2018 年秋-第 14 周

本周工作是把抽帧的报错解决了，开会之后转而用 thumos14 数据集并把 test 的帧抽出来了，并且弄明白了初始化 proposal 截止到周五中午 val 抽帧的程序还在服务器上运行，具体的 ssn 的结果下周一回学校提交。

下面是一些已经看了的工作，对于原始的视频数据要生成 proposal，作者在这里训练了一个二元动作分类器。

首先是生成一系列滑动窗口提议，程序为 gen_sliding_window_proposals.py，程序做了这几件事情：

把下面的内容全部读入了函数中，并存在相应的类，这些文件是跟随数据集被一同下载下来的，包括视频类别，动作开始时间，动作结束时间，视频总时长。

thumos_14	2018/11/30 10:43	文件夹	
activity_net.v1-2.min.json	2018/11/30 10:43	JSON 文件	4,738 KB
activity_net.v1-3.min.json	2018/11/30 10:43	JSON 文件	4,823 KB
activitynet1.2_tag_train_normalized...	2018/11/30 10:43	文本文档	8,028 KB
activitynet1.2_tag_val_normalized_...	2018/11/30 10:43	文本文档	3,860 KB
dataset_actionness_cfg.yaml	2018/11/30 10:43	YAML 文件	3 KB
dataset_cfg.yaml	2018/11/30 10:43	YAML 文件	3 KB
reference_models.yaml	2018/11/30 10:43	YAML 文件	2 KB
thumos14_tag_test_normalized_pro...	2018/11/30 10:43	文本文档	8,070 KB
thumos14_tag_val_normalized_pro...	2018/11/30 10:43	文本文档	6,687 KB
temporal_annotations_test	2018/11/30 10:43	文件夹	
temporal_annotations_validation	2018/11/30 10:43	文件夹	
test_avoid_videos.txt	2018/11/30 10:43	文本文档	1 KB
test_durations.txt	2018/11/30 10:43	文本文档	55 KB
th14_test_durations.txt	2018/11/30 10:43	文本文档	103 KB
th14_val_durations.txt	2018/11/30 10:43	文本文档	42 KB
validation_avoid_videos.txt	2018/11/30 10:43	文本文档	1 KB
validation_durations.txt	2018/11/30 10:43	文本文档	42 KB

建立了 `class THUMOSDB(object):`，里面有很多函数相互调用，最终的主要用于读取上述信息，同时在这之后把这些信息存入 video 类和 instance 类。

Video 类：

```
class Video(object):
    """
    This class represents one video in the activity-net db
    """
    def __init__(self, key, info, name_idx_mapping=None):
        self._id = key
        self._info_dict = info
        self._instances = [Instance(i, x, self._id, self._info_dict, name_idx_mapping)
                           for i, x in enumerate(self._info_dict['annotations'])]
        self._file_path = None
```

来看创建这个类的代码：

```
video_dict[v[0]] = Video(v[0], info_dict, name_idx_mapping)
```

Key：video 名

Info_dict 的结构：

```
info_dict = {
    'duration': float(v[1]),
    'subset': subset,
    'url': None,
    'annotations': [
        {'label': item[0], 'segment': (item[1], item[2])} for item in annotaion_table[v[0]] if i
    ]
}
```

Name_idx_mappping：排序后的 val 数据的文件夹名

这样基本就描述清楚了这个 video 类

下面是 instance 类，这个类的一个列表为 video 类的一个属性 Video._instances

```
class Instance(object):
    """
    Representing an instance of activity in the videos
    """

    def __init__(self, idx, anno, vid_id, vid_info, name_num_mapping):
        self._starting, self._ending = anno['segment'][0], anno['segment'][1]
        self._str_label = anno['label']
        self._total_duration = vid_info['duration']
        self._idx = idx
        self._vid_id = vid_id
        self._file_path = None

        if name_num_mapping:
            self._num_label = name_num_mapping[self._str_label]
```

下面代码描述了这个类是怎么生成的：

```
self._instances = [Instance(i, x, self._id, self._info_dict, name_idx_mapping)
                   for i, x in enumerate(self._info_dict['annotations'])]
```

关于生成 proposal：

```
def gen_exponential_sw_proposal(video_info, time_step=1, max_level=8, overlap=0.4):
    spans = [2 ** x for x in range(max_level)]
    duration = video_info.duration
    pr = []
    for t_span in spans:
        span = t_span * time_step
        step = int(np.ceil(span * (1 - overlap)))
        local_boxes = [(i, i + t_span) for i in np.arange(0, duration, step)]
        pr.extend(local_boxes)

    # fileter proposals
    # a valid proposal should have at least one second in the video
    def valid_proposal(duration, span):
        real_span = min(duration, span[1]) - span[0]
        return real_span >= 1

    pr = list(filter(lambda x: valid_proposal(duration, x), pr))
    return pr
```

Spans 从 2^0 到 2^8，生成不同起始位置和不同长度（起点减终点）的标记作为 proposal，然后对这些 proposal 的合法性进行判断，最终按照数据集给定样式输出 proposal 到 txt。