# 实验3（附加）：词嵌入的原理与预训练词向量使用报告

# 实验背景

本实验通过卷积神经网络（TextCNN）对文本进行分类，并进一步探索不同词嵌入方式对模型性能的影响。实验所使用的数据集为THUCNews，它是基于新浪新闻RSS订阅频道2005至2011年期间的历史数据生成的中文文本分类数据集。数据集中包含财经、彩票、房产、股票等14个类别。由于数据量较大，本实验对其进行了采样，选取了180,000条作为训练集，10,000条作为验证集和10,000条作为测试集。

实验模型TextCNN能够捕捉文本的局部特征（如N元语法），其核心思想是通过卷积操作提取文本中若干单词组成的滑动窗口内的特征。N-gram特征在自然语言处理中已经被广泛证明为有效的文本特征，而TextCNN能够进一步对这些特征进行自动组合和筛选，获得更抽象的语义信息。

此外，实验使用了预训练的Elmo词向量模型，Elmo是一种深度双向语言模型，它通过将句子中的单词转化为可训练的多维词向量，捕捉其在不同上下文中的语义。

# 实验方案设计

在先前的实验三中，已经构建一个基础的TextCNN模型，此次实验对其进行了改进，将原有的nn.Embedding()词嵌入模块替换为Elmo预训练词向量。在此过程中需要对数据预处理、模型输入层和卷积层等部分进行修改，使其适应Elmo词向量的不定长特性。

# 实验过程

实验首先构建了一个基础的TextCNN模型，该模型使用三层卷积操作分别提取不同尺寸的滑动窗口（2-gram、3-gram和4-gram）内的特征，并通过最大池化层将这些特征缩减至固定长度，最后通过全连接层进行分类。

对于THUCNews数据集，实验通过read\_data函数读取文本和标签数据，并通过built\_corpus函数生成词到索引的映射（word\_2\_index），并对文本数据进行分词和向量化处理。

训练过程中，使用了交叉熵损失函数（CrossEntropyLoss）和AdamW优化器。为了使用Elmo词向量，实验加载了预训练的Elmo模型并进行了词向量的生成。由于Elmo的预训练词表不可修改，我们通过AllenNLP中的batch\_to\_ids方法将输入句子转化为适合Elmo模型的输入格式，最终将生成的词向量作为TextCNN的输入。另外实验还对数据预处理、模型输入层和卷积层等部分进行修改，使其适应Elmo词向量的不定长特性

# 性能测试

在修改实验代码的过程中，使用了tqdm库实时计算并累加loss值，以直观显示训练进度；在训练过程中，模型的损失函数逐渐下降，验证集上的准确率逐步提升。根据输出日志，每个epoch的训练损失和验证准确率如下：

Epoch 1, Training Loss: 0.5546, Validation Accuracy: 0.7670

Epoch 2, Training Loss: 0.5015, Validation Accuracy: 0.7801

Epoch 3, Training Loss: 0.4760, Validation Accuracy: 0.7904

Epoch 4, Training Loss: 0.4537, Validation Accuracy: 0.7979

Epoch 5, Training Loss: 0.4368, Validation Accuracy: 0.7998

Epoch 6, Training Loss: 0.4154, Validation Accuracy: 0.7942

Epoch 7, Training Loss: 0.4035, Validation Accuracy: 0.8008

Epoch 8, Training Loss: 0.3838, Validation Accuracy: 0.7942

Epoch 9, Training Loss: 0.3582, Validation Accuracy: 0.7989

Epoch 10, Training Loss: 0.3422, Validation Accuracy: 0.7937

基于实验三中构建的模型，实验对两种方法进行了比较。实验比较了使用不同词嵌入方法的模型在测试集上的表现。但是实验效果欠佳，两方法准确率均在80%浮动。

经过资料查找，推理阶段要将ELMo的预训练词向量和原本的词向量相拼接，即：



但本实验代码的class TextCNNModel(nn.Module) 部分似乎没有这部分操作，可能制约了模型的准确度。

# 实验习题

1. 简述为何要执行embeddings = torch.unsqueeze(embeddings, dim=1)，包括unsqueeze操作的效果，及对embedding进行unsqueeze操作的理由

**答：**执行 embeddings = torch.unsqueeze(embeddings, dim=1) 的目的是为了增加词嵌入张量的维度，以适应卷积神经网络（CNN）输入的格式需求。在使用torch.unsqueeze(embeddings, dim=1) 之后，原本的词嵌入张量 embeddings 将增加一个新的维度，在文本处理中表示文本的“通道”数量。在TextCNN模型中，卷积操作需要输入的数据形状为 [batch\_size, channels, height, width]，因此 unsqueeze 操作将词嵌入张量的形状转换为符合卷积操作要求的格式。

理由：在 TextCNN 中，卷积操作需要输入的张量具有四个维度：批量大小 (batch\_size)、通道数 (channels)、序列长度 (max\_len) 和词嵌入维度 (embedding\_dim)。因为卷积核需要在二维空间上进行滑动（即在句子的不同词汇和特征维度之间进行操作），张量必须包含一个表示“通道”的维度。然而，默认情况下，embeddings 的输出是一个三维张量，不包含“通道”维度。因此，必须通过 unsqueeze 操作在 dim=1 位置增加一个通道维度，使数据能够输入到卷积层中进行计算。