Final Project

（本作业主要涉及使用 PyTorch 构建并训练基本的 Transformer 模型）

请在 6 月 19 日前于学在浙大提交

一、写在前面

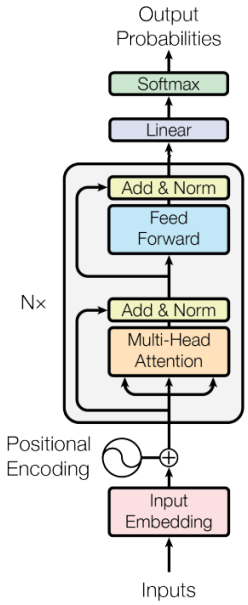
Transformer 可谓是当下最流行的深度学习模型之一，在各个领域（计算机视觉、自然语言处理等）的各个任务（分类、分割、翻译等）都有着不俗的表现，其变体层出不穷，推动了人工智能领域的研究发展。为了帮助大家建立对 Transformer 进一步的理解，在本次作业中，我们将尝试使用 PyTorch 构建和训练一个基本的 Transformer 模型，用于预测句子的情感分类。

二、SST-2 数据集介绍

SST-2（The Stanford Sentiment Treebank，斯坦福情感树库）数据集，是一个单句分类任务的数据集，包含电影评论中的句子和它们所包含的情感分类标签（正面情感标签为 1，负面情感标签为 0）。如下给出了一条数据样例，句子“this is one of polanski 's best films .”的标签为 1（正面情感）。这是一个二分类任务，对于输入到模型中的每一个句子，需要预测其情感分类。

```
1 this is one of polanski 's best films .
```

三、Transformer 模型结构与构建



具体的模型结构与细节请参考论文“Attention Is All You Need”，下面简单介绍一下本次作业相关的模块结构。

由于是分类任务，我们只需要用到 Transformer 的 encoder 部分。

首先在数据预处理阶段，我们通过一个预训练好的分词器将一个句子里的各个单词转换为其在词典里的下标，即将字符串转换为整型。将下标输入 embedding 层，从而将整型映射为高维向量。简单起见，在本次作业中我们不使用 Positional Encoding 而直接将 embedding 作为 Transformer encoder 的输入。

Transformer encoder 包含若干个 TransformerEncoderLayer（上图的灰色模块），分别包含一个 MultiHeadedAttention 和一个 PositionwiseFeedForward 模块，用于计算自注意力和进行前馈网络特征处理。在这些子模块后面各有残差连接（Add）和层归一化（Norm）。

然后我们将最后一个 TransformerEncoderLayer 的输出作为分类器的输入，分类器包含若干全连接层和 softmax 激活函数用于计算每个类别的概率。

此外，在本次作业代码中还加入了 dropout 来提升模型的泛化能力。

四、作业内容

1. 代码补充（必做）：请根据 Transformer 的模型结构图，在“src/models.py”中根据 TODO1~11 提示完善模型各个部分的代码；

2. 代码补充（bonus）：由于在 MultiHeadedAttention1 中我们用 for 循环来实现多个头的运算，这并没有真正发挥并行计算的优势，请思考如何直接用矩阵计算的方式来实现多头注意力，并完成 MultiHeadedAttention2 的 TODO12~13。完成后可以用 mhsa_parallel_check.py 来验证代码是否正确。

3. 训练参数选取：请测试在不同的 layer_num、batch_size、epochs、learning_rate、dropout_prob 的情况下，对模型训练的影响。可以用 tensorboard 曲线关注训练 loss 的下降快慢、是否收敛；比较 train_acc 和 val_acc 说明是否出现过拟合、欠拟合等问题；比较测试集上的准确率……

4. 选取一组你认为较优的参数，其在测试集上的准确率应达到 75%。

五、提交要求

请提交一个以 **final_project_学号.zip** 命名的 zip 格式压缩文件，此压缩文件应当包含：

1) 补充完整的 models.py；

2) 文件夹 src（模型的 checkpoint 文件 best.pt 可能较大，可以不用上传）；

3) 达到 75%准确率的模型训练 log 和 tensorboard 文件；

4) 作业报告 report.pdf。

作业报告要求：

1) 详见**四、作业内容**的 **3. 训练参数**选取；

2) 如果完成了 bonus，请描述其你所实现的并行计算方法的正确性，包括理论上矩阵

维度与多头的对应关系，以及 mhsa_parallel_check.py 运行结果；

3) 请画出含 dropout 层的 Transformer 模型结构图；

4) 报告的具体格式没有要求。

九、成绩评定

代码补充实现占 40%，report 占 60%，bonus 额外 10%。

十、Q&A

1. 如何使用不同的超参配置运行实验？

各项超参定义在了 main.py 中，具体如下：

- layer_num：TransformerEncoderLayer 的数量，即 Transformer 结构图中的 N；
- dropout_prob：dropout 的概率为 0 时不使用 dropout；
- learning_rate：学习率；
- max_grad_norm：若梯度 norm 超过该值，会进行截断；
- batch_size：一个 batch 的数据量；
- epochs：训练的 epoch 数量；
- checkpoint_dir：模型 checkpoint 保存的目录。

```
if __name__ == '__main__':
    start_time = time.time()
    parser = argparse.ArgumentParser(description='train process')
    parser.add_argument('--layer_num', default=2, type=int, help='the number of layers in transformer encod
    parser.add_argument('--dropout_prob', default=0.1, type=float, help='dropout probability')
    parser.add_argument('--learning_rate', default=1e-5, type=float)
    parser.add_argument("--max_grad_norm", default=1.0, type=float, help="max gradient norm")
    parser.add_argument('--batch_size', default=16, type=int)
    parser.add_argument('--epochs', default=10, type=int)
    parser.add_argument('--checkpoint_dir', default='./checkpoint', type=str, help='the directory to save c
    args = parser.parse_args()
```

如果在命令行不指定具体的超参，则会使用上图各项 default 里定义的默认值。

举个例子，进入 src 目录，在终端命令行输入以下内容：

```
python main.py --layer_num 2 --dropout_prob 0
```

对应的实验配置为使用 2 层 TransformerEncoderLayer，不使用 dropout，初始学习率为 0.00001，截断的梯度 norm 为 1，batch_size 为 16，训练 10 个 epoch，模型 checkpoint 的保存路径为./checkpoint。

2. 如何检查并行模块的代码是否正确以及比较其和原来代码的耗时？

```
python mhsa_parallel_check.py --function check
```

执行以上命令会输出如下结果：

```
output same rate: 100.00%
query gradient same rate: 100.00%
key gradient same rate: 100.00%
value gradient same rate: 98.04%
```

分别表示两种多头注意力实现的输出、对输入的query/key/value的回传梯度的差别，要求 output same rate 需达到 100%，query/key/value gradient same rate 达到 95% 即可。

```
python mhsa_parallel_check.py --function time_comp
```

执行以上命令会比较两种实现方式的耗时：

```
for循环计算mhsa用时: 6.29ms, pytorch矩阵并行计算mhsa用时: 2.52ms
```

3. 如何安装本次作业代码依赖的第三方库、使用 tensorboard 等？

请自行查阅网络资料，或参考 Homework5 里的相关说明。