Lab3. Transformer

TA: 王雨轩

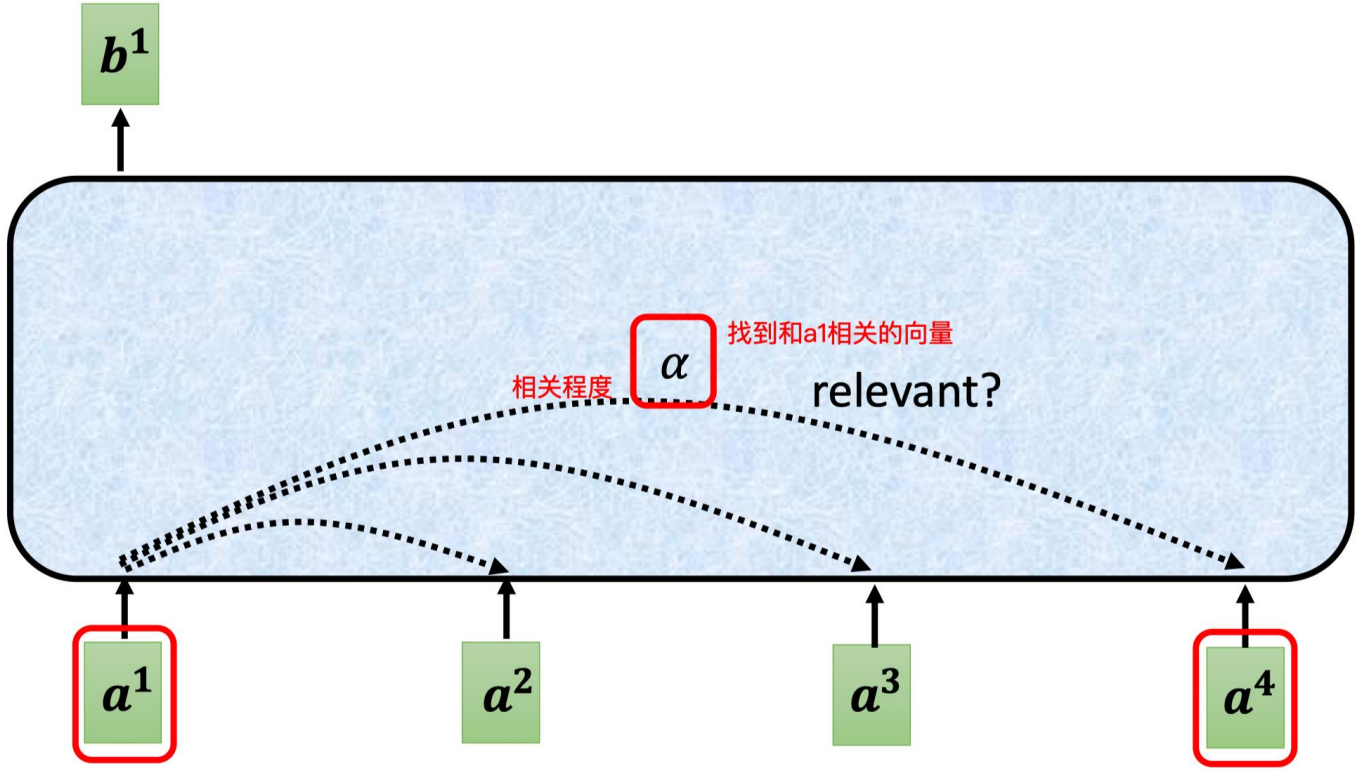
yuxwang22@m.fudan.edu.cn

Outline

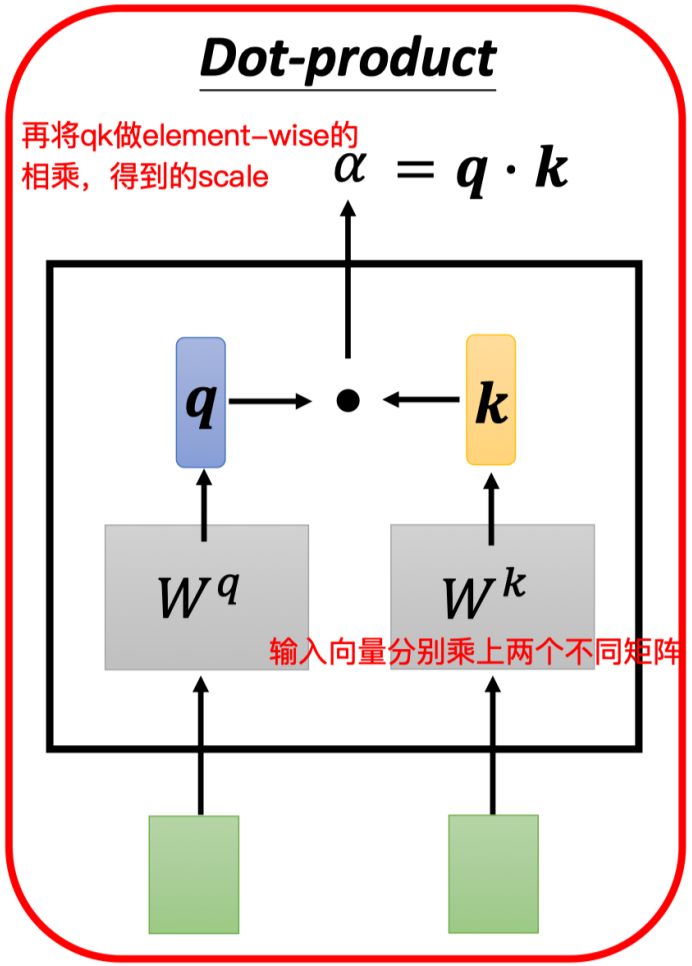
- **相关知识复习**
- 简答题
- 代码
  - 任务简介
  - 作业要求
  - 提交方式

Attention计算

Self-attention

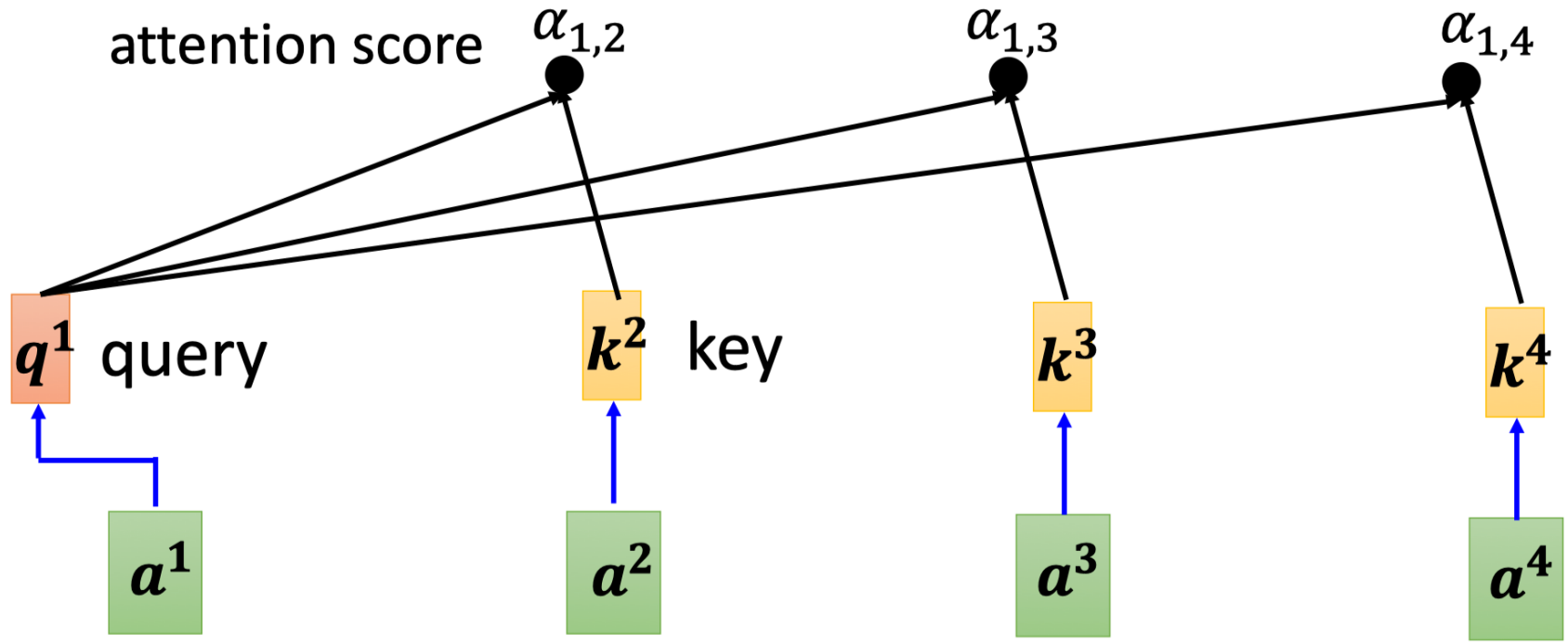


Find the relevant vectors in a sequence



Attention计算

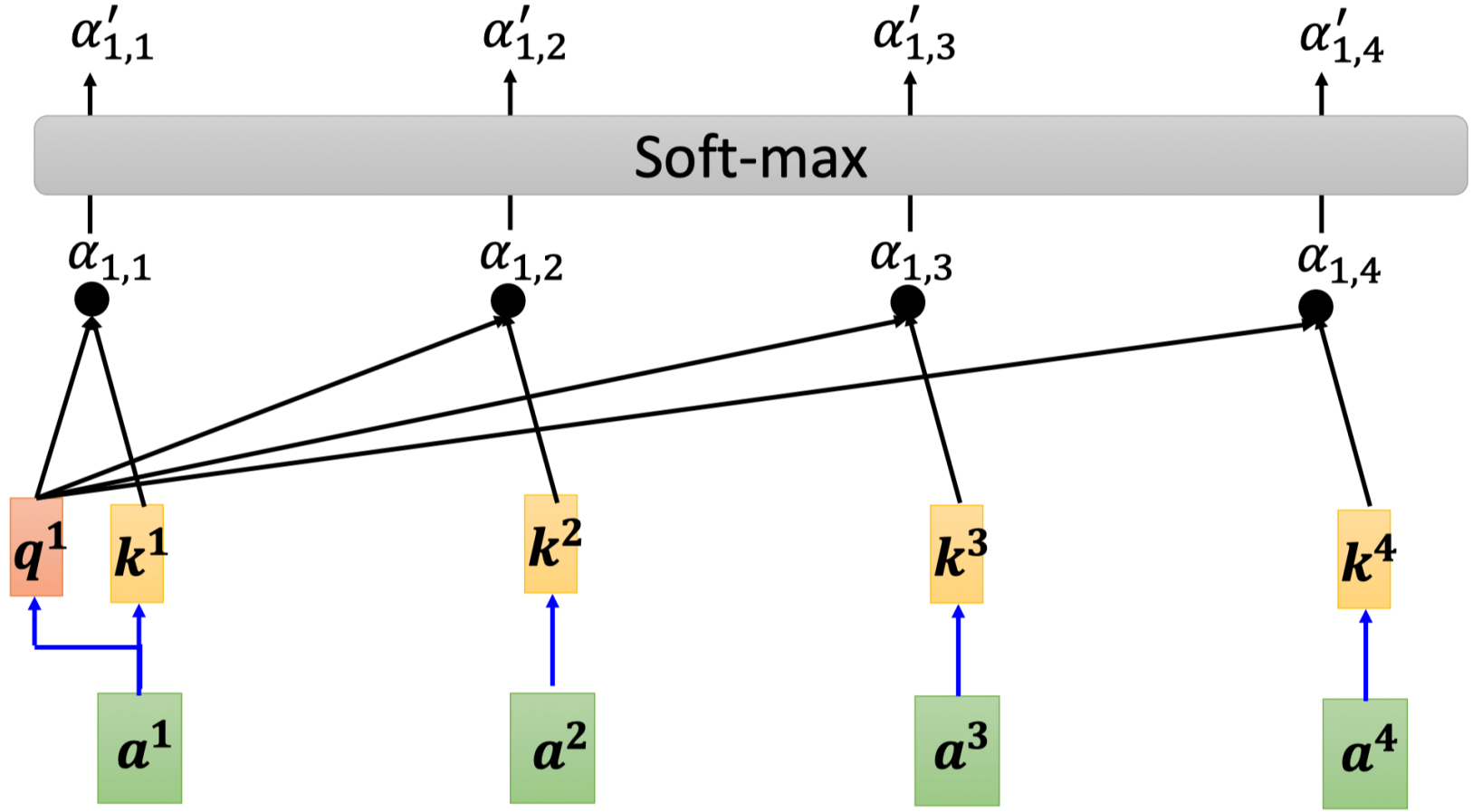
$$\alpha_{1,2} = \boldsymbol{q^1} \cdot \boldsymbol{k^2} \qquad \alpha_{1,3} = \boldsymbol{q^1} \cdot \boldsymbol{k^3} \qquad \alpha_{1,4} = \boldsymbol{q^1} \cdot \boldsymbol{k^4}$$



$$\boldsymbol{q^1} = W^q \boldsymbol{a^1} \qquad \boldsymbol{k^2} = W^k \boldsymbol{a^2} \qquad \boldsymbol{k^3} = W^k \boldsymbol{a^3} \qquad \boldsymbol{k^4} = W^k \boldsymbol{a^4}$$

Self-attention

$$\alpha'_{1,i} = exp(\alpha_{1,i}) / \sum_j exp(\alpha_{1,j})$$



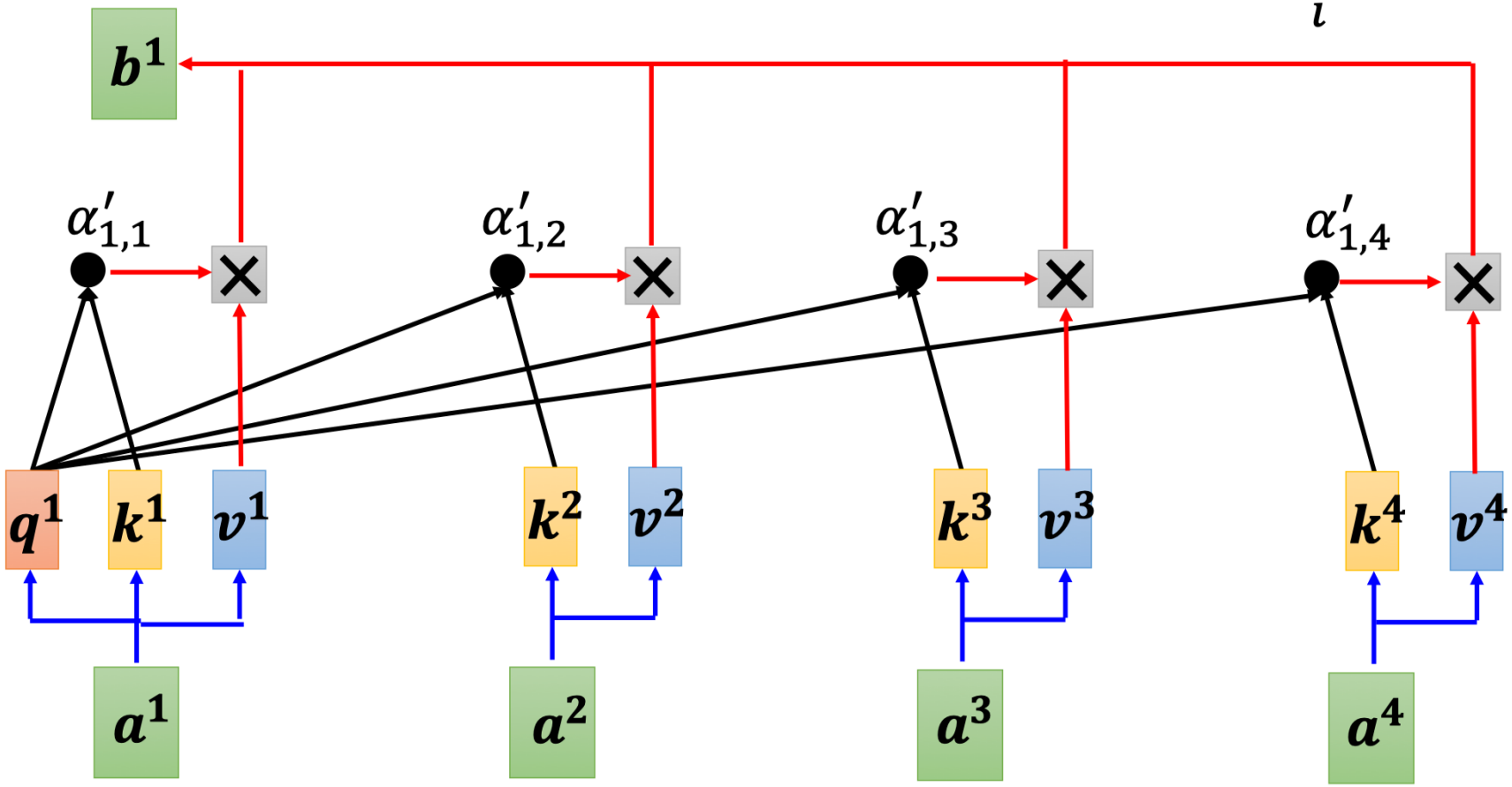
$q^1 = W^q a^1$ $k^2 = W^k a^2$ $k^3 = W^k a^3$ $k^4 = W^k a^4$

$k^1 = W^k a^1$

Self-attention

Extract information based on attention scores

$$b^1 = \sum_i \alpha'_{1,i} v^i$$



$v^1 = W^v a^1$ $v^2 = W^v a^2$ $v^3 = W^v a^3$ $v^4 = W^v a^4$

Self-attention

$$q^i = W^q a^i \qquad q^1 q^2 q^3 q^4 = W^q \; a^1 a^2 a^3 a^4$$

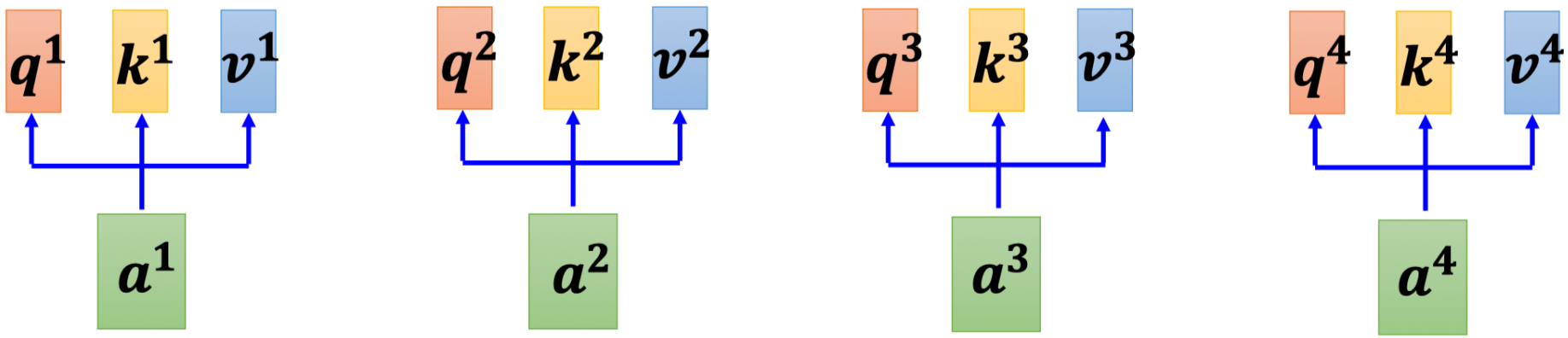
Q $\quad$ I

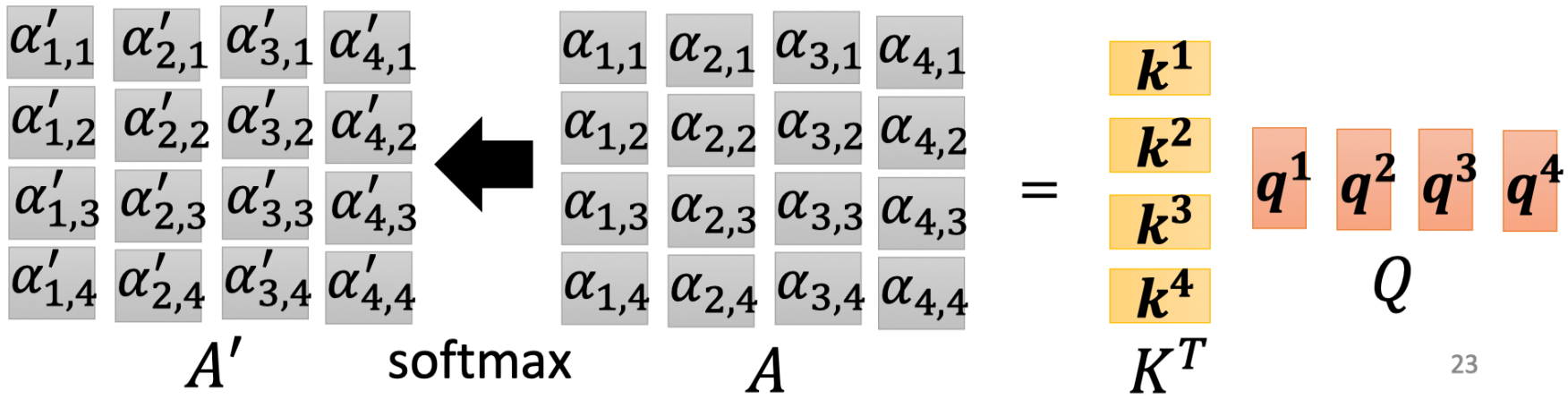
$$k^i = W^k a^i \qquad k^1 k^2 k^3 k^4 = W^k \; a^1 a^2 a^3 a^4$$

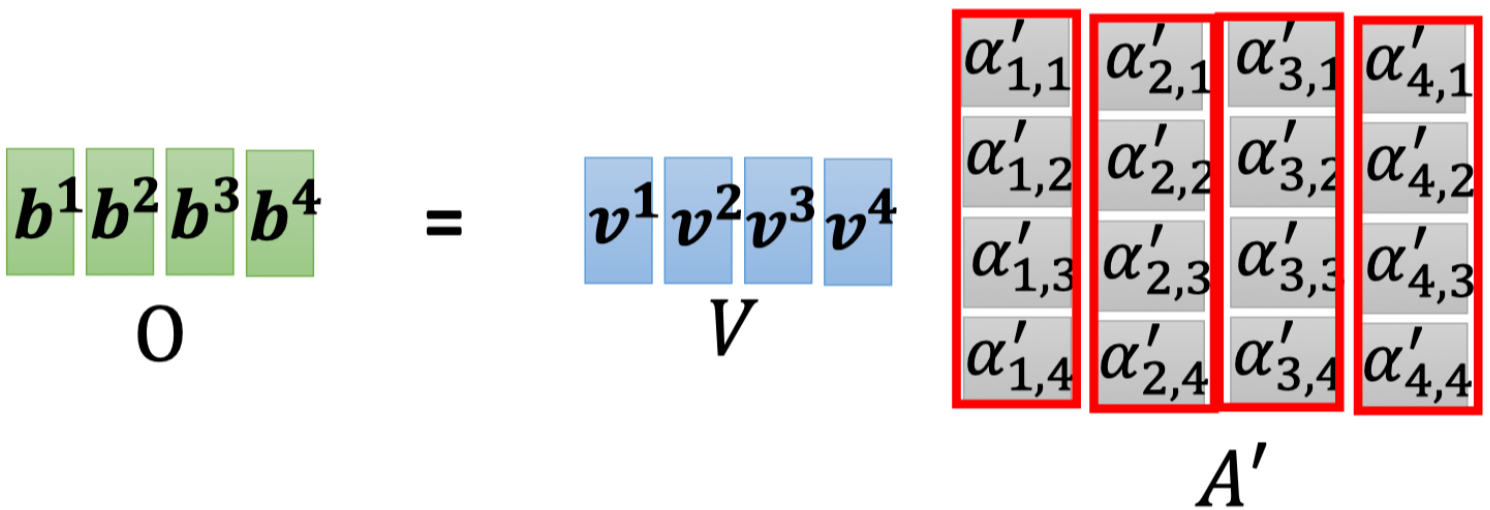
K $\quad$ I

$$v^i = W^v a^i \qquad v^1 v^2 v^3 v^4 = W^v \; a^1 a^2 a^3 a^4$$

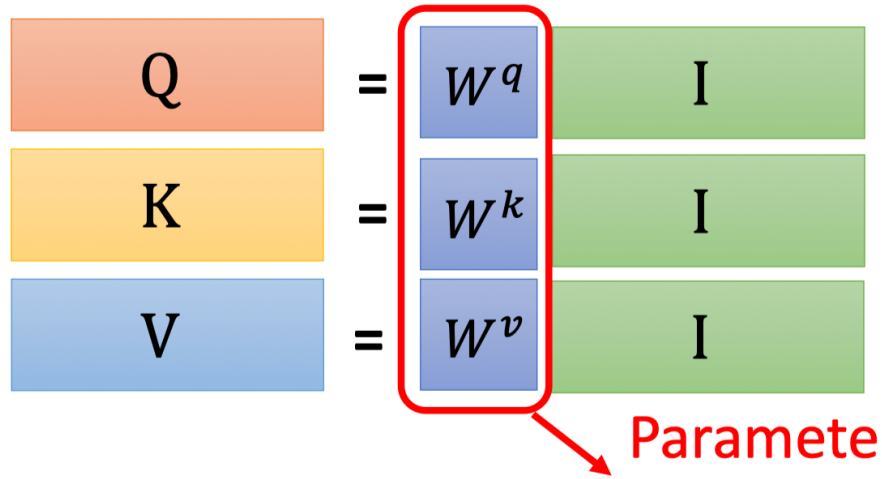
V $\quad$ I

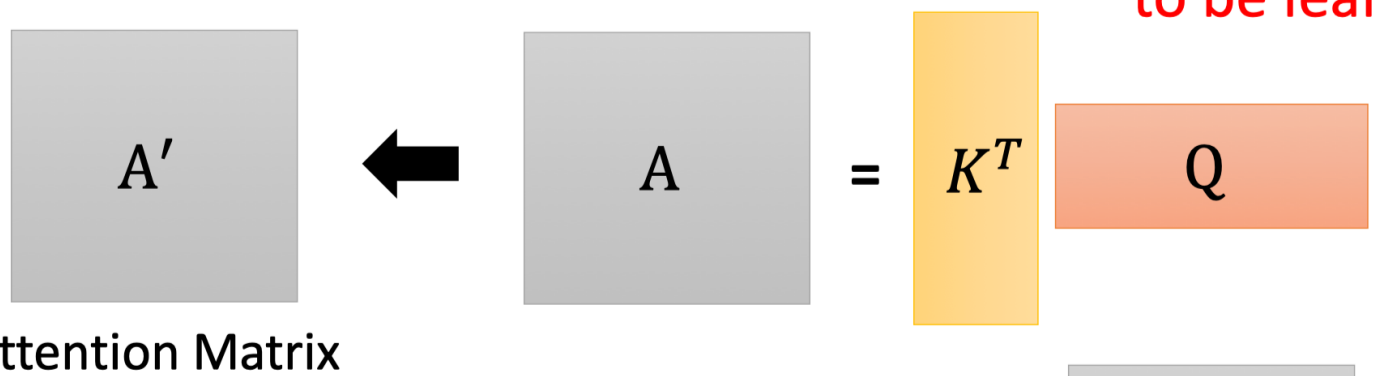



$\alpha'_{1,1}$ $\alpha'_{2,1}$ $\alpha'_{3,1}$ $\alpha'_{4,1}$
$\alpha'_{1,2}$ $\alpha'_{2,2}$ $\alpha'_{3,2}$ $\alpha'_{4,2}$
$\alpha'_{1,3}$ $\alpha'_{2,3}$ $\alpha'_{3,3}$ $\alpha'_{4,3}$
$\alpha'_{1,4}$ $\alpha'_{2,4}$ $\alpha'_{3,4}$ $\alpha'_{4,4}$
A'
softmax
$\alpha_{1,1}$ $\alpha_{2,1}$ $\alpha_{3,1}$ $\alpha_{4,1}$
$\alpha_{1,2}$ $\alpha_{2,2}$ $\alpha_{3,2}$ $\alpha_{4,2}$
$\alpha_{1,3}$ $\alpha_{2,3}$ $\alpha_{3,3}$ $\alpha_{4,3}$
$\alpha_{1,4}$ $\alpha_{2,4}$ $\alpha_{3,4}$ $\alpha_{4,4}$
A
=
k^1 k^2 k^3 k^4
K^T
q^1 q^2 q^3 q^4
Q


b^1 b^2 b^3 b^4
O
=
v^1 v^2 v^3 v^4
V
$\alpha'_{1,1}$ $\alpha'_{2,1}$ $\alpha'_{3,1}$ $\alpha'_{4,1}$
$\alpha'_{1,2}$ $\alpha'_{2,2}$ $\alpha'_{3,2}$ $\alpha'_{4,2}$
$\alpha'_{1,3}$ $\alpha'_{2,3}$ $\alpha'_{3,3}$ $\alpha'_{4,3}$
$\alpha'_{1,4}$ $\alpha'_{2,4}$ $\alpha'_{3,4}$ $\alpha'_{4,4}$
A'

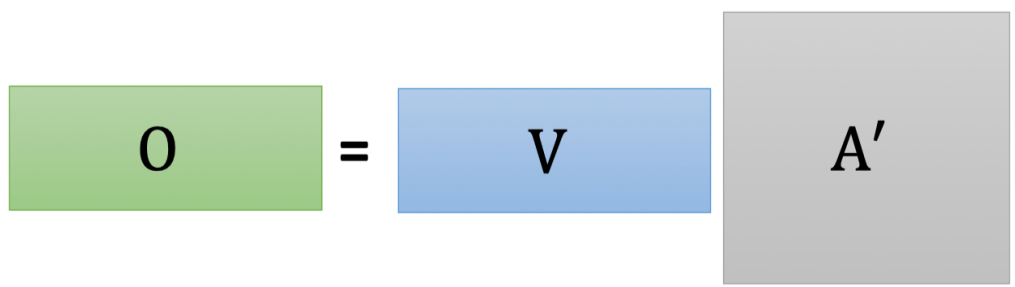
Self-attention





Attention Matrix

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



Outline

- 相关知识复习
- **简答题（30分）**
- 代码
  - 任务简介
  - 作业要求
  - 提交方式

简答题

- 第一题：注意力机制（10分）
- 1. 为什么要使用“多头”注意力机制？ （3分）
- 2. attention 计算的时间复杂度是多少？ （2分）
- 3.注意力机制计算中为什么要除以$\sqrt{d_k}$？(开放题， 5分)

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

简答题

- 第二题：Transformer（20分）
- 1. Transformer中使用残差连接的目的是什么？（2分）
- 2. Transformer的输入和输出的维度是一样的吗？（2分）
- 3. Transformer中使用位置编码的目的是什么？（3分）
- 4. Transformer为什么Q和K使用不同的权重矩阵生成，为何不能使用同一个值进行自身的点乘？（3分）
- 5. Transformer和CNN, RNN 相比，它们各自的优劣是什么？（5分）
- 6.为什么GPT中要使用causal attention（5分）

Outline

- 相关知识复习
- 简答题
- **代码（70分）**
  - 任务简介
  - 作业要求
  - 提交方式

作业链接

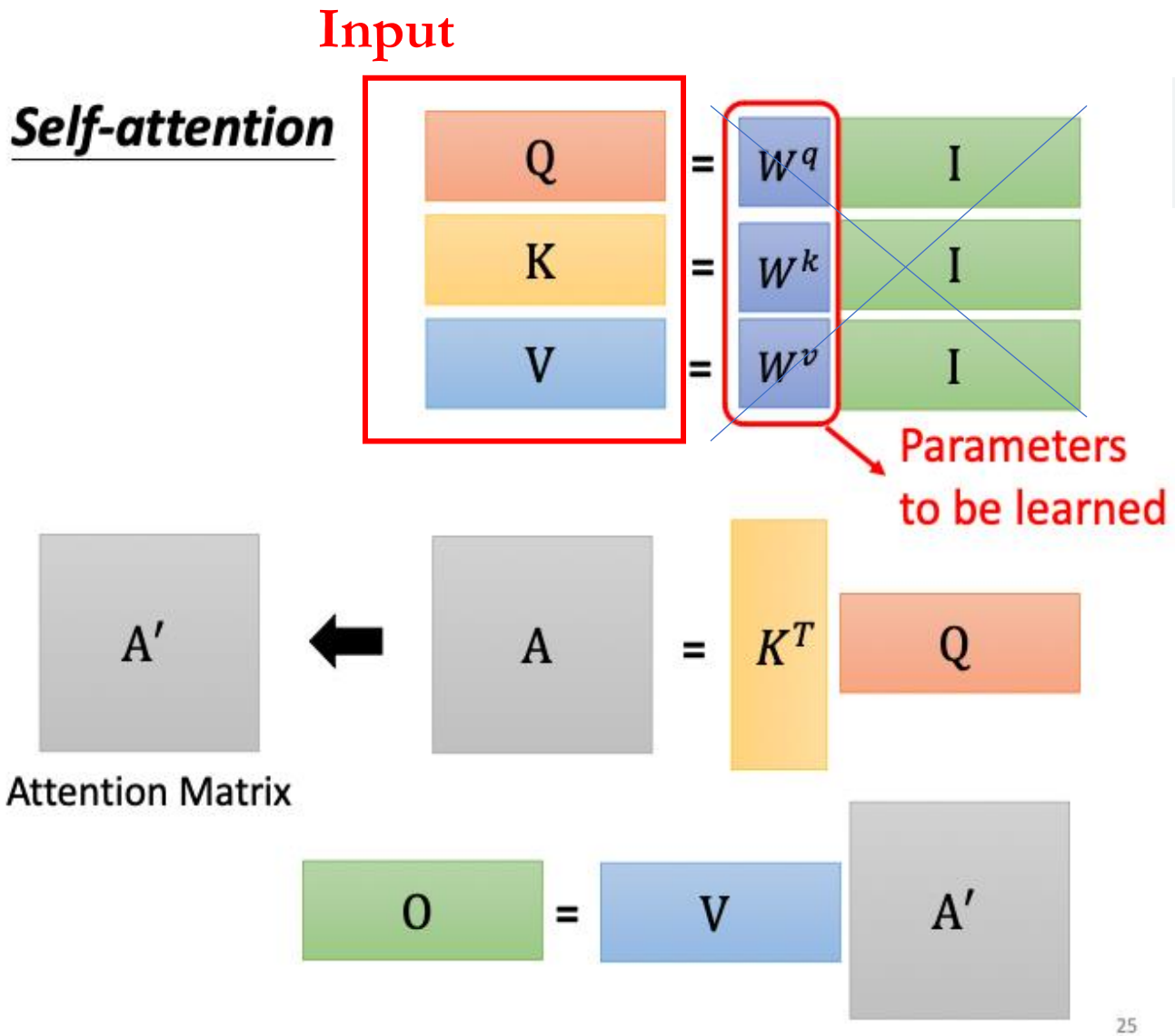
- 作业链接
- Kaggle 的使用方式可参考 Lab1 的 ppt
- Kaggle 每周有固定的 gpu 使用时间，同学们每次使用完之后要记得关闭环境，以免造成时间浪费
- Kaggle 后台运行方法：
  https://www.yuque.com/wjpoom/fudan-ai/qe1zxrx6l4sgltyg?singleDoc# 《Kaggle使用手册02：后台运行》
  密码：arg3

任务简介

- 学习如何逐步实现self-attention算法和基于编码器-解码器的Transformer结构
- 在AddSub数据集上完成一个定长的向量到向量的任务。

任务1: 补全注意力算法的代码

- 补全 **scaled_dot_product_no_loop_batch**（10分）
  - 请不要使用循环语句
  - 使用torch.bmm计算
  - 完成后在测试代码1中测试，error应小于1e-5




```python
def scaled_dot_product_no_loop_batch(
    query: Tensor, key: Tensor, value: Tensor, mask: Tensor = None
) -> Tensor:
```

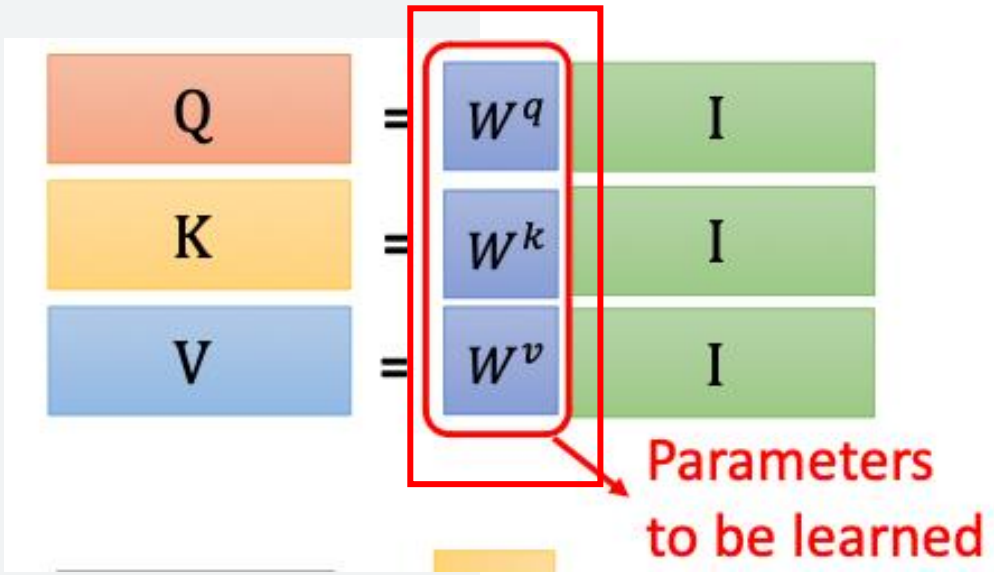
- input： Query， Key， Value
- 都为（N, K, M）的张量，其中 N 是批次大小，K 是序列长度，M 是序列嵌入维度
- 在这一部分，我们假设我们已经得到了查询、键和值向量(即我们需要处理的输入是如图所示的Q，K，V)

任务1: 补全注意力算法的代码

- 补全**SelfAttention**类 （5分）
  - 封装了自注意力层的实现
  - Init部分不需要改动，只需要补全forward部分
  - 完成后在测试代码2中测试，error应小于1e-5

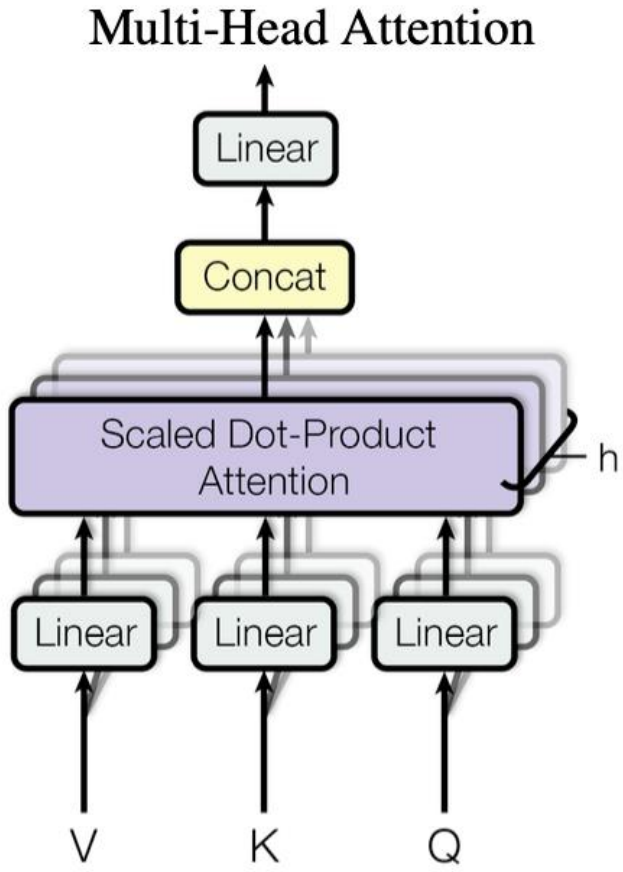
self.q, self.k, self.v

```python
class SelfAttention(nn.Module):
    def __init__(self, dim_in: int, dim_q: int, dim_v: int):
        super().__init__()

        """
        该类封装了自注意力层的实现。我们使用 MLP 层映射输入的查询、键和值，
        然后使用 scaled_dot_product_no_loop_batch 得到最终输出。

        参数:
            dim_in: 输入序列嵌入维度的整数值
            dim_q: 查询和键向量输出维度的整数值
            dim_v: 值向量的输出维度的整数值

        """
        ##########################################################################
        # 初始化三个 nn.Linear 层，可以将输入维度为 dim_in 的输入转换为维度为 dim_q 的查询，  #
        # 维度为 dim_q 的键，和维度为 dim_v 的值。                                  #
        ##########################################################################
        # 初始化q， k， v，  由于qk要进行点积计算，其维度需要保持一致
        self.q = nn.Linear(dim_in, dim_q)
        self.k = nn.Linear(dim_in, dim_q)
        self.v = nn.Linear(dim_in, dim_v)
        self.weights_softmax = None
```



任务2：补全Multi-head Attention

- 补全**MultiHeadAttention**的init和forward部分（10分）
- 这里完成的是一个简单的多头注意力层
  - 并行使用多个**SelfAttention**层，并将它们在最后一个维度上拼接起来，再通过一个线性层输出。
- 完成后在测试代码3中测试，error应小于1e-5



任务2：补全Multi-head Attention

如何并行多个SelfAttention层？

使用 nn.ModuleList 初始化一系列 SingleHeadAttention 层模块，

这个列表的长度应该等于注意力头的个数

```
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
self.heads = # TODO: self.heads = nn.ModuleList('add your code here')
self.linear = # TODO: nn.Linear
```

任务3.补全Layer Normalization（10分）

- 归一化是指将数据按**比例缩放**，使其落入一个特定的范围。Layer Normalization用于归一化每个隐藏层的激活值。
- 传统的batch normalization是针对每个批次的数据进行归一化，而layer normalization则是针对每个隐藏层的输出进行归一化。
- 这使得层归一化在处理较小的批次或者在序列数据上更为有效，因为它不依赖于批次维度。
- 因为自注意力对对输入数据的顺序敏感，所以常常使用层归一化

任务3.补全Layer Normalization

- 层归一化需要使输入具有零均值和单位方差。
  - 计算输入的均值和标准差，并使用它们对输入进行归一化。
  - 使用 self.gamma 和 self.beta 来缩放和偏移这个归一化的输入。
- TODO：计算均值，方差和标准差
- 完成后在测试代码4中测试，error应小于1e-5

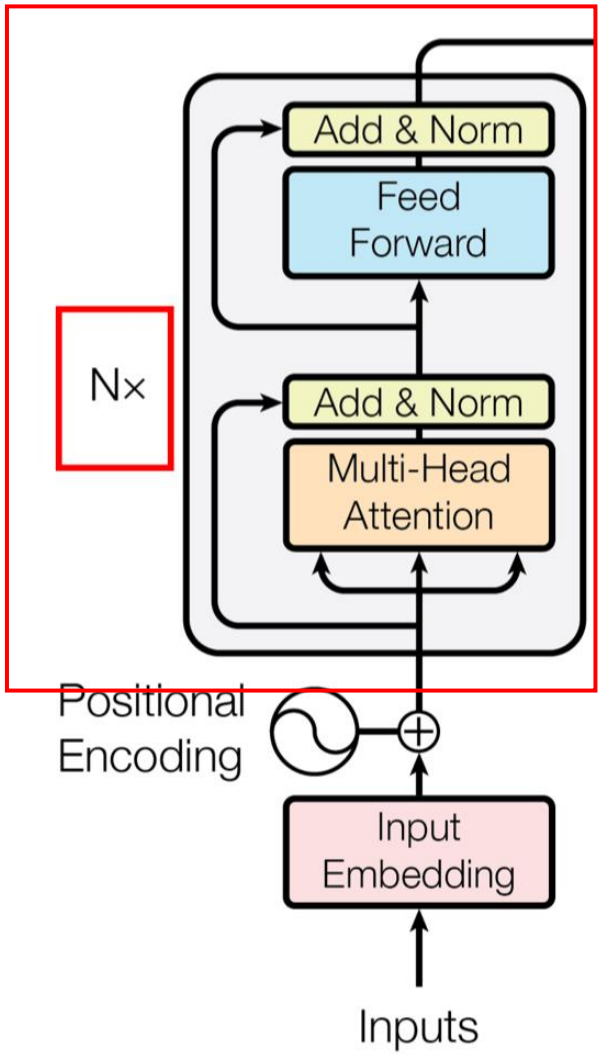
```
mean = # TODO:计算输入的均值，在最后1个维度上计算的， 可参考torch.mean，注意keepdim的用法
var = # TODO: 计算方差，在最后一个维度上计算
std = # TODO: 计算标准差
y = (x - mean) / std # 进行归一化
return self.gamma * y + self.beta # 使用self.gamma和self.beta来缩放和偏移归一化的输入
```

任务3.补全Layer Normalization

- 在最后一个维度上计算
  - 输入x:的形状为 (N, K, M)，其中 N 是批次大小，K 是序列长度，M 是嵌入维度，我们希望对每个样本（N）的每个序列（K）的**每个特征（M）**进行归一化。因此，我们在最后一个维度（即特征维度）上计算均值和标准差。
- Keepdim的作用：保持维度
  - 假设我们有一个形状为[3, 4]的张量，我们在第一个维度上求和，如果保持维度，那么结果的形状将会是[1, 4]；如果不保持维度，那么结果的形状将会是[4]。

任务4. 补全Encoder模块（10分）

- 根据已经完成的层，补全encoder模块。
- 其中init部分已经完成，只需要补全forward部分。
- Positional encoding 已经在dataloader中部署，不需要再添加。
- 注意residual连接
- 完成后在测试代码5中测试，error应小于1e-5



架构如下：

输入 - 多头注意力 - 输出1 - 层规范化(输出1 + 输入) - dropout - 输出2 \
 - 前馈 - 输出3 - 层规范化(输出3 + 输出2) - dropout - 输出

任务5.在AddSub数据集上训练Transformer

- 数据示例

表达式：`BOS NEGATIVE 30 subtract NEGATIVE 34 EOS` 输出：`BOS POSITIVE 04 EOS`：这里的表达式是 $(-30) - (-34)$。这里符号 `+` 有两个含义：一个是表示数字的符号，另一个是两个整数之间的加法操作。为了简化神经网络的问题，我们用不同的文本标记表示它们。$(-30) - (-34)$ 的输出是 $+4$。这里的 `BOS` 和 `EOS` 分别指示序列的开头和结尾。

- Token转化
  - 需要将原始输入序列转换为可以用神经网络处理的格式

任务5.在AddSub数据集上训练Transformer

- 训练调优占25分
  - 可以调节超参数（如lr,dropout等）
  - 可以选择loss function
  - 可以调节Transformer里的维度（如num_heads, emb_dim, dim_feedforward等）
  - 可以调节优化器等。
- 限制：

1. 不能用预训练好的参数，同时请不要改变模型的架构

2. 训练，验证及测试数据集已经划分好，请不要将训练集和验证集混合进行训练

```
num_heads = 4
emb_dim = 32
dim_feedforward = 32
dropout = 0.2
num_enc_layers = 4
num_dec_layers = 4
vocab_len = len(vocab)
loss_func = CrossEntropyLoss
poss_enc = position_encoding_sinusoid
num_epochs = 10
warmup_interval = None
lr = 1e-3
```

任务5.在AddSub数据集上训练Transformer

- 按测试集上的准确率分段给分
- 准确率分段左闭右开
- 准确率看小数点后两位的四舍五入，如0.804算作0.80。

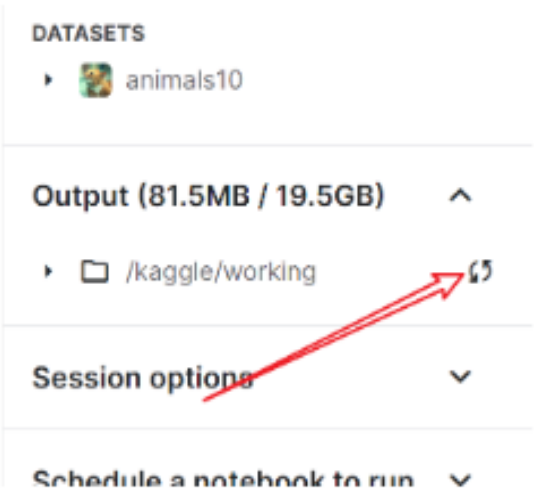
Baseline	Accuracy	Score
Vallina	[0.50, 0.63]	12
Simple	[0.63, 0.70)	14
Medium	[0.70, 0.74)	16
Hard	[0.74, 0.80)	20
Strong	[0.80, 1.00)	25

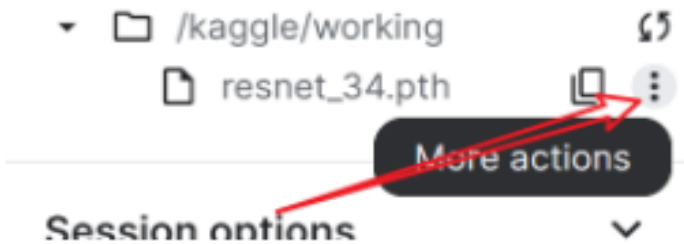
作业3：提交方式

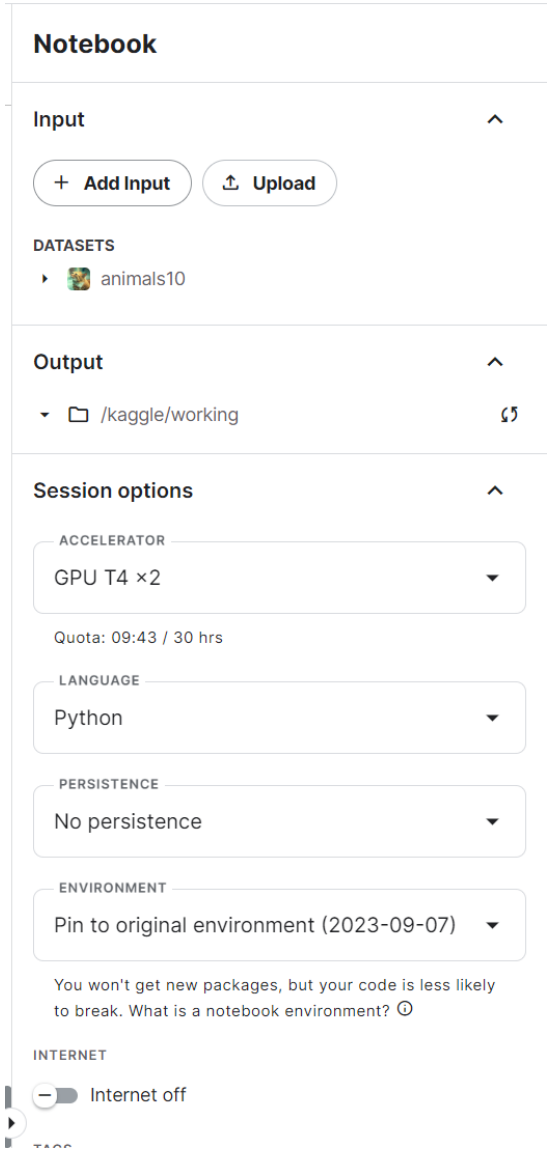
- 简答题
  - 用笔在稿纸上给出答案，或者提交word文档的截图。
  - 拍照，将其命名为：***学号_姓名_lab3_answer.jpg***
  - 在 eLearning 上提交
- 代码
  - 需要完成self-attention算法和Transformer结构的代码补全，相应的hint和TODO已经在代码中标明。
  - 重命名为：***学号_姓名_lab3.ipynb***
  - 在elearning上提交。

作业3：提交方式

- 代码填空题
  - 下载模型的操作如右图和下图
  - 模型重命名为：*学号_姓名_model.pth*
  - 作业截止后会抽取部分同学名单，被抽到的同学需要将模型文件发送至 yuxwang22@m.fudan.edu.cn







作业3：提交方式

- ddl：**2024.5.13 23:59**
- elearning 晚交一天倒扣 10 分
- 计算分数时以最后提交的版本为准（包括晚交的扣分）
- 若在完成作业的过程中遇到任何问题，均可通过各种方式提问
- **严禁抄袭，若发现抄袭者和被抄袭者本次作业均 0 分！**