size: 4096
intermediate_size: 11008
hidden_act: silu
```



$$Y = (SiLU(X \cdot W_{gate}^T) \times X \cdot W_{up}^T) \cdot W_{down}^T$$

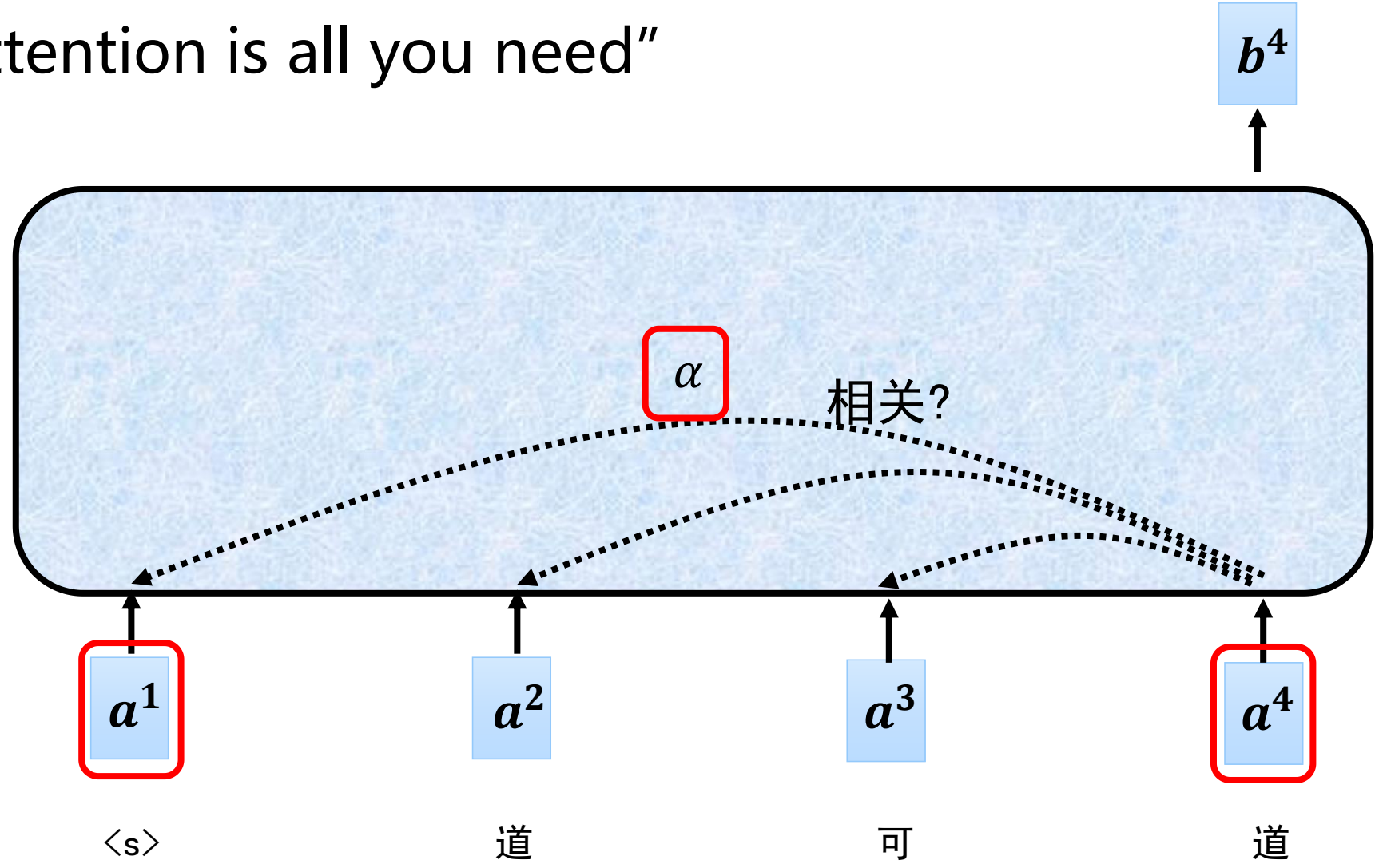
MLP层只接受固定特征（feature）数的输入



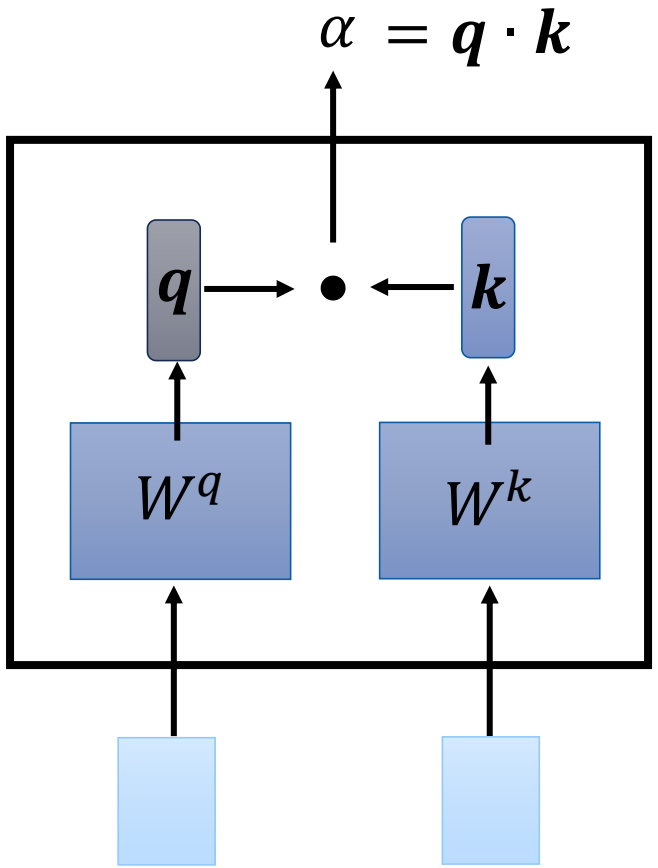


自注意力（Self-Attention）层

“Attention is all you need”

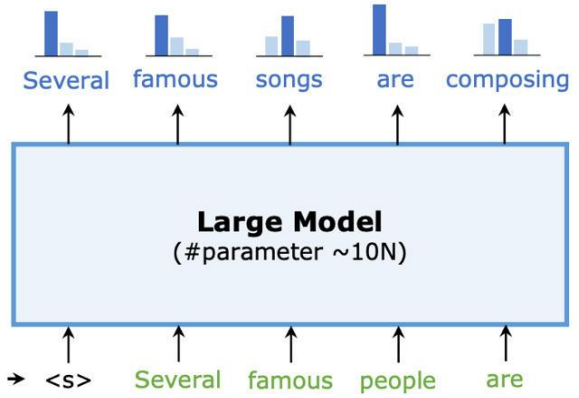


自注意力（Self-Attention）层

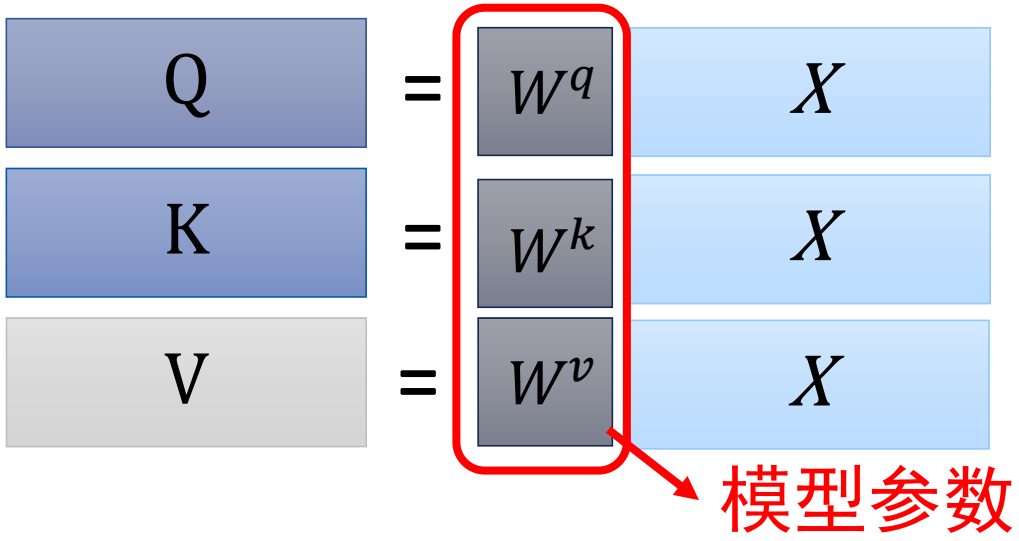


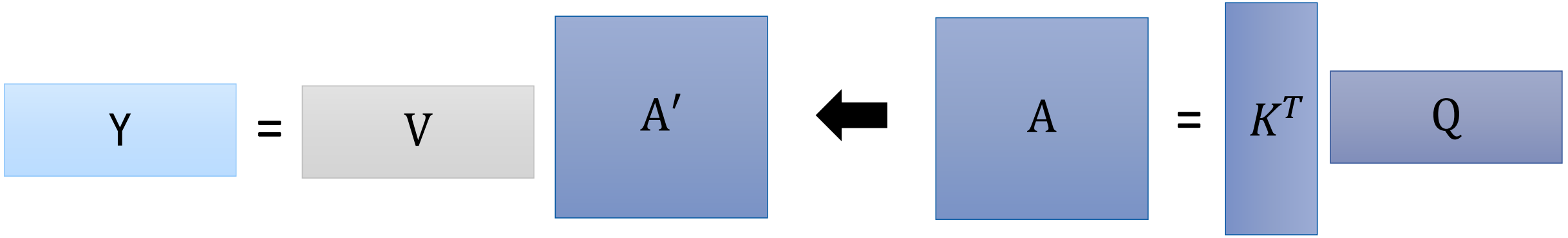
对Attention矩阵进行Masked Softmax操作

q\k	<s>	道	可	道
<s>	1.0	0	0	0
道	0.1	0.9	0	0
可	0.1	0.6	0.3	0
道	0.0	0.2	0.5	0.3

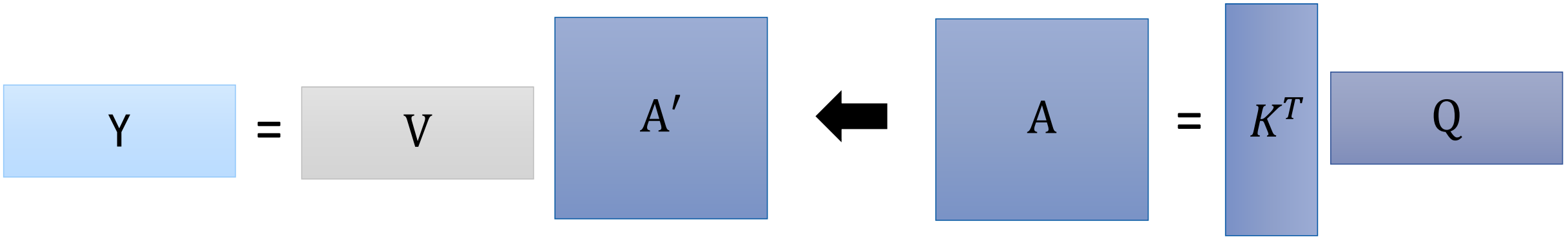


自注意力（Self-Attention）层





自注意力（Self-Attention）层

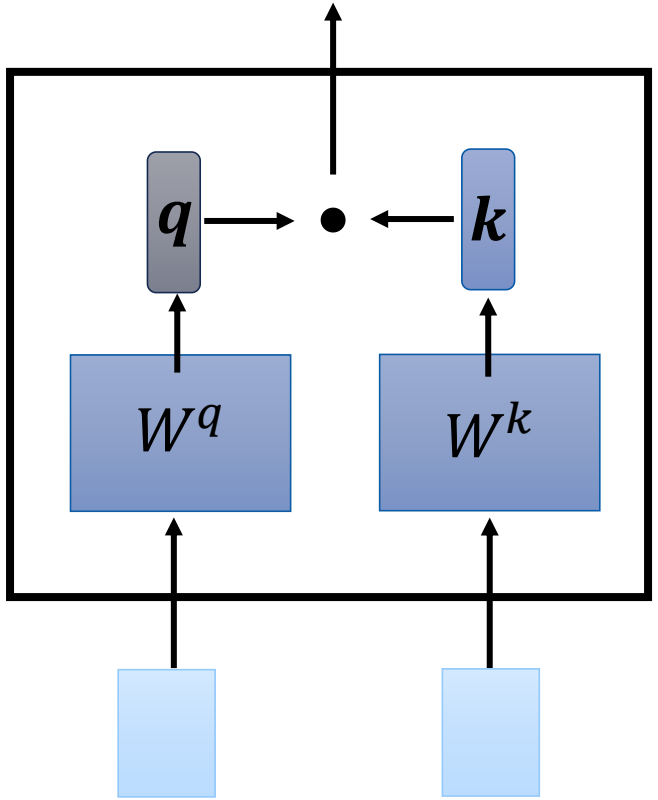


“一行行，行行行”

需要位置信息

旋转位置编码（RoPE）

$$\alpha(t,s) = Rope(\boldsymbol{q},\boldsymbol{t}) \cdot Rope(\boldsymbol{k},\boldsymbol{s})$$

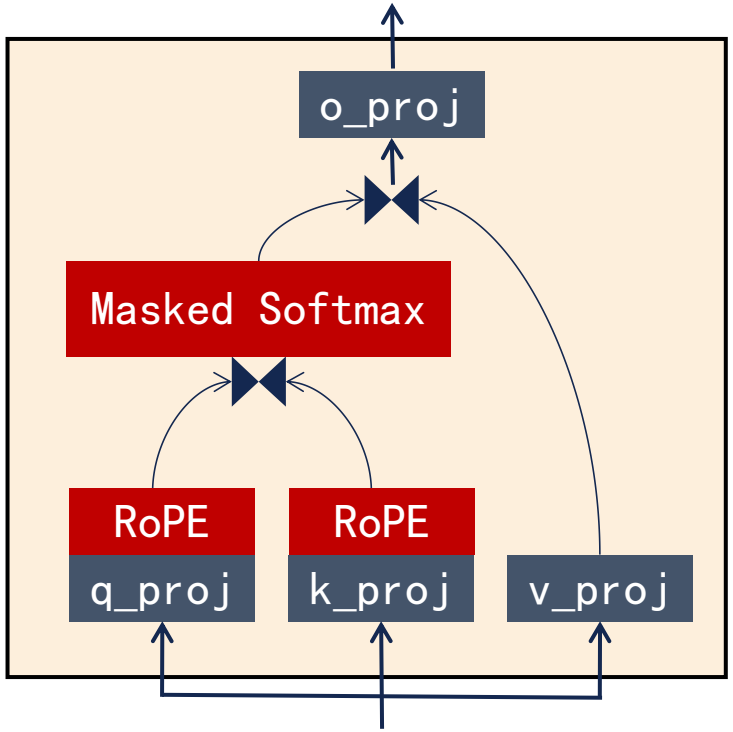


$$\begin{bmatrix} q_t^{(1)} \\ q_t^{(2)} \\ \vdots \\ \vdots \\ q_t^{(d-1)} \\ q_t^{(d)} \end{bmatrix} \leftarrow \begin{bmatrix} \cos t\theta_1 \\ \cos t\theta_1 \\ \vdots \\ \vdots \\ \cos t\theta_{d/2} \\ \cos t\theta_{d/2} \end{bmatrix} \odot \begin{bmatrix} q_t^{(1)} \\ q_t^{(2)} \\ \vdots \\ \vdots \\ q_t^{(d-1)} \\ q_t^{(d)} \end{bmatrix} + \begin{bmatrix} \sin t\theta_1 \\ \sin t\theta_1 \\ \vdots \\ \vdots \\ \sin t\theta_{d/2} \\ \sin t\theta_{d/2} \end{bmatrix} \odot \begin{bmatrix} -q_t^{(2)} \\ q_t^{(1)} \\ \vdots \\ \vdots \\ -q_t^{(d)} \\ q_t^{(d-1)} \end{bmatrix}$$

$$\theta_n = 10000^{-2n/d}$$ d为k（或q）的向量长度

两种不同的RoPE（GPT-J和GPT-NeoX）： https://zhuanlan.zhihu.com/p/631363482

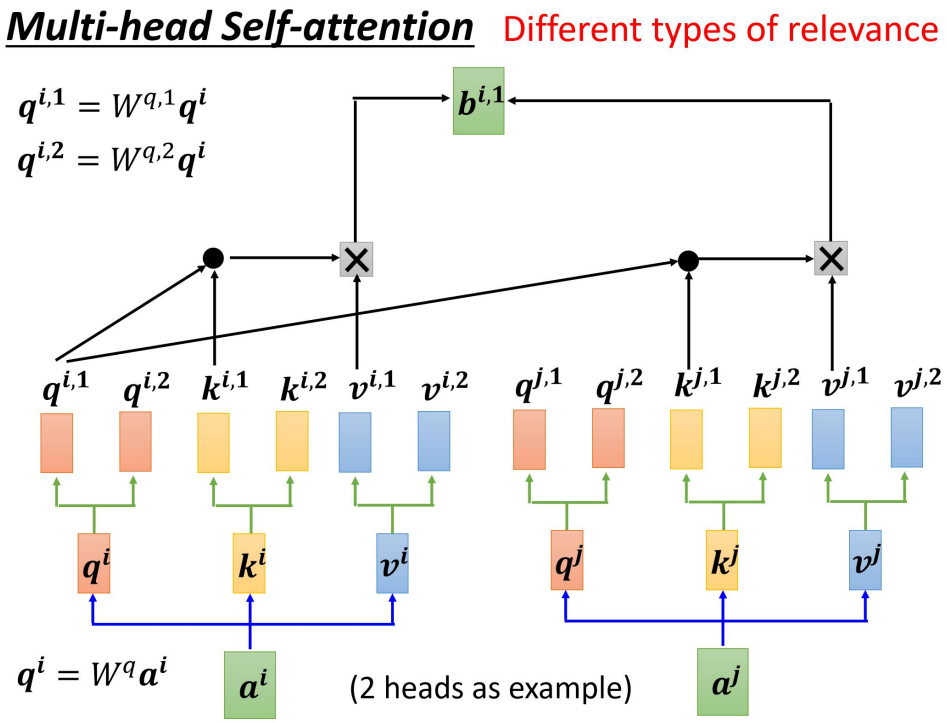
自注意力（Self-Attention）层



- 在Decoder中，每个token只对它前面的token求注意力，使用masked softmax
- Encoder使用普通的softmax
- Llama是Decoder-only模型

```
hidden_size: 4096
num_attention_heads: 32
num_key_value_heads: 32
```

多头注意力



Q/K的维度

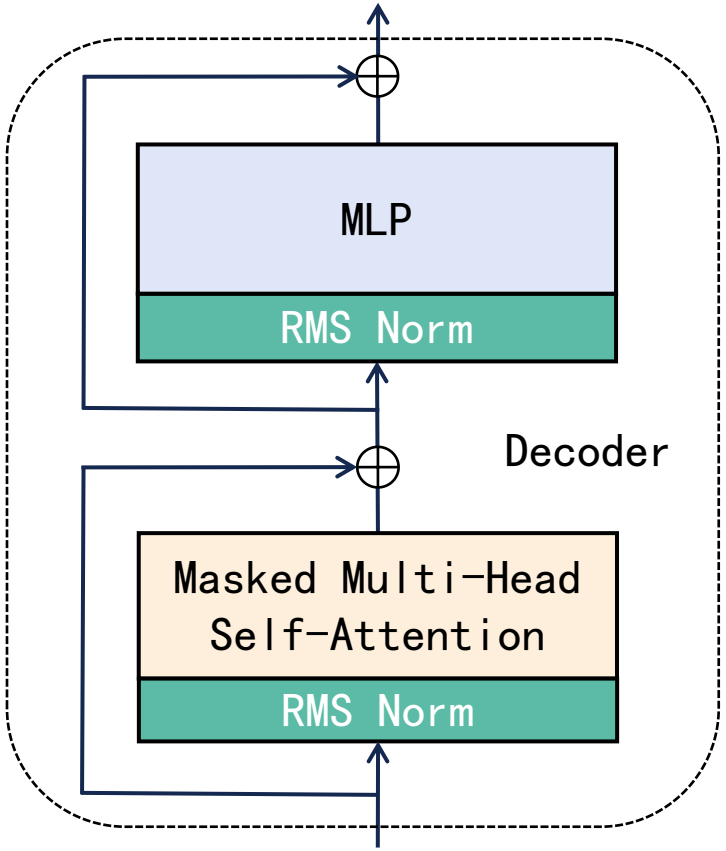
（L， 4096）-> （L， 32， 128）

```
hidden_size: 4096
num_attention_heads: 32
num_key_value_heads: 32
```

Q头数可以是KV头数的倍数

Q：（L， 32， 128）->（L， 8， 4， 128）
KV：（L， 8， 128）->（L， 8， 1， 128）

解码器（Decoder）模块



- Add（加）
- Mul（乘）
- Sqrt（平方根）
- Transpose（转置）
- Reshape
- Matmul（矩阵乘）
- Softmax
- Sigmoid
- Sin/Cos
- ReduceMean（平均值）

num_hidden_layers: 32

输出嵌入层

Output Embedding

（计算方式为矩阵乘）

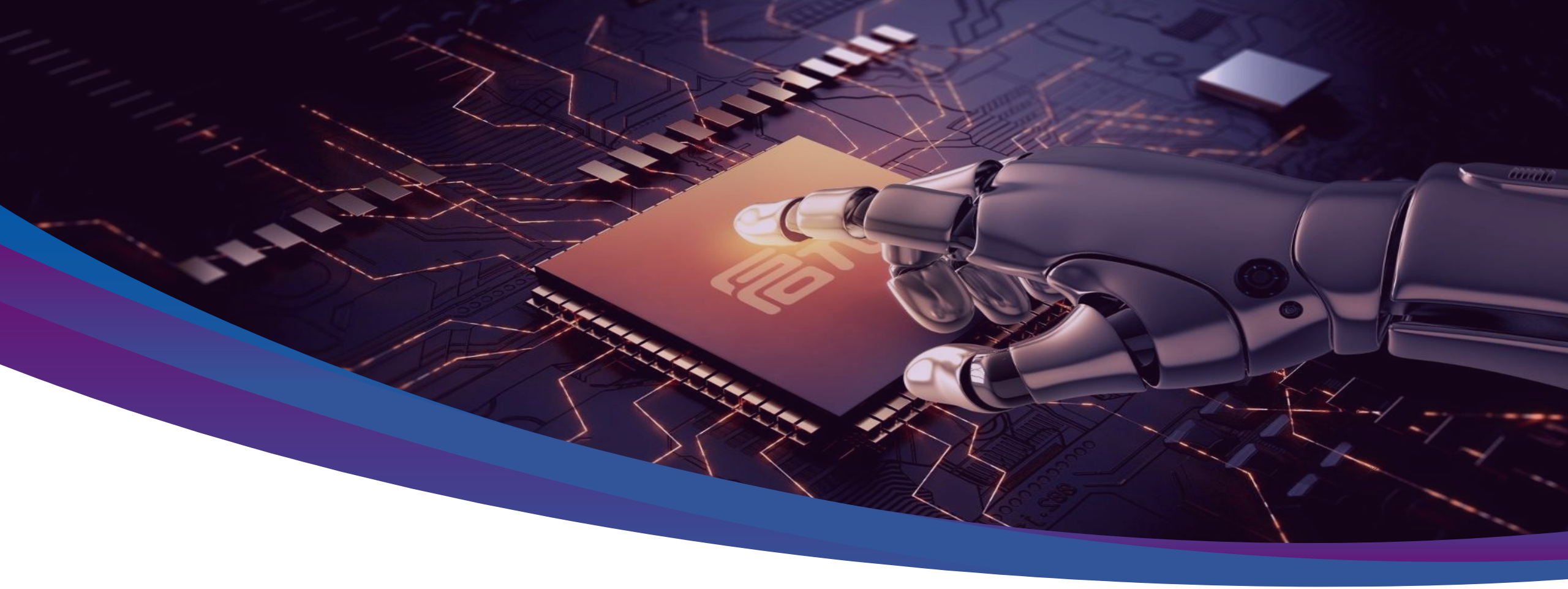
- 有些模型的输入输出Embedding会使用同一份参数
  - 一定程度上避免overfit
  - 降低模型大小
  - 并不常用

```
tie_word_embeddings: false
```

- 可以只对需要输出的logits进行这步计算

LLaMA模型的配置


```
{
  "_name_or_path": "meta-llama/Llama-2-7b-hf",
  "architectures": [
    "LlamaForCausalLM"
  ],
  "bos_token_id": 1,
  "eos_token_id": 2,
  "hidden_act": "silu",
  "hidden_size": 4096,
  "initializer_range": 0.02,
  "intermediate_size": 11008,
  "max_position_embeddings": 4096,
  "model_type": "llama",
  "num_attention_heads": 32,
  "num_hidden_layers": 32,
  "num_key_value_heads": 32,
  "pretraining_tp": 1,
  "rms_norm_eps": 1e-05,
  "rope_scaling": null,
  "tie_word_embeddings": false,
  "torch_dtype": "float16",
  "transformers_version": "4.31.0.dev0",
  "use_cache": true,
  "vocab_size": 32000
}
```



感谢聆听

GitHub开源组织：https://github.com/InfiniTensor