音频-视频联合文本转录

1 背景介绍

字幕，已经是如今视频中不可或缺的组成部分之一，然而制作字幕一直是一项繁重的工作，不仅耗时耗力，有时还对制作人员的语言水平提出了较高的要求。随着人工智能的迅速发展，如今自动文本转录的技术已经十分成熟了，本次作业将介绍现代文本转录方法是如何构建的，并且分别介绍如何基于视频和音频-视频两种模态分别完成文本转录。通过本次作业，你会学到

- 如何处理视频、音频、文本数据，将其转化为训练可用格式；
- 如何构建视频、音频的编码网络，结合不同模态的融合网络，以及针对文本的解码网络；
- 如何计算驱动网络训练的损失函数；
- 训练完成后，如何进行推理；
- 基于视频的文本转录器和基于音频-视频的文本转录器。

我们提供了完成本次作业需要的大部分代码框架，你需要按照本文档的说明完成其中部分功能，并且根据代码和实验结果回答问题，撰写报告。

2 作业说明

提交方式：每个小组需要提交一个单独的 ZIP 文件，命名为 “大作业 _ 小组编号.zip”。压缩包中需要包含可以运行的完整代码和报告文档。
报告内容：除了本文档中要求的内容外，请写出每位小组成员在大作业中完成的具体工作。

3 基于视频的文本转录

我们首先从更加简单的基于视频的文本转录任务开始。基于视频的文本转录也被称为唇读(lip reading)，即只根据人物在说话时的嘴唇变化视频来解读文本内容，这个任务非常有趣，但是即使是对人来说也是非常具有挑战性的。

3.1 数据处理

该任务需要人物说话的嘴唇变化视频作为输入，对应文本作为标签。我们已经对每个视频做过预处理，将其中人物的嘴唇附近区域截取出来保存为对应的视频。现在，给定视频路径和对应的文本路径，你需要加载视频和文本，并将其处理成神经网络可以使用的格式，具体任务如下。

视频本身的数据格式较为规整，因此它的加载处理方式比较简单，通常我们只需要读取视频，将其储存为多维 tensor，最后再进行一些归一化即可。请按照以下说明完成 dataset.py 文件中的 AudioVideoDataset 类的 load_video 方法：

1. 根据给定的 video_name，读取视频（你可以使用 cv2.VideoCapture 方法来完成这一过程）；

2. 将每一帧变为单通道的灰色图像（你可以使用 cv2.cvtColor 来完成，请注意 opencv 在加载图像时，默认的通道顺序为 BGR）；
3. 将视频转化为 numpy array，并且对其进行视频数据变换（数据变换被定义在 self.transform 中）；
4. 输出的特征维度为（T，H，W，1），T 表示视频帧数，H 和 W 分别表示图像的高宽，1 是人为增加的维度，方便后续处理。

Task 1
请实现 AudioVideoDataset 类的 load_video 方法。（完成代码即可，不用在报告中写文字说明）

文本数据通常需要更加细致的处理。因为文本不像图像有天然的数值表示（RGB），在对文本进行计算之前，我们需要为其设计一种便于使用的数值形式。现在通用的做法分为两步：首先切分文本，将其划分成离散的词符（token）；然后对不同的词符建立词表（vocabulary），以整数来代表词符。其中第一步分词（tokenization）最为关键，简单的基于字母的分词或者基于单词的分词都存在问题，大家使用最多的是一种叫做 byte-pair-encoding（BPE）的分词技术。BPE 的代码实现已经非常成熟，这里我们使用 sentencepiece 库提供的实现版本。

请按照以下说明完成 dataset.py 文件中的 AudioVideoDataset 类的 get_label 方法：

1. 根据给定的 index，读取标签文本。你可以通过 index 在 self.label_offsets_list 查询需要的文本在标签文本中的位置，得到元组 (s, e) 表示起始和终止位置。你可以使用 python 文本 IO 的 seek 和 read 函数来读取对应的部分；
2. 对标签文本进行分词（你可以使用 self.bpe_tokenizer 来完成，参考这部分代码）；
3. 对分词后的文本进行编码（你可以使用 self.dictionary 来完成，参考这部分代码，注意如果某个词符不在词表中，请不要向词表中加入新词）；
4. 输出的标签是维度为（L）的长整型数组，L 表示词符个数。

Task 2
请实现 AudioVideoDataset 类的 get_label 方法。（完成代码即可，不用在报告中写文字说明）
阅读 BPE 的论文（论文链接），简要说明其构建过程，以及其相比基于字母的分词和基于单词的分词的优势。

3.2 模型构建

接下来可以开始构建模型。我们采用的模型结构如图 1所示，首先利用 resnet 网络对每一帧图像进行编码，抽取每帧图像的全局特征；接着用 transformer 模型将图像特征映射到文本空间，因为文本标签已经转化为整数，最后直接利用交叉熵作为损失函数驱动训练。

ResNet 网络是通用的视觉信号处理模型，通过著名的残差连接能够有效训练深层网络，抽取视觉特征。请按照以下说明完成 modality_encoder.py 文件中的 BasicBlock 类和 ResNet 类：

1. BasicBlock 类是堆叠 ResNet 网络的最小单元，他的基本结构如图 2所示，注意残差连接通常如左图所示是一个直连，但如果特征的维度发生改变，则会变为右图；
2. 请完成 BasicBlock 类的 ___init___ 函数，inplanes，planes，stride 为 conv1 的输入通道，输出通道，以及步长；conv2 使用 planes 作为输入输出通道数，1 作为步长即可；

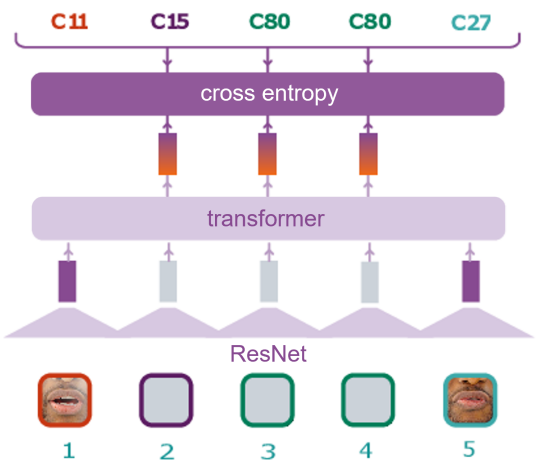

图 1: 基于视频的文本转录模型

3. 请完成 BasicBlock 类的 forward 函数，注意需要处理图中展示的两种情况；
4. 请完成 ResNet 类的 _make_layer 函数，该函数负责构建 resnet 网络中的一个 stage，block 为 BasicBlock 类，planes 为该 stage 的输出通道数，blocks 为该 stage 的 block 数量，stride 为该 stage 的降采样率（降采样通过 conv 实现，所以也是步长）。请注意每个 stage 的输入通道数需要自己计算，最开始的输入通道数为 self.inplanes。

Task 3
请实现 BasicBlock 类的 ___init___ 方法和 forward 方法，与 ResNet 类的 _make_layer 方法。(完成代码即可，不用在报告中写文字说明)

提示：我们提供了预训练的特征提取器（详见第 4章）。如果你的实现是正确的，预训练的特征提取器可以正确加载，即每个参数的名称和尺寸是对应的。你可以关注加载预训练 checkpoint 时返回的信息，也可以将预训练 checkpoint 中的参数名称和尺寸打印出来，来确认你的实现是否正确。

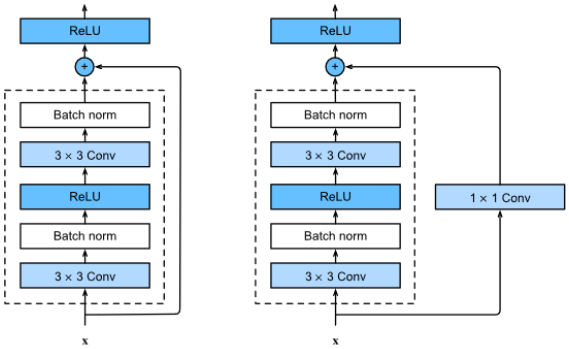

图 2: ResNet 网络的 BasicBlock

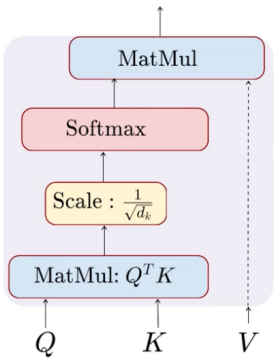

图 3: Transformer 网络的 attention 计算

Transformer 网络从自然语言处理领域诞生，因其通用性逐渐地被拓展到其他场景中。Transformer 网络通常由 encoder-decoder 两部分组成，encoder 对输入信号做进一步的编码处理，decoder

则根据 encoder 的输出来解码得到预测。两部分网络虽然功能上有所区分，但都是建立在 multi-head attention 这一基础模块上。

multi-head attention 的通用计算表示为图 3，Q 表示 query，K 和 V 表示 key 和 value。请按照以下说明完成 multihead_attention.py 文件中的 MultiheadAttention 类：

1. 按照图示完成 attention 的计算过程，代码中 attn 表示最后的输出，attn_weights 表示 value 的加权和矩阵，你可以不需要 scale 的过程，我们已经在 Q 的计算中考虑了；
2. 在原始的计算过程中，加入对 attn_mask 的处理，attn_mask 表示 query（对应行）能够看到哪些 key（对应列），如果输入有 attn_mask，它的矩阵元素要么是 0 要么是-inf，0 的位置表示对应的 query 可以看到 key，而-inf 的位置表示对应的 query 不能看到可以。请结合 attn_mask 使得 query 不会使用指定 key 的信息；
3. 在原始的计算过程中，加入对 key_padding_mask 的处理，key_padding_mask 指明了哪些输入只是起到填充占位的作用，在 attn 的计算时不应该考虑这些符号。如果输入中有 key_padding_mask，它的矩阵元素由 0、1 组成，0 的位置表示非占位符，1 的位置表示占位符。请结合 key_padding_mask，保证 attn 的计算过程不会使用占位符的特征；
4. 在上述任务中，请注意对 multihead 的处理，最开始使用的 Q，K，V 的维度为（batch_size× num_head, seq_len, feat_dim）；
5. 如果需要返回注意力权重 attn_weights，即 need_weights 为 True 时，你需要返回所有 head 的注意力权重的平均值，维度为（batch_size, tgt_len, src_len）。

Task 4
请实现 MultiheadAttention 类的 forward 方法。（完成代码即可，不用在报告中写文字说明）

3.3 损失函数

本次项目使用交叉熵损失函数来训练网络。请按照以下说明完成 losses/label_smoothed_cross_entropy.py 文件中的 label_smoothed_nll_loss 函数：

1. 阅读 pytorch 关于交叉熵损失函数的介绍链接，实现基本的交叉熵计算，代码中 lprobs 表示 log probability，target 表示标签，reduce 表示是否要对所有 loss 求和；
2. 在原始的计算过程中，加入对 ignore_index 的处理，ignore_index 表示需要忽略的 target 的取值，不加入到 loss 计算中；
3. 在原始的计算过程中，加入标签平滑（label smooth），标签平滑是一种有效防止过拟合，提升模型泛化能力的技术，请阅读论文链接中和标签平滑相关的部分，并且实现代码；
4. 在上述任务中，请不要使用 pytorch 提供的交叉熵损失函数。

Task 5
请实现 label_smoothed_nll_loss 函数。（完成代码即可，不用在报告中写文字说明）

4 开始训练

在完成上述的代码之后，我们就可以开始准备训练了。训练的一些说明如下

- 训练的启动脚本为 train.sh，你可以运行该脚本来启动训练，不过在那之前，你需要指定好参数；
- –config-name 表示你用来训练的配置文件名，在 configs 文件夹下我们提供了两个配置文件，请选择你需要的配置文件；
- task.data 与 task.label_dir 表示数据和标签的路径,我们将数据放在了/data2/final_project_data/下，请将这两个路径都设置为/data2/final_project_data/30h_data；
- task.tokenizer_bpe_model 表示分词器的模型路径，请设置为/data2/final_project_data/spm_1000/spm_unigram1000.model；
- model.pretrained_path 表示预训练模型的路径，为了加快训练速度，我们为大家提供了预训练的特征提取器，预训练 + 微调的训练方法也是很多领域的常用方法。请将路径设置为/data2/final_project_ckpt/pretrained_model.pth；
- hydra.run.dir 表示实验结果的储存路径，可以自由设置，请注意训练程序会自动读取实验目录下的 checkpoint（如果存在）并继续训练，如果你想从头开始训练，请改变实验目录或者删除原有的 checkpoint。

Task 6
请完成基于视频的文本转录任务训练，绘制训练和测试的损失函数与准确率曲线。

5 基于音频-视频的文本转录

在视频的基础上，我们还可以利用音频信息进行文本转录。基于音频-视频信息的文本转录可以取得很好的效果，基本能够满足自动字幕生成的要求。这一任务只需要在基于视频的文本转录模型基础上加入音频信息即可完成。

5.1 数据处理

音频数据的处理比较特殊，大多数方法不会将原始的音频信号直接作为网络输入，而是将其频谱作为输入，其中变换方式也非常多样（比如梅尔谱）。请按照以下说明完成 dataset.py 文件中的 AudioVideoDataset 类的 load_audio 方法：

1. 根据 audio_name 加载音频数据（你可以使用 wavfile 来完成）；
2. 使用 logfbank 算法提取音频特征（你可以使用 python_speech_features 来完成）；
3. 将每 4 帧（由 stack_order_audio 参数确定）的音频特征拼起来组成这 4 帧对应时间的共同特征，如果总帧数不是 4 的倍数，少的部分补零；
4. 最后返回的音频特征维度应为（T，C），T 表示帧数（每 4 帧拼起来后的数量），C 表示特征维度。

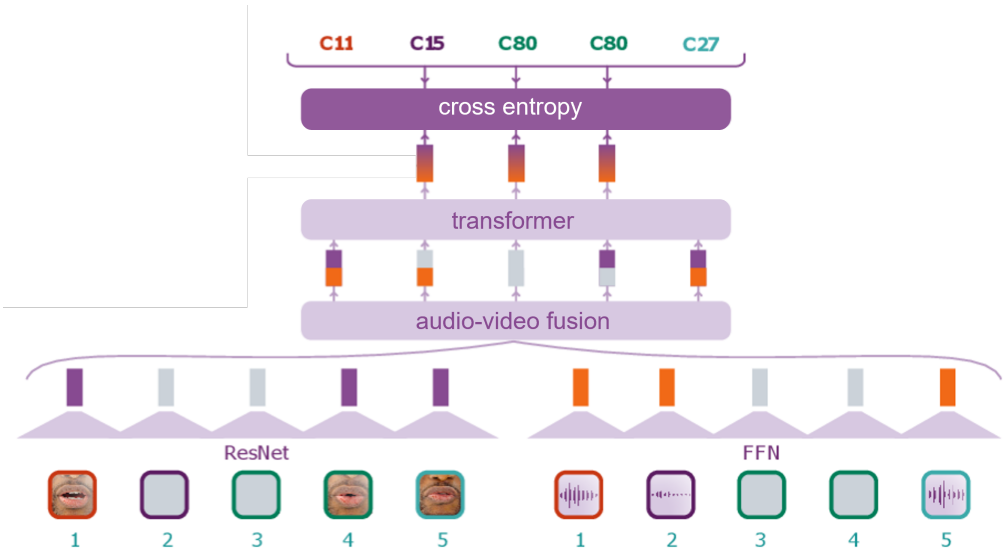

图 4: 基于音频-视频的文本转录模型

Task 7
请实现 AudioVideoDataset 类的 load_audio 方法。（完成代码即可，不用在报告中写文字说明）
调研 logfbank 算法，简要说明其计算过程。

5.2 模型构建

处理音频的网络结构没有视频的网络那么复杂，通常只需要一层线性层来调整特征的维度。请按照以下说明完成 modality_encoder.py 文件中的 AudioEncoder 类：

1. 完成 ___init___ 方法，创建线性层，输入维度为 audio_feat_dim，输出维度为 encoder_embed_dim；
2. 完成 forward 方法，请注意输入的维度为（B，C，T），B 表示批次大小，C 表示特征维度，T 表示帧数。

Task 8
请实现 AudioEncoder 类。（完成代码即可，不用在报告中写文字说明）

得益于 Transformer 的通用性，模态融合模块不需要改动，只需要把输入的音频特征和视频特征拼起来即可，之后的网络结构和原来是完全一样的。

Task 9
请完成基于音频和视频的文本转录任务训练，绘制训练和测试的损失函数与准确率曲线。

6 开始推理

模型的训练完成后，我们就可以实际用他们来自动生成字幕了。推理是通过自回归的方式完成，即每次生成一个词符，生成之后将该词符加入已经生成的内容中作为 decoder 部分的输入继续生成下一词

符。这种生成方式虽然速度快，但是可能陷入局部最优解，因此通常使用的解码方式是一种叫做 beam search 的搜索算法。

- 推理的启动脚本为 test.sh，你需要指定一下参数：
- common_eval.path 为训练好的模型 checkpoint 的路径，最好使用绝对路径。
- common_eval.results_path 为保存结果的路径，最好使用绝对路径。
- override.modalities 表示推理时使用什么模态的信息，可以设置为 ['video'] 或 ['video','audio']。

Task 10
请完成基于视频、基于音频和视频的文本转录推理，分析推理结果，以及推理准确率和之前得到的测试准确率的差异。
请阅读 sequence_generator.py 的 _generate 方法，简要说明 beam search 的具体过程。

7 参考结果

我们在表格 1中提供了 Baseline 结果供参考。当正确完成以上任务后，你应该能得到接近的结果。

模型	train loss	train acc	valid loss	valid acc	inference WER
视频	59.95	67.85%	63.71	66.65%	46.46
视频 + 音频	33.91	94.40%	37.33	91.55%	15.17

表 1: 基本框架的 Baseline 参考

8 改进创新

以上就是基于音频和视频的文本转录的基本框架，你可以用它完成效果不错的自动字幕生成，但是依然存在不小的提升空间。下面，请你思考如何进一步提升这个模型的效果。请注意以下事项：

- 不要只是关注模型结构的改变，你可以改进的入手点有很多：数据预处理、损失函数、训练流程、推理过程等等，都有潜在的提升空间；
- 不要只是关注最终准确率，你也可以提升模型的其它方面能力：速度、小样本场景的快速迁移等等；
- 围绕你的改进创新，请设计严谨的对比实验，尽可能展示你的改进的效果；
- 最终的报告请写清楚改进的动机、具体方法、实验设计与结果展示，如果有涉及相关文献，也请按照学术规范引用。

Task 11
请提出你的改进创新，并设计实验进行验证，完成完整的实验报告。