深度学习课程报告

第 11 组

小组成员：71118415 叶宏庭

71118416 南　希

71118417 刘隆琦

71118420 牛熠玮

71118320 陈　瀚

71117224 宋旻晖

2021　年 1 月　7 日

目录

一、引言.....3
二、课程任务.....3
（一）视频描述数据集制作.....3
1 任务介绍.....3
2 任务实践.....3
（二）视频标注算法调试与竞赛.....5
1 任务介绍.....5
2 实验数据集.....5
3 模型介绍.....7
4 词向量提取.....9
5 视频特征提取.....10
6 模型结构.....12
7 模型训练.....12
8 模型评估.....13
9 实验总结.....13
（三）CycleGAN 风格迁移.....14
1 CycleGAN 简介.....14
2 CycleGAN 原理.....14
3 实验过程.....15
4 结果展示.....16
5 实验总结.....17
（四）课堂展示内容（语音识别）.....18
1 领域简介.....18
2 问题描述.....18
3 模型介绍.....22
4 总结.....33
三、课程总结与反思.....33
(一) 课程任务完成情况.....33
(二) 课程任务难点与反思.....34

一、引言

本报告将针对深度学习课程的全部任务进行综述，任务包括：视频描述数据集制作、视频标注算法调试与竞赛、CircleGAN 风格迁移算法调试、深度学习课程展示。

二、课程任务

（一）视频描述数据集制作

1 任务介绍

本任务的目的是制作一个可以进行视频描述的数据集，主要工作就是对所有的短视频进行一个简短描述，并且保存结果，作为任务二的数据集。

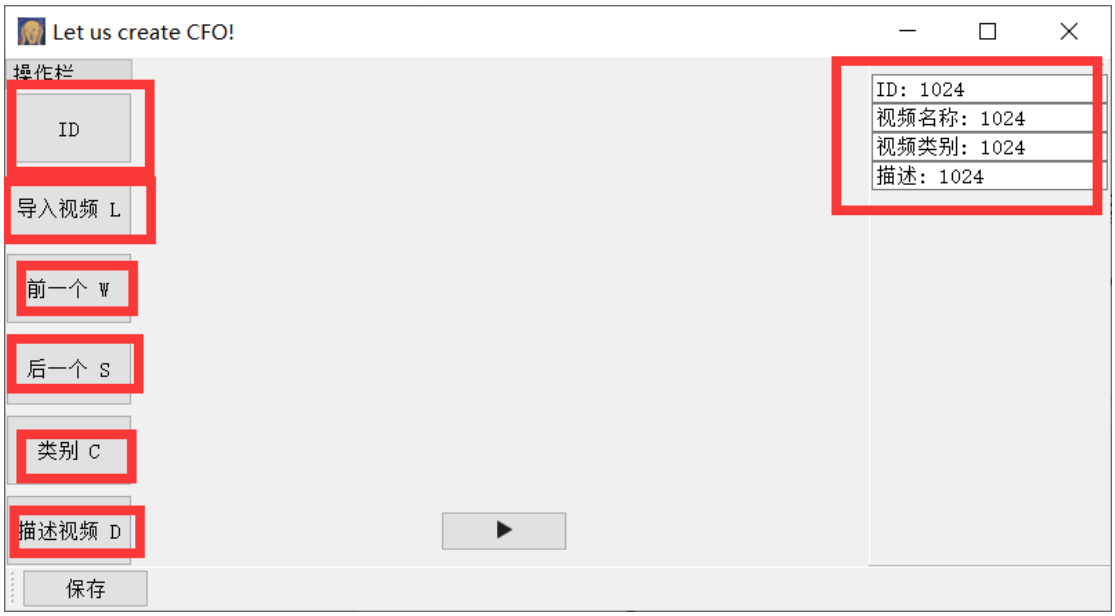
2 任务实践

2.1 环境要求

由于本任务所采用的程序由 Python 编写，因此对环境提出以下要求：
基础环境：Python3
需要安装包：sip，PyQt5

2.2 任务过程

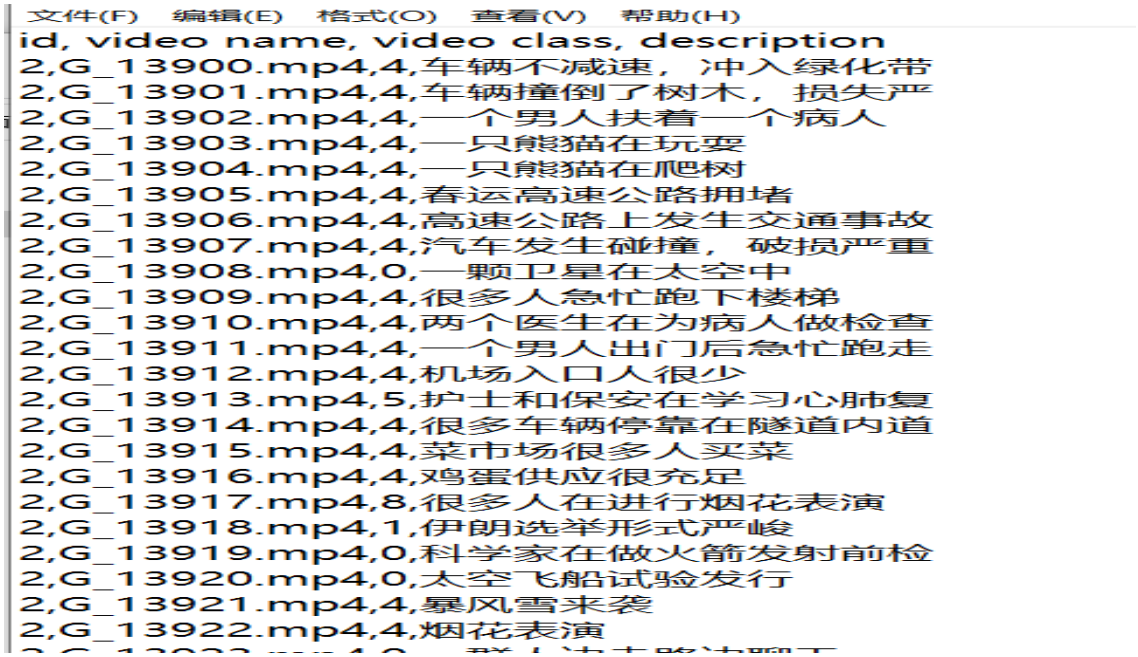
1 运行所给文件夹下的 window.py 文件
cmd 命令内输入：python window.py
2 得到下图所示的操作画面



- 在 ID 按钮中输入自己的组内 ID，用于区分不同组员的描述；
- 导入视频按钮，选择视频所在文件夹，导入视频；
- 前一个，后一个按钮用于切换视频；
- 类别按钮用于输入该视频的类别
- 描述视频按钮用于输入对视频的描述
- 右侧显示目前视频的信息

3 得到最终数据结果

最终数据结果保存在 label 的文件夹下，为一个 x.csv 文件，其中 x 为本人的组内 ID。


文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
id, video name, video class, description
2,G_13900.mp4,4,车辆不减速， 冲入绿化带
2,G_13901.mp4,4,车辆撞倒了树木， 损失严
2,G_13902.mp4,4,一个男人扶着一个病人
2,G_13903.mp4,4,一只熊猫在玩耍
2,G_13904.mp4,4,一只熊猫在爬树
2,G_13905.mp4,4,春运高速公路拥堵
2,G_13906.mp4,4,高速公路上发生交通事故
2,G_13907.mp4,4,汽车发生碰撞， 破损严重
2,G_13908.mp4,0,一颗卫星在太空中
2,G_13909.mp4,4,很多人急忙跑下楼梯
2,G_13910.mp4,4,两个医生在为病人做检查
2,G_13911.mp4,4,一个男人出门后急忙跑走
2,G_13912.mp4,4,机场入口人很少
2,G_13913.mp4,5,护士和保安在学习心肺复
2,G_13914.mp4,4,很多车辆停靠在隧道内道
2,G_13915.mp4,4,菜市场很多人买菜
2,G_13916.mp4,4,鸡蛋供应很充足
2,G_13917.mp4,8,很多人在进行烟花表演
2,G_13918.mp4,1,伊朗选举形式严峻
2,G_13919.mp4,0,科学家在做火箭发射前检
2,G_13920.mp4,0,太空飞船试验发行
2,G_13921.mp4,4,暴风雪来袭
2,G_13922.mp4,4,烟花表演
```

描述结果

4 整理全部描述结果

整理全部组员的描述结果，最终提交给助教，生成最后的完整数据集。

（二）视频标注算法调试与竞赛

1 任务介绍

视频分类算法能实现自动分析视频所包含的语义信息、理解其内容，对视频进行自动标注、分类和描述，达到与人媲美的准确率。视频分类的主要目标是理解视频中包含的内容，确定视频对应的几个关键主题。视频分类（Video Classification）算法将基于视频的语义内
容如人类行为和复杂事件等，将视频片段自动分类至单个或多个类别。视频分类不仅仅是要理解视频中的每一帧图像，更重要的是要识别出能够描述视频的少数几个最佳关键主题。视频分类的研究内容主要包括多标签的通用视频分类和人类行为识别等。与之密切相关的是，视频描述生成（Video Captioning）试图基于视频分类的标签，形成完整的自然语句，为视频生成包含最多动态信息的描述说明。本次实验主要关注研究融合视频本身的空间和时间特征，也称为基于视觉的视频分类。

图像描述（image captioning）是为一张图像生成一句描述，视频描述（Video captioning）
与其类似，是为一个短视频片段（video clip）生成一句描述，往往这个短视频包含多帧视频
图像，所以相比于图像描述，视频描述更要考虑帧与帧之间的关系（时序因素）对生成句子的影响，这也是视频描述特有的一个难点。

深度网络为解决大规模视频分类问题提供了新的思路和方法。卷积神经网络（Convolutional Neural Networks, CNN）采用卷积与池化操作，可以自动学习图像中包含的
复杂特征，在视觉对象识别任务中表现出很好的性能。将视频帧视为一张张静态图像，应用 CNN 识别每一帧，然后对预测结果进行平均处理来作为该视频的最终结果。

本次实验参考 GitHub 开源项目 xiadingZ/video-caption.pytorch
（https://github.com/xiadingZ/video-caption.pytorch）

2 实验数据集

2.1 数据集介绍

本次实验的数据集是来自课程第一次任务的视频描述数据集制作，同学们对视频数据进行描述，并且采用 Google 翻译将中文描述译为英文，最后进行整理得到该数据集。

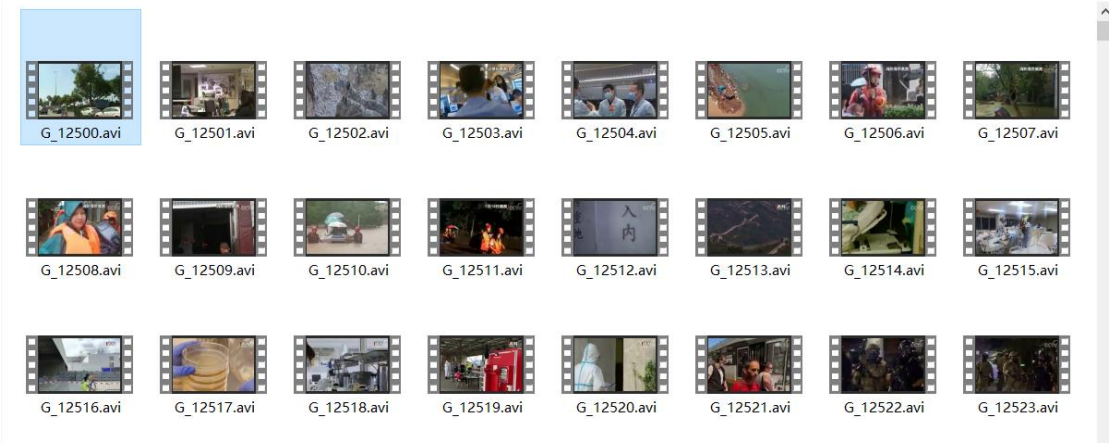
视频数据为央视的《共同关注》播出的部分新闻视频，每个视频被分割成若干个长为几秒到几十秒不等的短视频，同学们对每一个短视频进行描述，一个短视频将作为一个数据记录存储。

实验数据集共 2400 条视频与描述记录，其中前 1920 条数据用作训练集，后 480 条数据

用作训练，并且最终参加网上竞赛。

2.2 数据集格式

本实验数据集参照 MSR-VTT 数据集，视频信息以及视频描述以 JSON 格式保存，视频文件单独存放，视频文件格式为.avi 文件。



数据集视频文件

视频的信息存放在 JSON 文件的 videos 对象下，字段信息包括：

- category（视频分类）：分类包含时政、国际、军事、警法、社会、公益、教育、财经、娱乐、文化；
- url：视频来源网址；
- video_id：视频的 id 标识
- start_time：视频片段在原视频中的开始时间
- end_time：视频片段在原视频中的终止时间
- split：标识该条数据属于训练集或测试集；
- id：视频的 id 标识


```
{
  "category": 4,
  "url": "http://tv.cctv.comhttps://tv.cctv.com/2020/07/19/VIDEzj3L7VW5VBcn3BX59HGa200719.shtml",
  "video_id": "G_12500",
  "start time": "0:0:35,13",
  "end time": "0:0:43,9",
  "split": "train",
  "id": 12500
},
```


视频信息格式

所有的视频描述信息存放在 sentence 对象下，每个 video_id 对应五条 caption，即五位同学对该条视频的描述。同一个 video_id 的不同描述采用 sen_id 进行区分。

```
{
  "video_id": "G_12500",
  "caption": "Nearly 1000 new crown cases reported in California kindergartens",
  "sen_id": 0
},
{
  "video_id": "G_12500",
  "caption": "The epidemic situation in the United States is serious",
  "sen_id": 1
},
{
  "video_id": "G_12500",
  "caption": "The traffic police are directing the traffic.",
  "sen_id": 2
},
{
  "video_id": "G_12500",
  "caption": "New coronavirus diagnosis increases",
  "sen_id": 3
},
{
  "video_id": "G_12500",
  "caption": "The epidemic is getting worse in the United States",
  "sen_id": 4
},
```


视频描述格式

2.3 数据集划分

本次实验，数据集总共 2400 条视频以及对应的描述，为了使得模型训练能够取得更好的结果，我们采用 0-1919，共计 1920 条数据进行模型训练，对 1920-2399，共计 480 条视频描述进行预测。

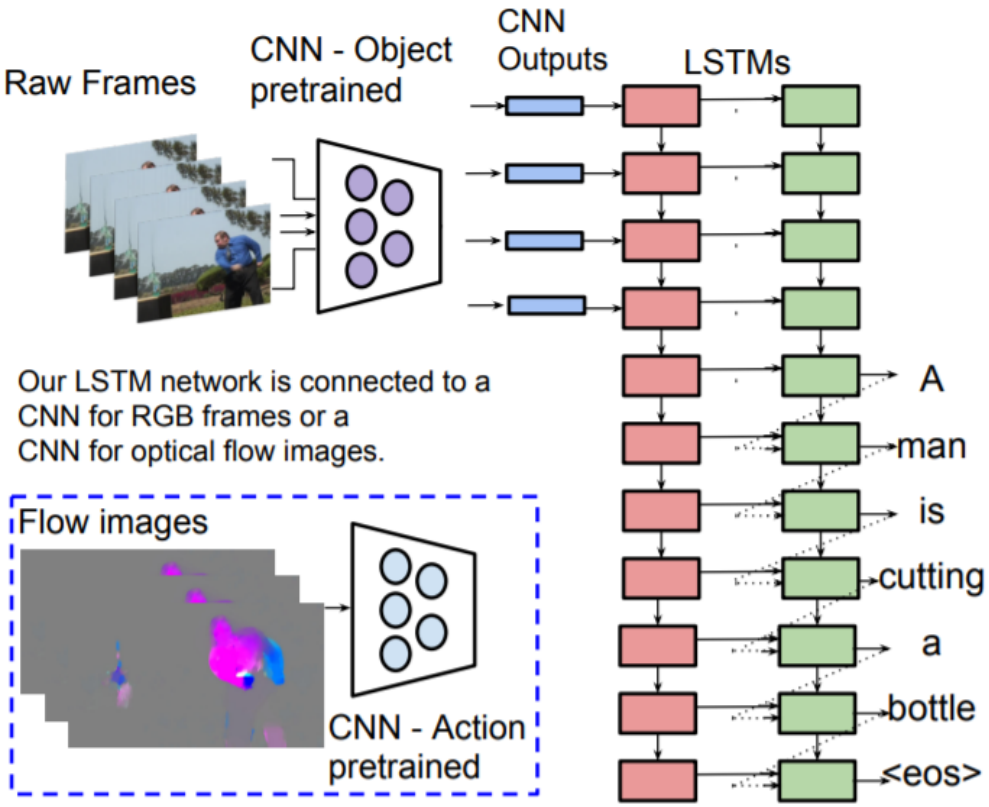
训练集	1920
测试集	480
共计	2400

3 模型介绍

3.1 S2VT 模型

本实验采用的视频标注模型是 Sequence to Sequence Video to Text 模型，是 Venugopalan 等人在 2015 发表的论文中所提出的对视频帧序列输入、文字序列输出的一个端到端视频描述模型。这篇论文提出的利用 LSTM 解决视频与文字可变长度的解决思路，以及整个视频描述的 S2VT 网络结构设计都是比较经典的，很多最新几年发表的视频描述相关的论文中都看到了 S2VT 的影子。

S2VT 这个新模型，它直接学习从帧序列到单词序列的映射关系。如图所示，S2VT 由两个 LSTM 网络叠加而成，第一个 LSTM 将通过卷积神经网络提取到的帧特征向量作为输入进行逐个编码。一旦读取完所有的帧，模型会逐个单词的生成一个句子。帧和单词表示的编码和解码工作由平行语料库学习得到。为了更好地表示视频中活动的时序特点，模型还计算了连续帧之间的光学流，流图像也是先通过 CNN 网络并作为输入提供给 LSTM 网络，在相关论文中已经展示流卷积网络有益于识别活动类型。



这是第一次将通用的序列到序列模型应用到视频描述中，这使得该模型能够处理可变长度的输入帧，学习并使用视频的时序结构，以及通过学习语言模型来生成既符合语法规范又能自然表达视频内容的句子。该模型同时包含对帧图像输入和光学流图像输入的处理，且不需要精准的注意力模型。我们在标准的 YouTube 语料库(MSVD)、M-VAD 以及 MPII 电影描述数据集这三个不同数据集上进行实验，都具有比其他相关方法更好的性能。模型基于 Caffe 这个深度学习框架实现，代码可以在 Github 上找到。

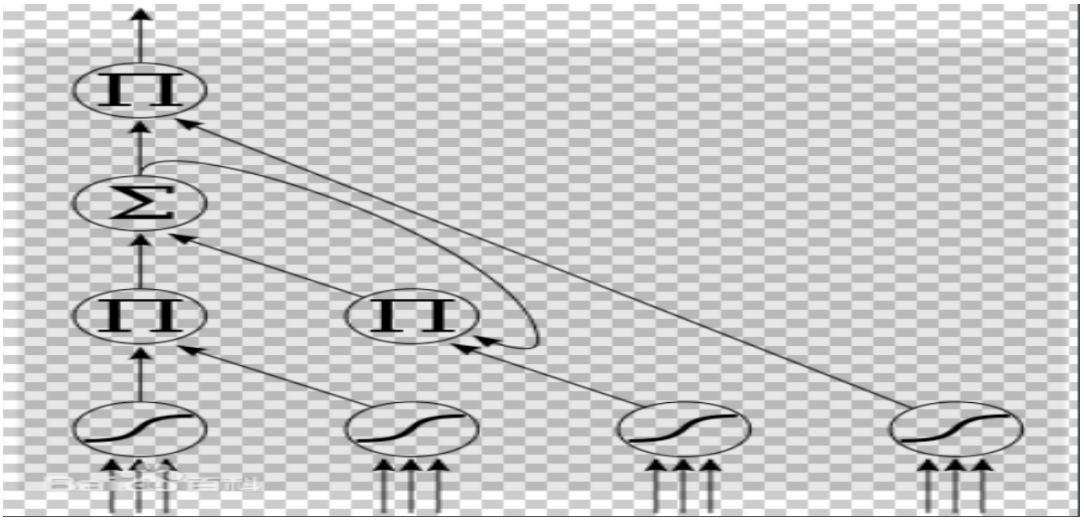
3.2 LSTM 网络

本部分针对模型中的 LSTM 网络进行简介，长短期记忆(Long short-term memory, LSTM)是一种特殊的 RNN，主要是为了解决长序列训练过程中的梯度消失和梯度爆炸问题。简单来说，就是相比普通的 RNN，LSTM
能够在更长的序列中有更好的表现。

LSTM 是一种含有 LSTM 区块（blocks）或其他的一种类神经网络，文献或其他资料中 LSTM 区块可能被描述成智能网络单元，因为它可以记忆不定时间长度的数值，区块中有一个 gate 能够决定 input 是否重要到能被记住及能不能被输出 output。

下图是四个 S 函数单元，最左边函数依情况可能成为区块的 input，右边三个会经过 gate 决定 input 是否能传入区块，左边第二个为 input gate，如果这里产出近似于零，将把这里的值挡住，不会进到下一层。左边第三个是 forget gate，当这产生值近似于零，将把区块里记住的值忘掉。第四个也就是最右边的 input 为 output gate，他可以决定在区块

记忆中的 input 是否能输出 。



4 词向量提取

4.1 分词统计

对视频描述的文字进行格式化的数据格式转划；

- 首先对英文描述进行分词，将完整的句子拆分成单独的英文单词；
- 在句子的开始添加“<sos>“标记，表明句子的开始；
- 在句子结束位置添加“<eos>“标记，表明句子结束；
- 对所有英文单词进行词频统计，设置词频阈值，单词词频低于该阈值时，将该单词舍弃，并且以”<UNK>”代替；本次实验的词频阈值为 1.

```python
def build_vocab(vids, params):
    count_thr = params['word_count_threshold']
    # count up the number of words
    counts = {}
    for vid, caps in vids.items():
        for cap in caps['captions']:
            ws = re.sub(r'[.!,;?]', ' ', cap).split()
            for w in ws:
                counts[w] = counts.get(w, 0) + 1
    # cw = sorted([(count, w) for w, count in counts.items()], reverse=True)
    total_words = sum(counts.values())
    bad_words = [w for w, n in counts.items() if n <= count_thr]
    vocab = [w for w, n in counts.items() if n > count_thr]
    bad_count = sum(counts[w] for w in bad_words)
    print('number of bad words: %d/%d = %.2f%%' %
          (len(bad_words), len(counts), len(bad_words) * 100.0 / len(counts)))
    print('number of words in vocab would be %d' % (len(vocab), ))
    print('number of UNKs: %d/%d = %.2f%%' %
          (bad_count, total_words, bad_count * 100.0 / total_words))
    # lets now produce the final annotations
    if bad_count > 0:
        # additional special UNK token we will use below to map infrequent words to
        print('inserting the special UNK token')
        vocab.append('<UNK>')
    for vid, caps in vids.items():
        caps = caps['captions']
        vids[vid]['final_captions'] = []
        for cap in caps:
            ws = re.sub(r'[.!,;?]', ' ', cap).split()
            caption = [
                '<sos>'] + [w if counts.get(w, 0) > count_thr else '<UNK>' for w in ws] + ['<eos>']
            vids[vid]['final_captions'].append(caption)
    return vocab
```

分词统计部分代码

```
number of bad words: 2613/6996 = 37.35%
number of words in vocab would be 4383
number of UNKs: 2613/65651 = 3.98%
inserting the special UNK token
```

分词统计结果

- 最终得到的可用单词数量为 4383 个；
- 所有句子中的单词有 3.98%的坏词；
- 其中坏词，即出现次数低于阈值的词汇，占单词比重达到 37.35%，因为本次数据集存在质量问题，所以产生了坏词占比大的问题。


```
{
    "G_12500": {
        "captions": ["Nearly 1000 new crown cases reported in California kindergartens",
            "The epidemic situation in the United States is serious",
            "The traffic police are directing the traffic.",
            "New coronavirus diagnosis increases",
            "The epidemic is getting worse in the United States"],
            "final_captions": [["<sos>", "Nearly", "1000", "new", "crown", "cases", "reported", "in", "California", "kindergartens", "<eos>"],
                               ["<sos>", "The", "epidemic", "situation", "in", "the", "United", "States", "is", "serious", "<eos>"],
                               ["<sos>", "The", "traffic", "police", "are", "directing", "the", "traffic", "<eos>"],
                               ["<sos>", "New", "coronavirus", "diagnosis", "increases", "<eos>"],
                               ["<sos>", "The", "epidemic", "is", "getting", "worse", "in", "the", "United", "States", "<eos>"]
                               ]},
```


分词后的英文描述

4.2 词序列化

- 将所有单词进行索引操作，为每个单词建立一个索引，实现单词文本到数字序列的映射；
- 建立”ix_to_word”对象，实现由索引提取单词的操作；


```
"32": "National",
"33": "Bureau",
"34": "of",
"35": "Statistics",
"36": "made",
"37": "statement",
"38": "Firefighter",
"39": "picks",
```


部分单词与索引关系

5 视频特征提取

5.1 图像特征提取

本模型，将视频分解为帧序列，并对帧画面进行图像信息的提取；

本实验采用与训练的 ResNet152 网络对视频画面进行图像特征提取；该模型时基于 Image-Net 进行训练，网络层数为 152 层；网络对每一个输入的图片输出一个 2048 维的特征向量。

在此基础上，对视频画面随机挑选 40 帧，对 40 帧画面单独输入网络，得到画面的图像特征，最终对每一个视频，得到一个尺寸为［40, 2048］的视频图像特征张量；

```python
def extract_frames(video, dst):
    with open(os.devnull, "w") as ffmpeg_log:
        if os.path.exists(dst):
            print(" cleanup: " + dst + "/")
            shutil.rmtree(dst)
        os.makedirs(dst)
        video_to_frames_command = ["ffmpeg",
                                   # (optional) overwrite output file if it exists
                                   '-y',
                                   '-i', video,  # input file
                                   '-vf', "scale=400:300",  # input file
                                   '-qscale:v', "2",  # quality for JPEG
                                   '{0}/%06d.jpg'.format(dst)]

        subprocess.call(video_to_frames_command,
                        stdout=ffmpeg_log, stderr=ffmpeg_log)
```

提取帧序列

```python
def extract_feats(params, model, load_image_fn):
    global C, H, W
    model.eval()

    dir_fc = params['output_dir']
    if not os.path.isdir(dir_fc):
        os.mkdir(dir_fc)
    print("save video feats to %s" % (dir_fc))
    video_list = glob.glob(os.path.join(params['video_path'], '*.avi'))
    for video in tqdm(video_list):
        #video_id = video.split("/")[-1].split(".")[0]
        video_id = video.split("_")[-1].split(".")[0]
        dst = params['model'] + '_' + video_id
        extract_frames(video, dst)

        image_list = sorted(glob.glob(os.path.join(dst, '*.jpg')))
        samples = np.round(np.linspace(
            0, len(image_list) - 1, params['n_frame_steps']))
        image_list = [image_list[int(sample)] for sample in samples]
        images = torch.zeros((len(image_list), C, H, W))
        for iImg in range(len(image_list)):
            img = load_image_fn(image_list[iImg])
            images[iImg] = img
        with torch.no_grad():
            fc_feats = model(images.cuda()).squeeze()
        img_feats = fc_feats.cpu().numpy()
        # Save the inception features
        outfile = os.path.join(dir_fc, 'video_' + video_id + '.npy')
        np.save(outfile, img_feats)
        # cleanup
        shutil.rmtree(dst)
```

提取图像特征

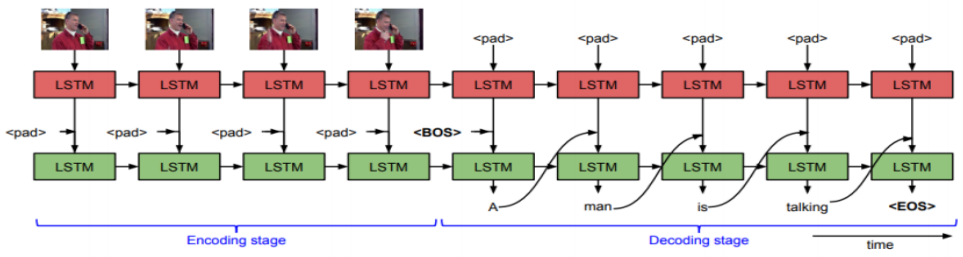
```
save video feats to data/feats/resnet152
100%|██████████| 2400/2400 [2:21:01<00:00,  3.53s/it]
```

特征提取完毕

最终提取得到 2400 个视频的特征向量，并用.npy 格式保存。

6 模型结构

本次试验中参照使用 S2VT 模型，即双层的循环神经网络结构，未使用 Attention 机制；而网络中的循环神经网络单元使用了 GRU 与 LSTM 两种类型；



模型结构

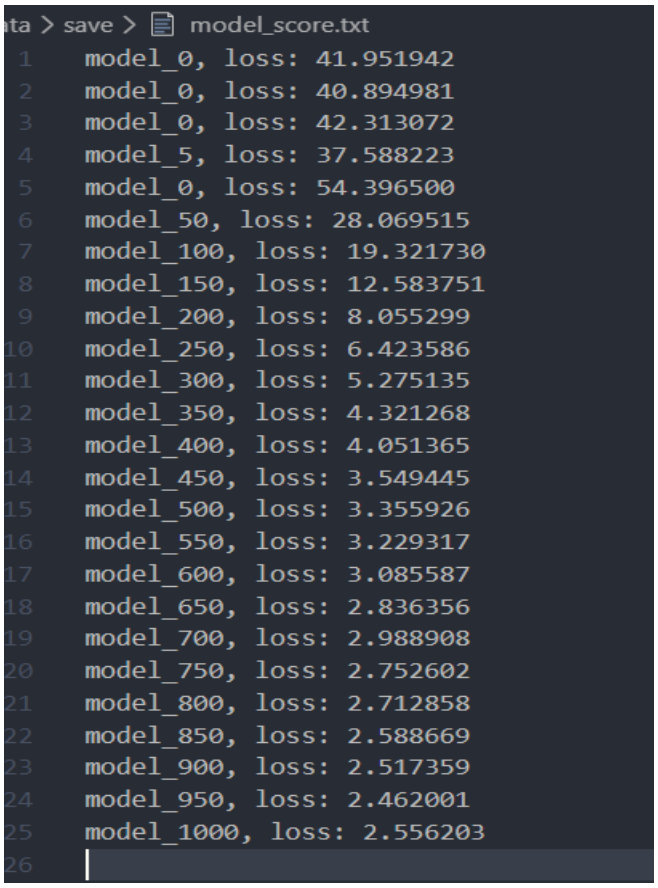
模型使用两层 RNN，每个具有 512 个隐藏单元。第一个红色的 LSTM 层用于帧序列进行家建模，输出隐藏状态作为第二层绿色 LSTM 层的输入用于对最终的输出子序列进行建模。

7 模型训练

设定如下的训练参数：

参数名	参数取值	参数描述
dim_vid:	2560 & 2048	输入特征维数
dim_hidden	512	隐藏层数量
batch_size	64	输入数据数量
epochs	1000	训练迭代次数
rnn_dropout	0.5	对 RNN 模型的 dropout；随机去除权重以增强模型泛化性能
learning_rate	4e-4	学习速率
rnn_type	GRU & LSTM	RNN 类型
with_c3d	1 & -1	是否适用 3D 特征

实验分别训练了多个模型，对 GRU 以及 LSTM 两种 RNN 类型都进行了训练；也对是否使用 3D 特征的情况进行分别训练；使用 3D 特征时数位维数为 2560，不使用时为 2048；以此比较模型在不同状态下的表现；



训练损失变化

8 模型评估

采用 score 中得分最高的模型，即 loss 最小的模型进行测试集的测试，本实验中为 model_950，最终得到一个 answer.json 的结果，提交到 codalab 平台，最终得分 0.15643。排名第二。

9 实验总结

通过本次实验，初步了解了 S2VT 模型的工作流程，也对内部代码逻辑进行了初步了解，能够理清模型结构，模型工作原理，但是最终的测试结果并没有很好，一方面是本次实验的数据集存在质量问题，另一方面，本实验可以适当调整别的参数，也许也能产生更好的结果。本次实验相对来说还是比较成功的。

（三）CycleGAN 风格迁移

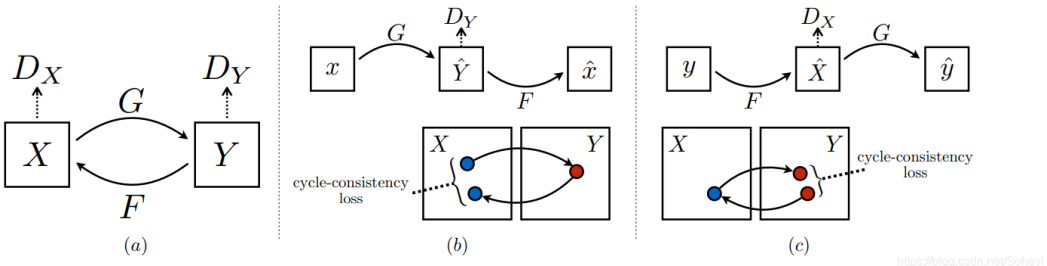
1 CycleGAN 简介

CycleGAN 是发表于 ICCV17 的一篇 GAN 工作，可以让两个 domain 的图片互相转化。传统的 GAN 是单向生成，而 CycleGAN 是互相生成，网络是个环形，所以命名为 Cycle。并且 CycleGAN 一个非常实用的地方就是输入的两张图片可以是任意的两张图片，也就是 unpaired。

CycleGAN 最经典的地方是设计和提出了循环一致性损失。以黑白图片上色为例，循环一致性就是：黑白图（真实）—>网络—>彩色图—>网络—>黑白图（造假）。为了保证上色后的彩色图片中具有原始黑白图片的所有内容信息，文章中将生成的彩色图像还原回去，生成造假的黑白图，通过损失函数来约束真实白图和造假黑白图一致，达到图像上色的目的。除此之外，CycleGAN 不像 Pix2Pix 一样，需要使用配对数据进行训练，CycleGAN 直接使用两个域图像进行训练，而不用建立每个样本和对方域之间的配对关系，一下子让风格迁移任务变得简单很多。

2 CycleGAN 原理

这是 CycleGAN 的网络结构图：



CycleGAN 其实就是一个 A→B 单向 GAN 加上一个 B→A 单向 GAN。两个 GAN 共享两个生成器，然后各自带一个判别器，所以加起来总共有两个判别器和两个生成器。一个单向 GAN 有两个 loss，而 CycleGAN 加起来总共有四个 loss。其实理解了单向 GAN 那么 CycleGAN 已经很好理解。

X→Y 的判别器损失为，字母换了一下，和上面的单向 GAN 是一样的：

$$L_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{data}(x)}[\log (1 - D_Y(G(x)))]$$

同理 Y→X 的判别器损失为

$$L_{GAN}(F, D_X, Y, X) = \mathbb{E}_{x \sim p_{data}(x)}[\log D_X(x)] + \mathbb{E}_{y \sim p_{data}(y)}[\log (1 - D_X(F(y)))]$$

而两个生成器的 loss 加起来表示为：

$$L_{cyc}(G, F) = \mathbb{E}_{x \sim p_{data}(x)}[||F(G(x)) - x||_1] + \mathbb{E}_{y \sim p_{data}(y)}[||G(F(y)) - y||_1]$$

最终网络的所有损失加起来为：

$$L(G, F, D_X, D_Y) = L_{GAN}(G, D_Y, X, Y) + L_{GAN}(F, D_X, Y, X) + L_{cyc}(G, F)$$

3 实验过程

本实验项目源码来自 https://github.com/junyanz/CycleGAN 的 CycleGAN。

3.1 安装

从安装 torch 和相应依赖，安装 nngraph，class，display 等 torch 包；

```
luarocks install nngraph
luarocks install class
luarocks install https://raw.githubusercontent.com/szym/display/master/display-scm-0.rockspec
```

复制 repo：

```
git clone https://github.com/junyanz/CycleGAN
cd CycleGAN
```

3.2 尝试使用预训练模型

下载测试图片：

```
bash ./datasets/download_dataset.sh ae_photos
```

下载预训练的 style_cezanne 模型：

```
bash ./pretrained_models/download_model.sh style_cezanne
```

将生成 cezanne 风格的图片：

```
DATA_ROOT=./datasets/ae_photos name=style_cezanne_pretrained model=one_direction_test
```

```
loadSize=256 fineSize=256 resize_or_crop="scale_width" th test.lua
```

3.3 结果

4 结果展示

使用自己训练的模型 summer2winter 模型

下载 summer2winter 的数据集：

```
bash ./datasets/download_dataset.sh winter2summer
```

训练一个模型：

```
DATA_ROOT=./datasets/summer2winter name=summer2winter_model th train.lua
```

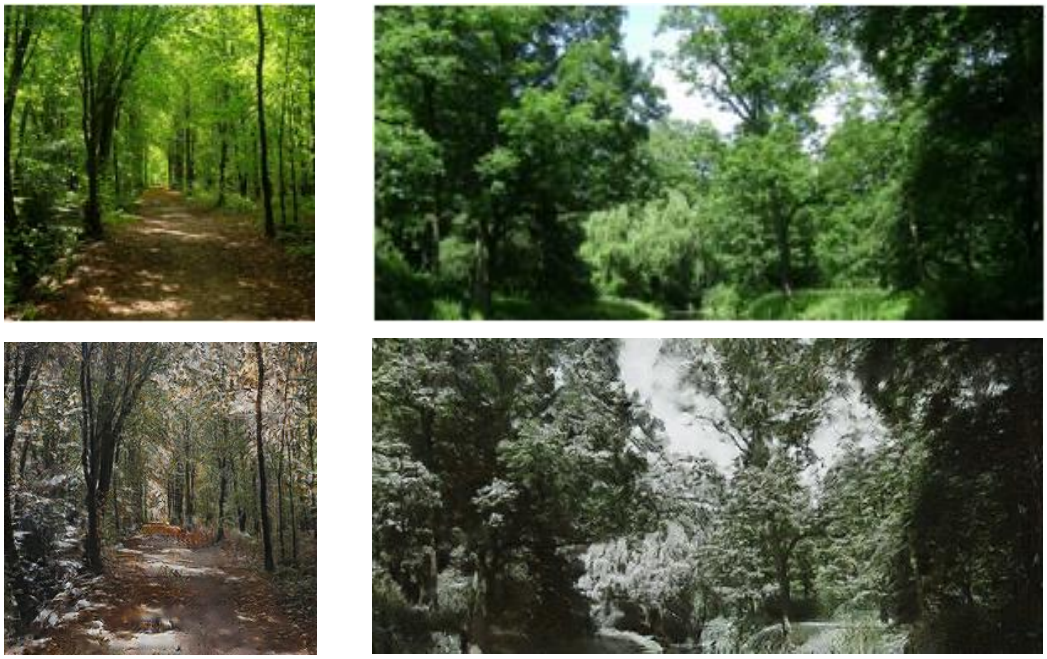
查看模型训练进展：

```
th -ldisplay.start 8000 0.0.0.0
```

测试模型结果：

```
DATA_ROOT=./datasets/summer2winter name=summer2winter_model phase=test th test.lua
```

结果：



5 实验总结

本次 CycleGAN 实验我们主要完成的是对于图像风格的一个转换，由于网上大多数的资料都是由苹果变为橘子，在我们看来风格并没有产生很大的一个变迁，所以我们这次选择了将图片转化为油画的这样一种风格上的彻底的转化来讲作为我们这一次实验的内容。

实现图像风格转换，通常需要一个包含成对图片的训练集，比如 Pix to Pix 方法的关键是提供了在这两个域中有相同数据的训练样本。CycleGAN 打破了这个限制，CycleGAN 能够在源域和目标域之间，无须建立训练数据间一对一的映射，实现图像风格的转换。

但 CycleGAN 也仍有一些问题，CycleGAN 有如下几个缺点：1.会在改变物体的同时改变背景；2.缺少多样性；生成的图片的指定特征只有一种；Source domain 和 target domain 的维度应该是不一样的；

（四）课堂展示内容（语音识别）

1 领域简介

语音识别技术就是让机器通过识别和理解过程把语音信号转变为相应的文本或命令的技术，也就是让机器听懂人类的语音，属于自然语言处理方向的一个部分。

技术难点：

内容的有效界定

日常生活中句子间的词汇通常是不会孤立存在的，需要将话语中的所有词语进行相互关联才能够表达出相应的含义，一旦形成特定的句子，词语间就会形成相应的界定关系。如果缺少有效的界定，内容就会变得模棱两可，无法进行有效的理解。例如他背着母亲和姐姐悄悄的出去玩了。这句话中如果不对介词“和”作出界定，就很容易形成母亲和姐姐两个人不知道他出去玩，或者是母亲不知道他和姐姐出去玩。

消歧和模糊性

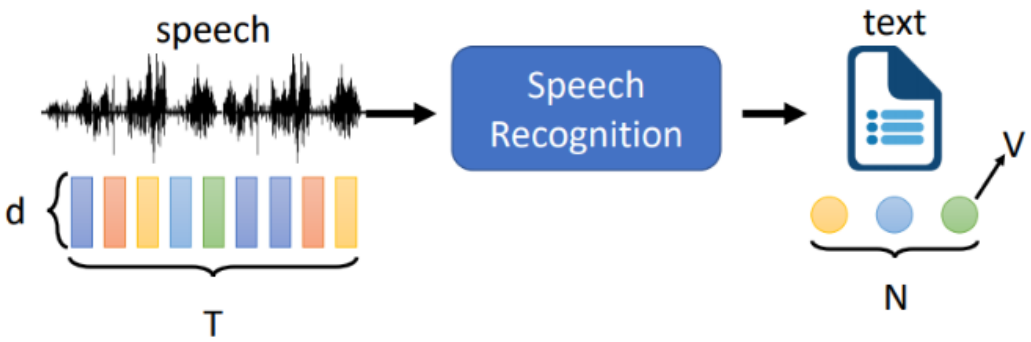
词语和句子在不同情况下的运用往往具备多个含义，很容易产生模糊的概念或者是不同的想法，例如高山流水这个词具备多重含义，既可以表示自然环境，也能表达两者间的关系，甚至是形容乐曲的美妙，所以自然语言处理需要根据前后的内容进行界定，从中消除歧义和模糊性，表达出真正的意义。

有瑕疵的或不规范的输入

例如语音处理时遇到外国口音或地方口音，或者在文本的处理中处理拼写，语法或者光学字符识别（OCR）的错误。

2 问题描述

一个典型的语音识别系统如下，输入一段语音到模型，模型输出一段文本



Speech：表示一个长度为 T，维度为 d 的向量序列

Text：一个 token 序列，长度为 N，V 个不同的 token，通常 T>N

输出 Token：

- 音位（phoneme，发音的基本单位）

在深度学习没有流行之前，以音位为输出是很常见的，因为音位和声音的对应关系比较强，那输出是一系列音位，怎么变成我们能看懂的文字呢？需要一个词典，需要语言学家标出来，音位同样也需要语言学家帮忙。

- 字母（Grapheme，书写的基本单位）

    1. 英文（基本书写单位：字母）

        总的 token：26 个英文字母+一个空格+标点符号

    2. 中文（基本书写单位：单个汉字）

        总的 token：常用的汉字（和英文区别在于没有空格）

- 词（word）

英文：one punch man；N=3，通常 V>100K

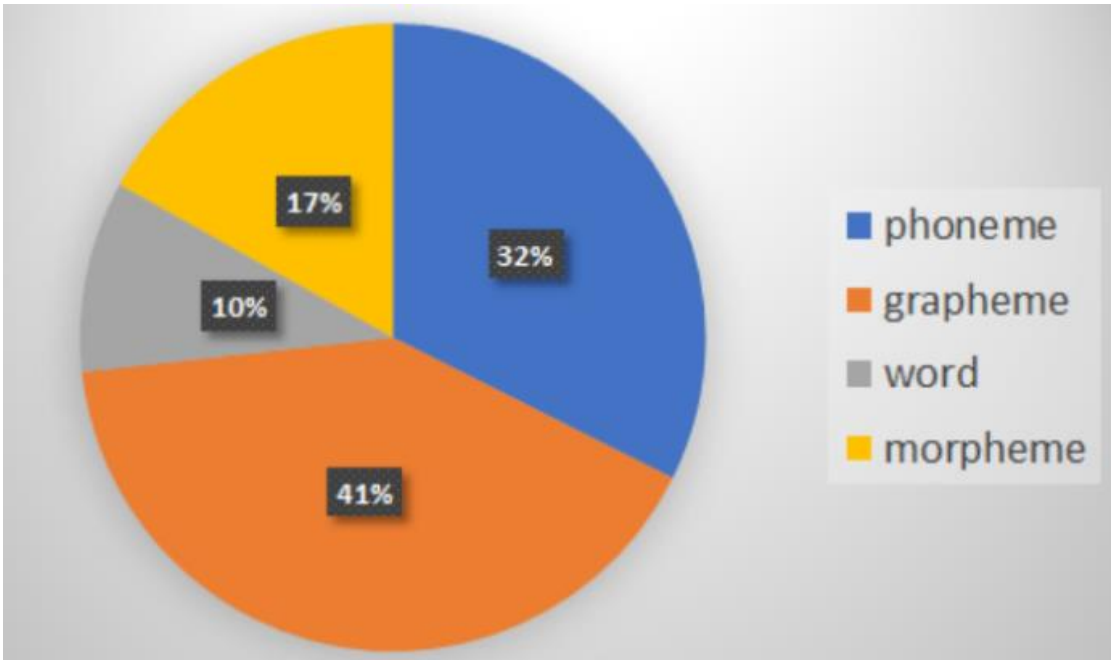
中文：“一拳 ”超人“；N=2，V=?

使用词做为输出单位很难，因为中文没有空格，没有词的明确分界，对于一些语言，V 可能超大，无法穷举

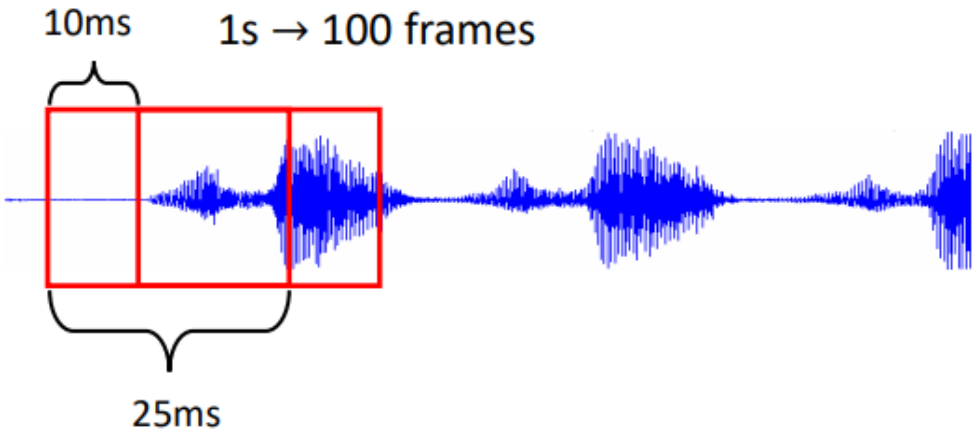
- 语素（Morpheme，可以传达意思的最小单位，小于词，大于字母）

例如英文中：unbreakable 可以拆成 “un“ ”break“ ”able”

语素如何获取？统计学方法

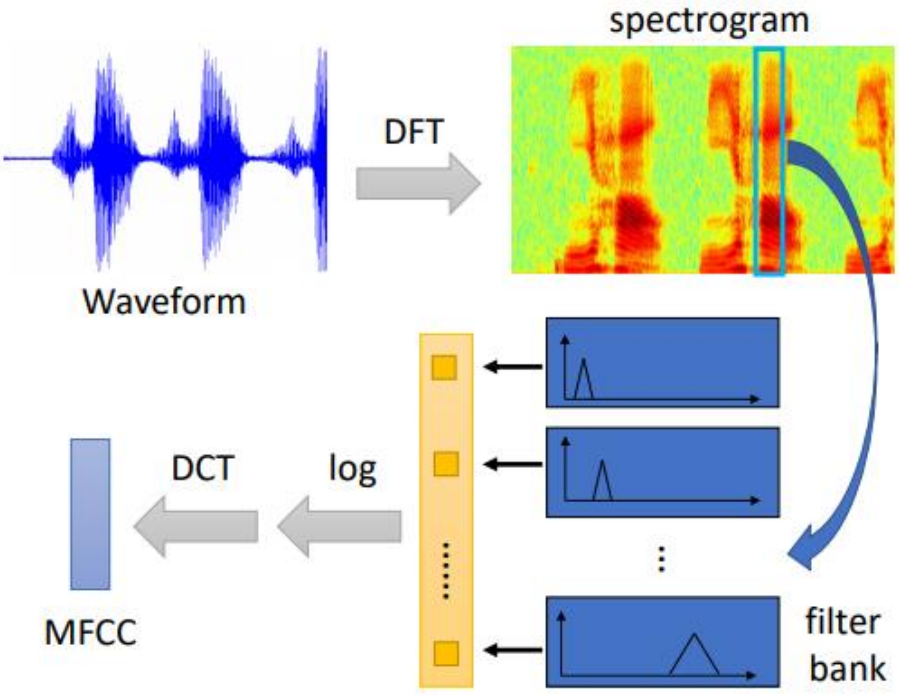


输入部分(声学特征，acoustic feature)：

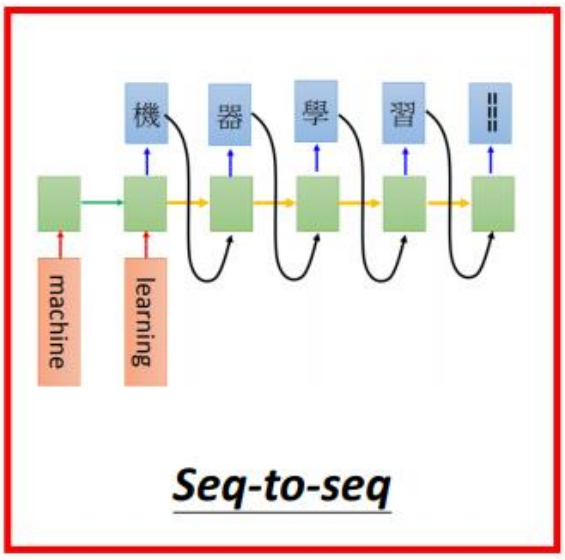


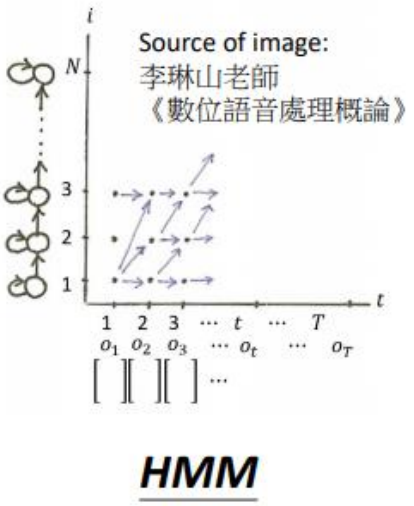
对输入的声音信号，使用 25ms 的时间窗取出一个 frame，对应就有 400 个采样点（16KHz）（使用 MFCC 会得到 39 维向量、filter bank 输出是 80 维），通常的每个时间窗的间隔为 10ms，那么 1s 内就有 100 个 frame

输入声音信号 经过 离散傅里叶变换变成频谱图，经过多个不同的 filter bank （古圣先贤们设计出来的）处理后，得到向量。使用对数变换，经过离散余弦变换，使用 MFCC 方法得到向量



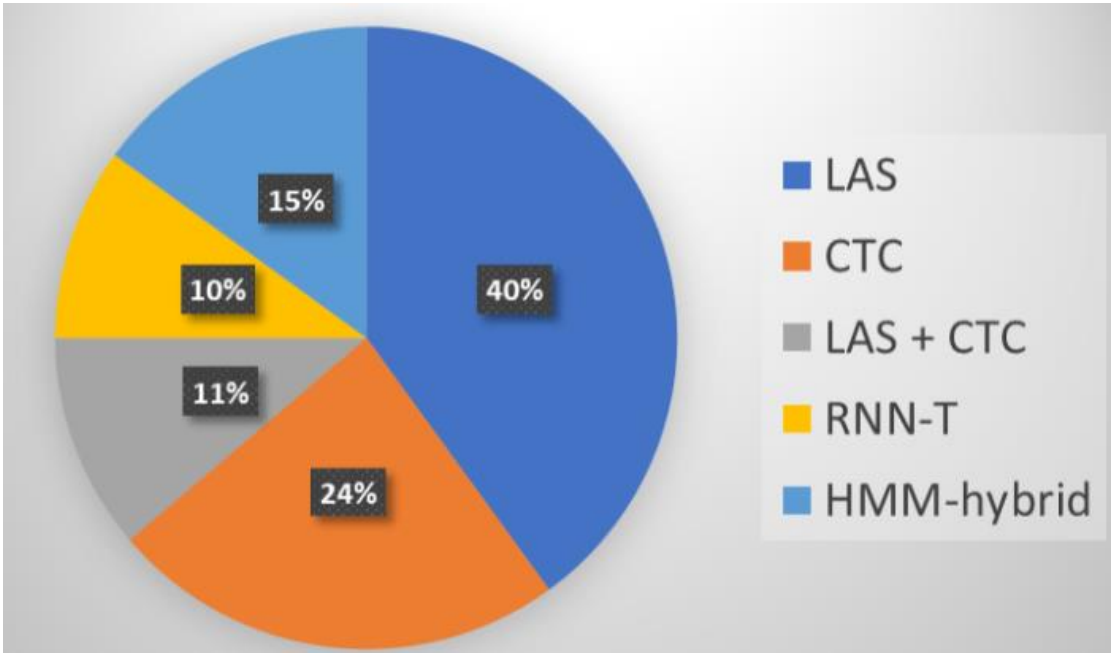
语音识别模型两个角度：





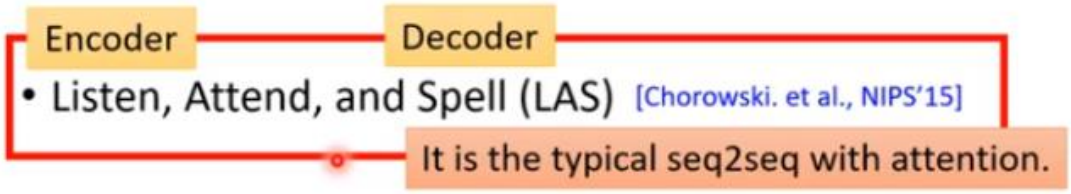
seq-to-seq 将要被介绍的模型

- Listen, Attend, and Spell (LAS)
- Connectionist Temporal Classification (CTC)
- RNN Transducer (RNN-T)
- Neural Transducer
- Monotonic Chunkwise Attention (MoChA)



3 模型介绍

3.1 LAS 模型



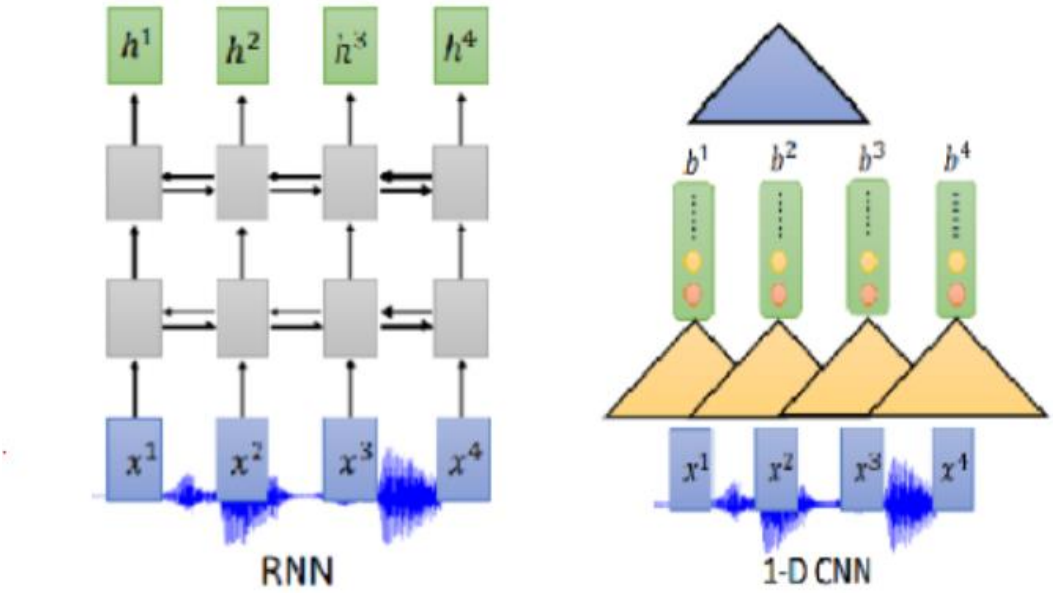
a) Listen

Listen 是一个典型的 Encoder 结构，输入为声学特征 $x_1, x_2, \ldots, x_T$，$x_1, x_2, \ldots, x_T$，输出和输入长度相同，是对声学特征的高阶表示，$h_1, h_2, \ldots, h_T$，$h_1, h_2, \ldots, h_T$.

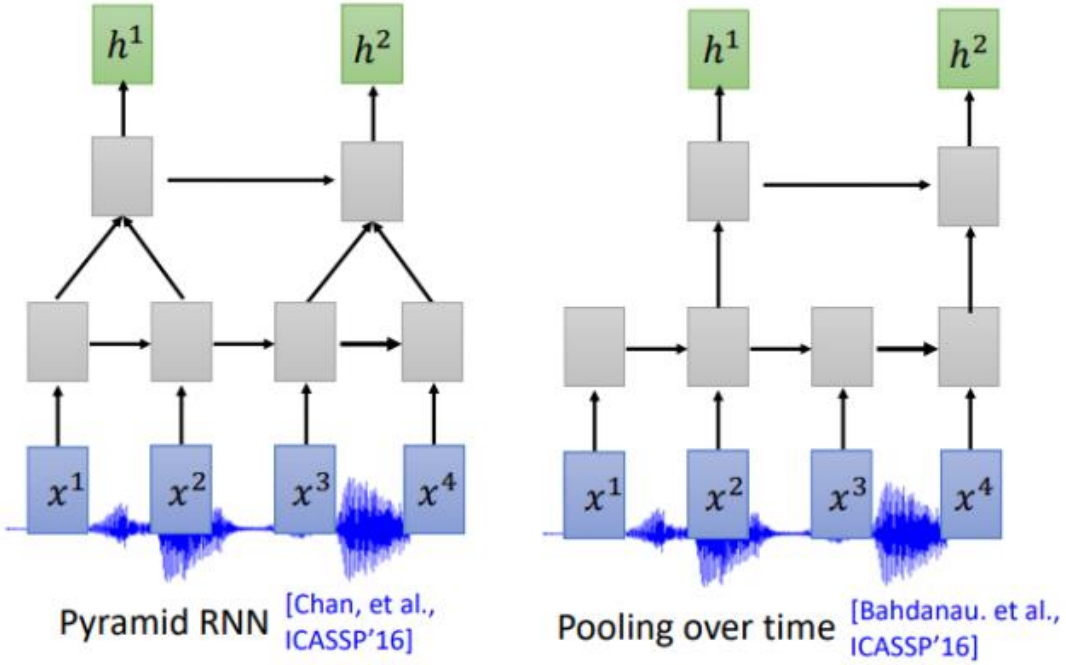
我们希望 Encoder 可以做到以下两件事：

- 提取输入的内容信息
- 移除不同说话者之间的差异，去掉噪音

那 Encoder 怎么做呢？可以用 RNN、CNN



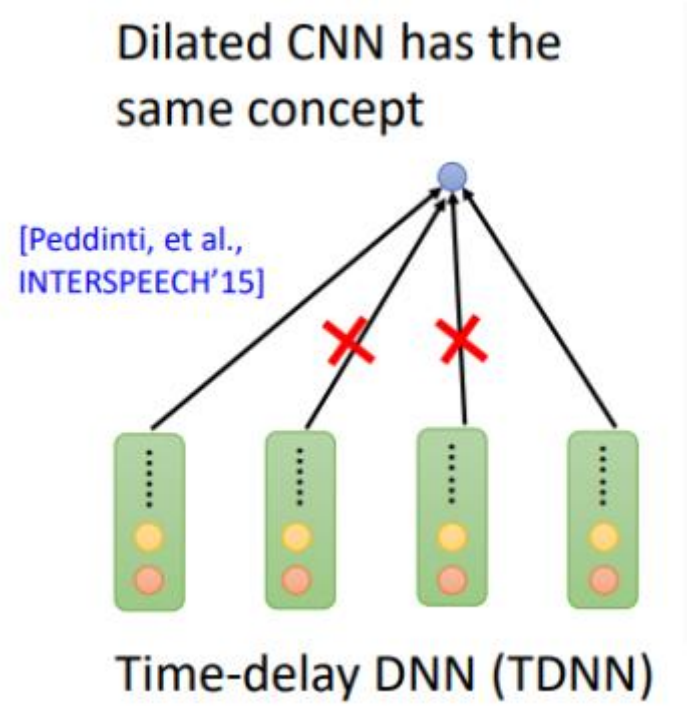
RNN



Pyramid RNN 将下层每两个隐状态加起来作为下一层

Pooling over time 和 Pyramid RNN 很像，不同没有加起来，直接每两个隐状态取一次作为下层输入。

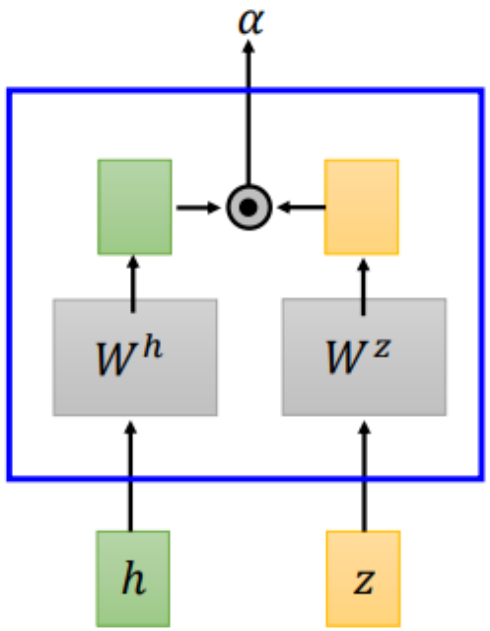
CNN



对于 CNN 常用的变形是 TDNN (Time-delay DNN)，不同于传统的 CNN 做卷积操作时会考虑范围内所有的输入，TDNN 相当于只让部分参与了运算，提高效率。

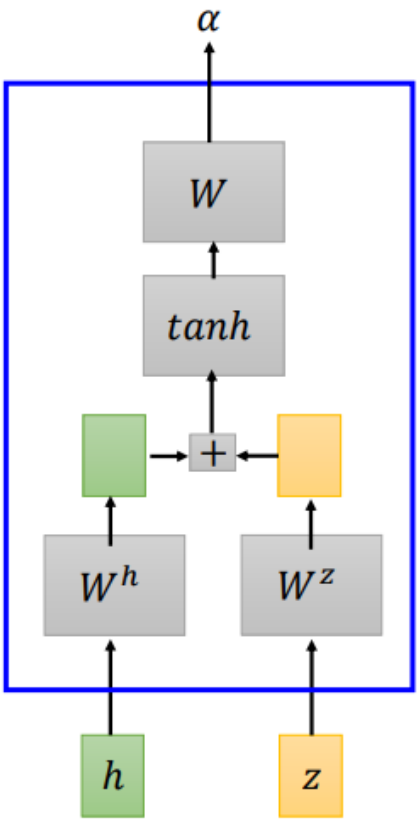
b) Attention

- dot-product attention



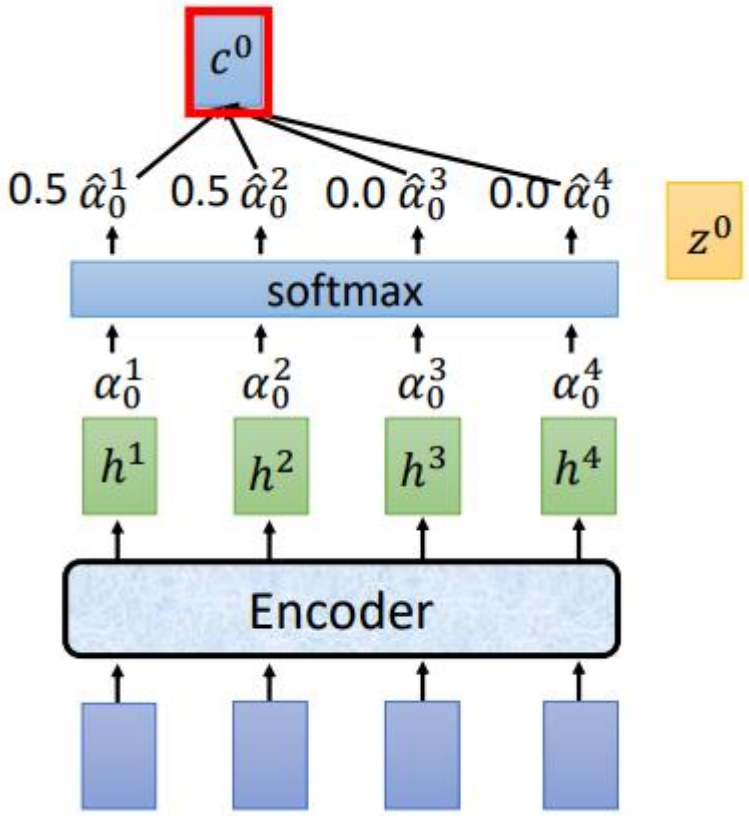
dot-product attention 将输入h和z经过矩阵W^h、W^z转换，将转换结果进行点积，得到α

- additive attention



additive attention 将输入h和z经过矩阵W^h、W^z转换，将转换结果相加，经过一个线性变换得到α

LAS 的 attention 具体做法：



将 z0z0 分别和 h1,h2,h3,h4h1,h2,h3,h4 做 attention 运算得到 α10,α20,α30,α40, α01,α02,α03,α04，经过 softmax 归一化，再将归一化后的结果和 hihi 相乘求和得到 c0c0，将 c0c0 作为 Decoder 部分的输入。

c) Spell

LAS 的 Listen 对应 Encoder，Spell 对应的就是 Decoder，假设 Encoder 的输入为“cat”，Decoder 的每一个时间步对应的输出就是词汇表中每个词的分布，通常会选概率最大的那个作为输出。

d)LAS 问题

LAS 采用经典的 Encoder 和 Decoder 架构，也就是说，只有在完整的听完一句话之后模型才会输出

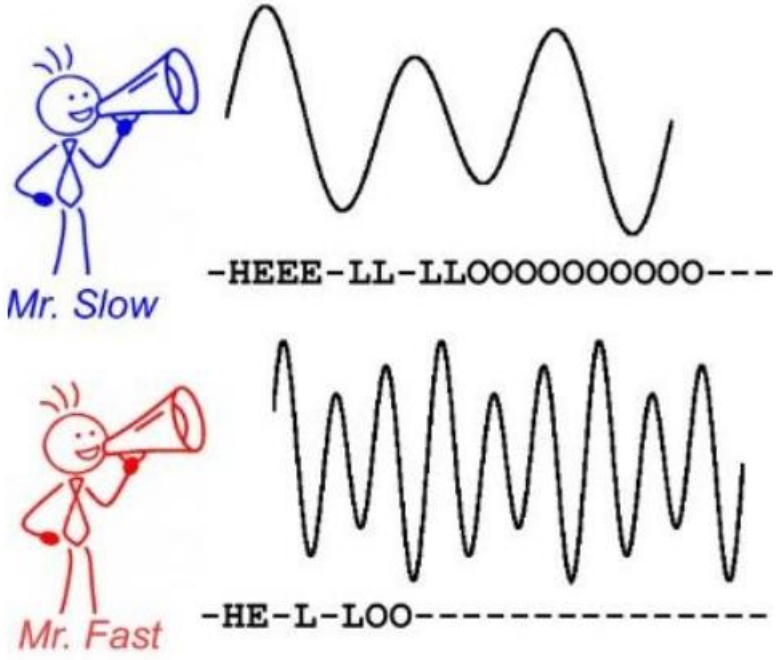
3.2 CTC 模型

a)CTC 应用范围

LAS 做语音识别需要看完一个完整的序列才能输出，但是在很多时候，例如实时语音或者使用翻译软件时，我们希望语音识别模型可以在听到声音的时候就进行输出，这个该怎么做呢？一个很直观的想法就是用单向的 RNN，下面我们来介绍一下 CTC 是怎么做的。

CTC ，Connectionist Temporal Classification 是用来解决输入序列和输出序列难以一一对应的问题的算法。举例来说，在语音识别中，我们希望音频中的音素和翻译后的字符可以一一对应，这是训练时

一个很天然的想法。但是要对齐是一件很困难的事，如下图所示：



有人说话快，有人说话慢，每个人说话快慢不同，不可能手动地对音素和字符对齐，这样太耗时。

b)CTC 具体算法思想

在线语音识别，模型在听到声音的时候就需要输出。

假设输入的语音为 $\vec{x} = x_1 x_2 \ldots x_T$,而对应的label为 $\vec{y} = y_1 y_2 \cdots y_L$ ，其输出空间为 Y ，对于ASR，通常 $L <= T$ ，为解决alignment问题，CTC规定了额外的空格标记blank symbol，假设用标记<b>来表示。我们利用 $B(\vec{y}, \vec{x})$ 来代表所有的长度为T的输出序列，且每一个元素均属于 $Y \cup <b>$ 的集合，并且经过把所有相连的重复字母合并且除去所有blank symbol的操作后与原label $\vec{y}$ 相同。

举个例子，假设我们输入的语音有八帧，并且其对应的文字是 cat 仅有三个字符，如何使输出与输入对应呢？

我们可以是 cc<b>aa<b>t<b>

也可以是 c<b><b>aa<b>t<b>

或者是 c<b>aaa<b>t<b>等等…

这些分割都可以经过上述操作对应到最终的文字输出 cat 上，这就省去了我们需要带有 alignment 的标记的需要。

c)CTC 存在问题

CTC 算法存在一个问题，对于一个真实的输出，<b>的位置有很多种情况，例如：

真实的输入输出：

input：x1,x2,x3,x4 ； ouput：好棒

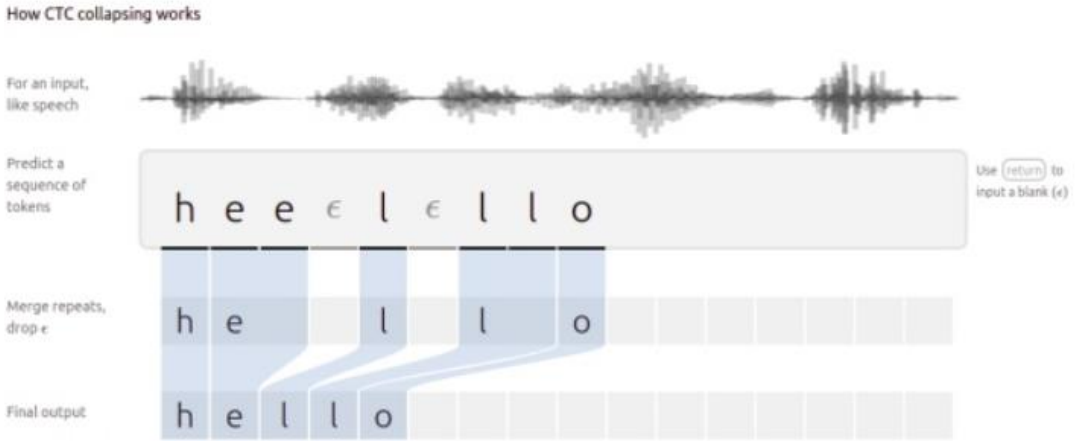
可能的输出：

好好棒<b>，<b>好好棒，好好棒棒 ，好棒<b><b>，好<b><b>棒……

所以 CTC 在训练的时候需要**穷举所有可能组合！**

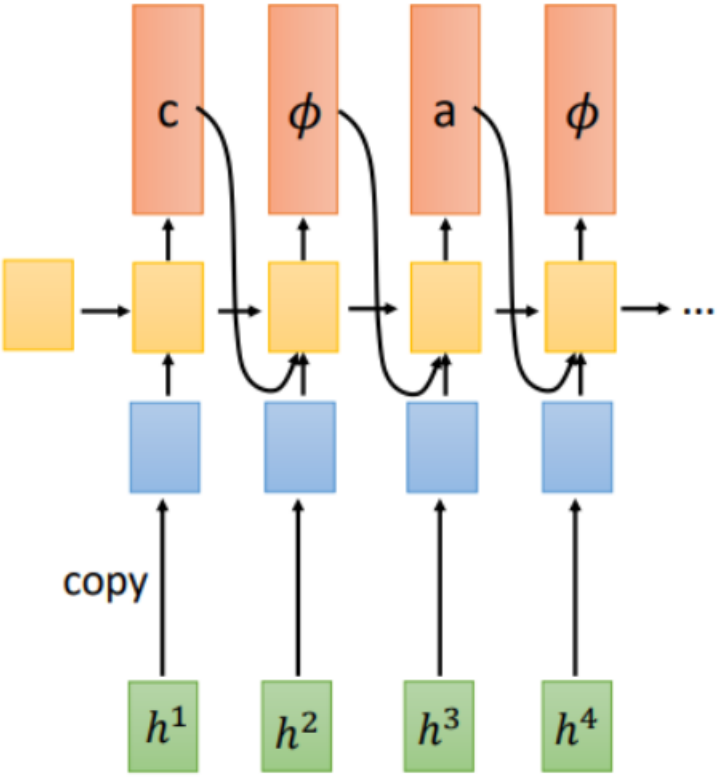
d)CTC 的特点

- 引入 blank 字符，解决有些位置没有字符的问题
- 通过递推，快速计算梯度



3.3 RNN-T 模型

在介绍 RNN-T 之前，我们先看下 RNA（Recurrent Neural Aligner）

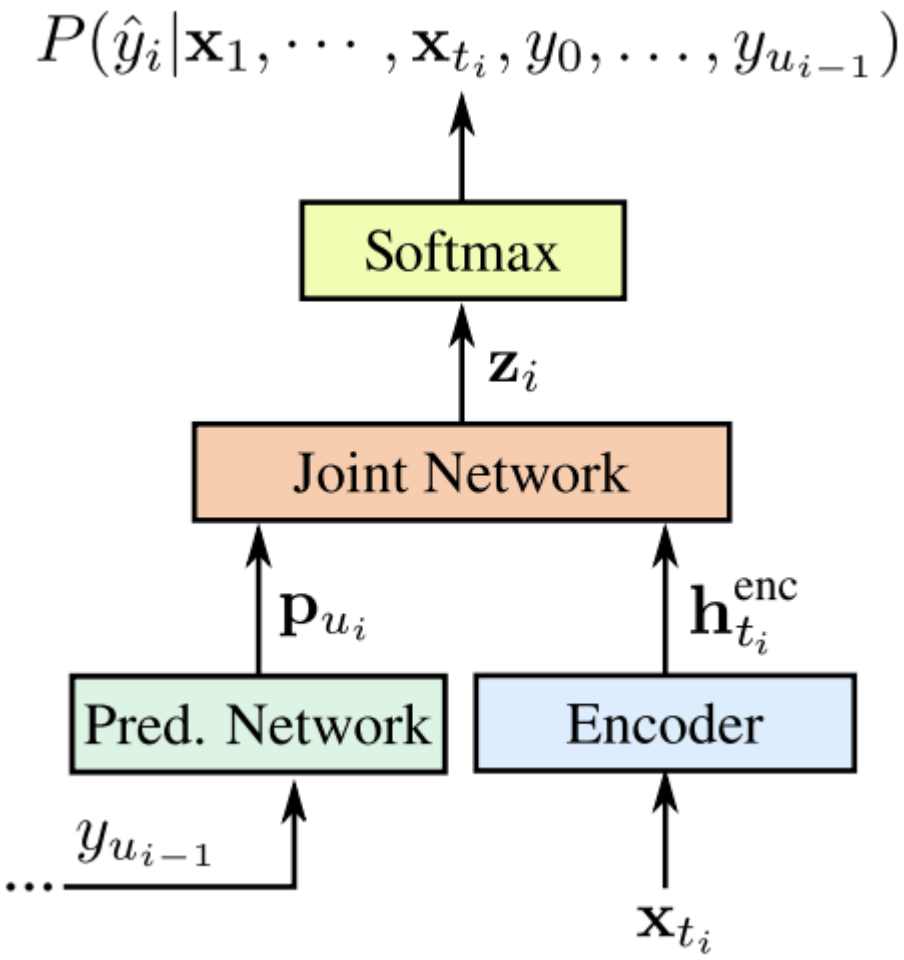


在从上面的图可以看到，RNA 就是增加了输出之间的依赖。

a)RNN-T 概念

RNN-T全称是Recurrent Neural Network Transducer，是在CTC的基础上改进的。CTC的缺点是它没有考虑输出之间的dependency，即 y_t 与之前帧的 y_j (j<t) 没有任何关联，而RNN-T则在CTC模型的Encoder基础上，又加入了将之前的输出作为输入的一个RNN，称为Prediction Network，再将其输出的隐藏向量 p_u 与encoder得到的 h^{enc} 放到一个joint network中，得到输出logit再将其传到softmax layer得到对应的class的概率。

整体模型结构如下:

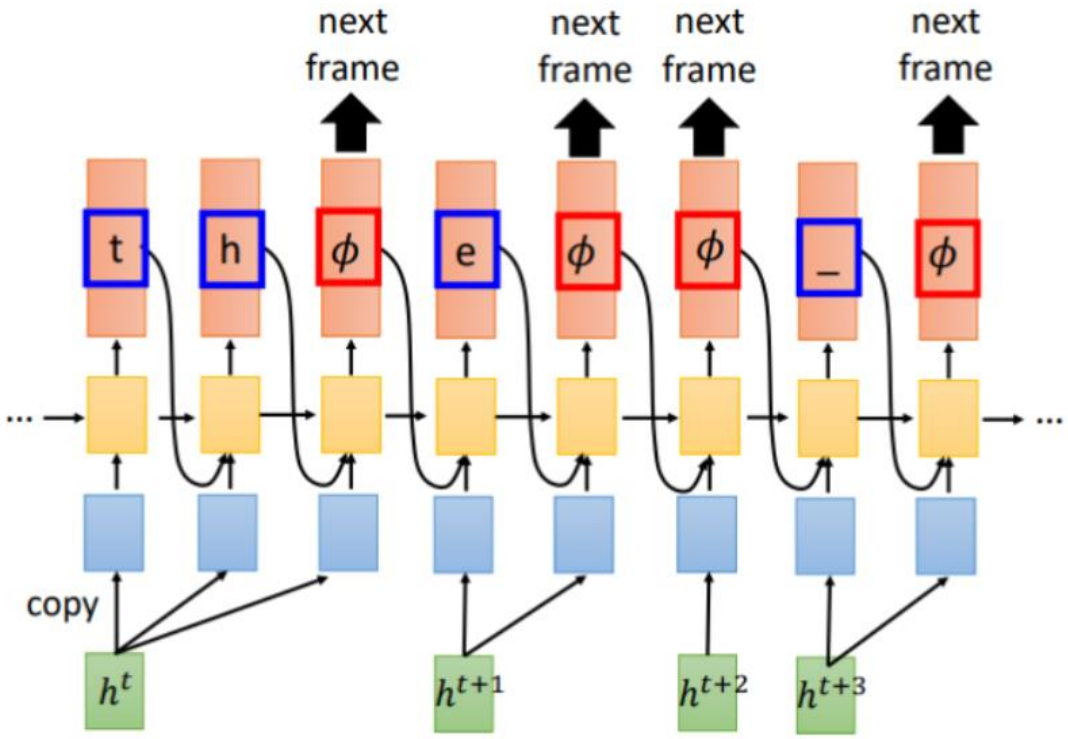


b)RNN-T 解决的问题

目前为止我们看到的都是吃一个输入，输出一个 token，那有没有可能吃一个输入，输出多个 token 呢？举例来说，输入为“th”，只有一个音，但是需要两个输出。当然也可以把“th”作为一个单独的 token 加入到词典

RNN-T 就是为了解决这个问题。

每次看到一个 h^t ，输出对应的 token，直到对应输入被解码完了，就输出一个 ϕ，告诉模型给我下一个输入，所以一共会有 T 个 ϕ 会被输出。



c)RNN-T 存在的不足

RNN-T 和 CTC 有同样的组合问题。

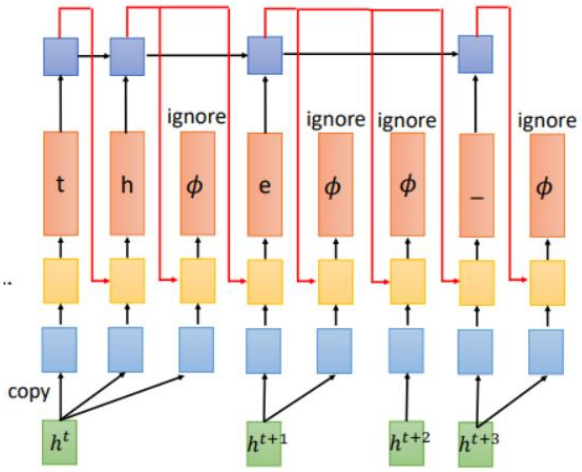
假设“好棒”有四个声学特征：

$\phi1$ 好 $\phi2$ $\phi3$ $\phi4$ $\phi5$ 棒 $\phi6$

$\phi1$ $\phi2$ $\phi3$ $\phi4$ $\phi5$ 好 棒 $\phi6$

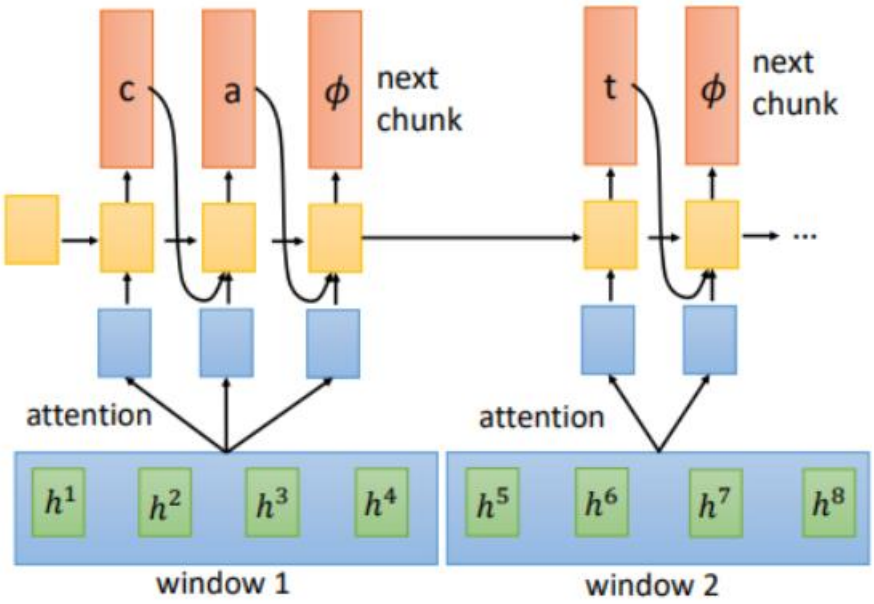
所以，RNN-T 同样会在训练时穷举所有的组合。

实际上 RNN-T 训练了一个独立的 RNN 网络，输出会被接到 RNN，ϕ 会被忽略，如下图：



4.4 Neural Transducer 模型

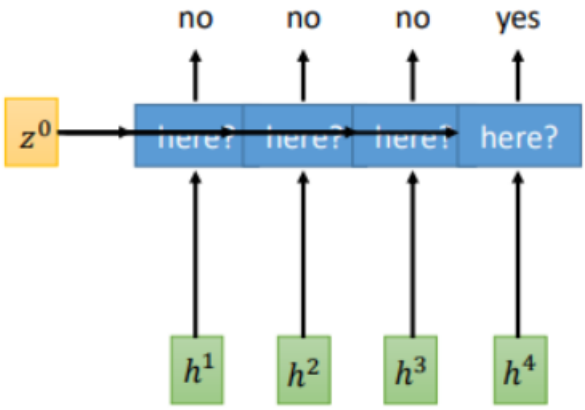
前面的几个模型每次都是输入一个，Neural Transducer 一次接受窗口范围的输入



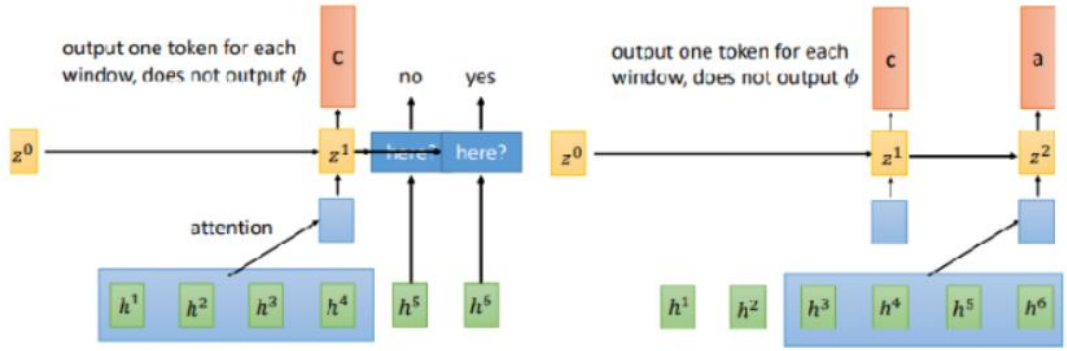
对 Encoder 的输出，每次取一个窗口作为 Decoder 的输入，先做一下 attention 然后开始输出，每次输出 ϕ 就会读下一个窗口。

5.5 Monotonic Chunkwise Attention 模型

在 Neural Transducer 每次窗口长度是固定的，在 MoChA 中设计了动态的窗口。



首先使用 attention 决定窗口应该放在哪里，然后对每一个窗口输出一个 token，注意这里不需要 ϕ 了，然后移动找下一个窗口。



4 总结

总结一下语音识别中用到的几个 seq2seq 模型

- LAS：就是 seq2seq
- CTC：decoder 是 linear classifier 的 seq2seq
- RNA：decoder 是 RNN 的 seq2seq，输入一个就要输出一个
- RNN-T: decoder 是 RNN ，输入一个就要输出多个的 seq2seq
- Neural Transducer：每次输入 一个 window 的 RNN-T
- MoCha： window 移动伸缩自如的 Neural Transducer

三、课程总结与反思

（一）课程任务完成情况

首先我们组内分工完成了视频描述的任务，为第二次的实验数据集打下基础。在视频描述竞赛中，我们的第一步工作是对数据集进行划分，采用 1920 条视频作为训练集，480 条视频作为测试集。第二步，我们选定了 S2VT 作为我们的视频标注模型，接下来的主要工作是词分量的提取和视频特征提取与处理，以及模型的训练。训练好我们的模型后，将实验结果提交到线上平台，分数比较低，但排名第二。

此外，我们还完成了 CycleGan 这个选做实验。CycleGAN 是发表于 ICCV17 的一篇 GAN 工作，可以让两个 domain 的图片互相转化。我们的实验主要是将实拍图进行了油画风格的转换，效果比较理想

在课堂展示环节，我们选择语音识别这个主题来向大家展示其中的原理。报告分为四个环节：领域介绍、问题描述、模型介绍与小结。

（二）课程任务难点与反思

视频标注竞赛是我们的主要工作，也是最难的一部分工作。虽然数据集和模型我们有了，但训练过程以及最后得分都有一些问题。在训练模型时，算法对机器的硬件条件有较高要求，比如显存、运算速度、散热等，这对我们训练模型有一定的阻碍。第二点，我们班级所用的视频描述训练集的质量非常不好，这是由同学们在描述视频时不同的表达习惯造成的，也是造成我们最后得分比较低的主要原因。

CycleGan 实验的问题： CycleGAN 有如下几个缺点：1.会在改变物体的同时改变背景；2.缺少多样性；生成的图片的指定特征只有一种；Source domain 和 target domain 的维度应该是不一样的。这些问题也有待我们以后再去解决。