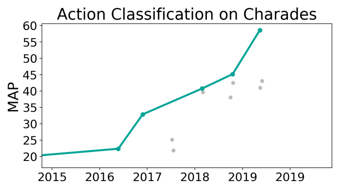
행동 인식 / 검출

Action Classification

Papers with Code - Charades Benchmark (Action Classification)

The current state-of-the-art on Charades is AssembleNet. See a full comparison of 14 papers with code.

https://paperswithcode.com/sota/action-classification-on-charades



성능비교

표.1

RANK	MODEL	MAP	YEAR
1	AssembleNet	58.6	2019
2	AssembleNet-101	58.6	2019
3	SlowFast	45.2	2018
4	HAF+BoW/FV/OFF halluc. +MSK×8/PN	43.1	2019
5	LFB	42.5	2018
6	PA3D +(GCN + I3D + NL I3D)	41	2019
7	PoTion +(GCN + I3D + NL I3D)	40.8	2018
8	STRG	39.7	2018
9	EvaNet	38.1	2018
10	I3D	32.9	2017
11	MultiScale TRN	25.2	2017
12	Asyn-TF	22.4	2016
13	CoViAR	21.9	2017
14	2-Strm	18.6	2014

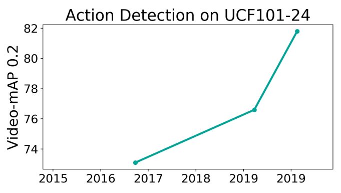
- AssembleNet 는 가장 성능이 높지만 공개된 소스코드는 없음
- SlowFast, LFB는 소스코드에 학습하는 부분도 제공함

Action Detection

Papers with Code - UCF101-24 Benchmark (Action Detection)

The current state-of-the-art on UCF101-24 is MOC. See a full comparison of 5 papers with code.

https://paperswithcode.com/sota/action-detection-on-ucf101-24



성능비교

표.2

# RANK	Aa MODEL	# MAP	# VIDEO-MAP 0.5	# VIDEO-MAP 0.75	# VIDEO-MAP 0.2	# VIDEO-MAP 0.1	# YEAR
1	MOC	0	0	0	81.8	0	2020
2	STEP	0	0	0	76.6	83.1	2019
3	T-CNN	0	0	0	73.1	77.9	2017
4	Ours(MOC)	27.7	53.9	28.5	0	0	2020
5	two-in-one two stream	22.02	0	0	0	0	2019

- 최고 성능 모델 : MO

  표.3

Aa Metric Name	Value	Global Rank
Video-mAP 0.5	53.9	# 1
Video-mAP 0.75	28.5	# 1
Video-mAP 0.2	81.8	# 1

결론

- 성능과 빠른 테스트를 위해서 MOC > LFB > SlowFast 순서로 진행
- MOC GitHub

  MCG-NJU/MOC-Detector

  Pytorch implementation of Actions as Moving Points (ECCV 2020). View each action instance as a trajectory of moving points. Visualization results on validation set. (GIFs will take a

  https://github.com/MCG-NJU/MOC-Detector

- LFB GitHub

wei-tim/YOWO

PyTorch implementation of the article " You Only Watch Once: A Unified CNN Architecture for Real-Time Spatiotemporal Action Localization". In this work, we present

https://github.com/wei-tim/YOWO

- SlowFast GitHub

facebookresearch/SlowFast

PySlowFast is an open source video understanding codebase from FAIR that provides state-of-the-art video classification models with efficient training. This repository includes

https://github.com/facebookresearch/SlowFast

[SlowFast]

SlowFast

1. DataSets

1.1 Train

ex. Kinetics

- 동영상 파일(height : 256)
  - 필요한 소스 코드
    - 다운로드

      activitynet/ActivityNet
      This repository is intended to host tools and demos for ActivityNet - activitynet/ActivityNet
      https://github.com/activitynet/ActivityNet/tree/master/Crawler/Kinetics

      - 필수기능 : youtube-dl
    - 전처리

      facebookresearch/video-nonlocal-net
      We have prepared one copy of the Kinetics dataset. To obtain the dataset, please email xiaolonw@cs.cmu.edu. Our copy includes
      https://github.com/facebookresearch/video-nonlocal-net/blob/master/DATASET.md

      - 과정1 : 라벨링 리스트(classids.json) 기준으로 각 동영상 경로당 라벨링 숫자 매칭된 txt파일 생성
      - 과정2 : 동영상을 height를 256 pixel로 조정
- csv 파일 3개(train / val / test)

```
<동영상 파일 저당된 경로> <라벨링에 매칭이 되는 숫자>
ex.
/root/SlowFast/Datasets/sample_val_256/applauding/-r8c7F4tOI8_000054_000064.mp4 3
```

⚠ 이슈 사항

- 데이터셋 용량이 너무 커서 서버에 저장하기 힘듬(대안으로 샘플로 약 10개로 테스트 진행)
- 전처리한 전체 데이터셋 요청함(xiaolonw@cs.cmu.edu)

1.2 Inference

- 동영상 파일
- 라벨링 파일(.json)
    - class name ~ id mapping
        - {"class_name1": id1, "class_name2": id2, ...}
- CHECKPOINT_FILE_PATH (더 파악해야 함)

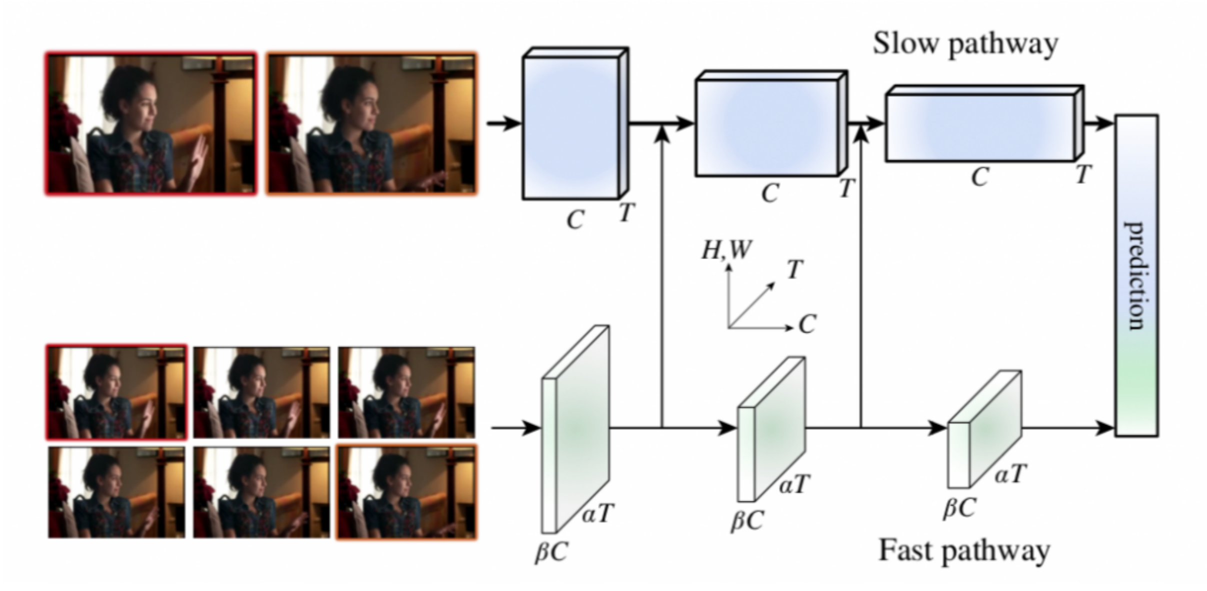
1.2.1 output

- 동영상

특징

- detection은 detectron2(https://github.com/facebookresearch/detectron2)를 사용하는 것으로 보임

2. Model



- Backbone
  - 2D
    - ResNet50 또는 ResNet101 로 진행
  - Two-Stream Method(Slow, Fast)
- Idea
  - 사람 망막의 작동을 모방해서 Two-Stream 방식으로 설계
    - 사람의 시각 시스템은 공간과 움직임 두 가지를 인식하는 부분이 별도 존재
      - M-Cells : 움직임 → Fast
      - P-Cells : 공간 구조, 색 → Slow
      - 기능 비율 → M-Cells : P-Cells = Fast : Slow = 2 : 8 동일하게 설정

MOC 모델 데이터 구조

추론 데이터

- 동영상 W / H 사이즈 제약은 없음
    - Default로 288 x 288 로 사이즈 조정해서 입력(Output은 원본 사이즈로 나옴)
- 입력 비디오 : .mp4 ⇒ 1차 변환 : .jpg ⇒ 추론 : .pkl ⇒ 이미지에 그리기 : .jpg ⇒ 합치기 : .avi
- Pre-trained Model (UFC 데이터로 학습 / 7초 단위로 동영상 분할) 동영상이 7초 내외가 아니면 에러가 발생

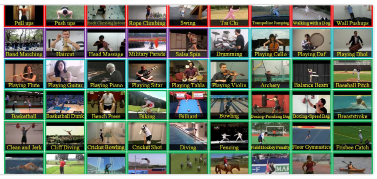
```
(MOC) root@c8802d0806ad:~/MOC-Detector/src/vis# python vis_det.py  --vname test2.mp4 --inference_dir ../../inference/myvideo/test2
start extracting frames
create rgb model
loaded ../../experiment/result_model/ucf_dla34_K7_rgb_coco.pth, epoch 0
Traceback (most recent call last):
  File "vis_det.py", line 174, in <module>
    det()
  File "vis_det.py", line 157, in det
    stream_inference(opt)
  File "vis_det.py", line 125, in stream_inference
    detector = MOCDetector(opt)
  File "../detector/stream_moc_det.py", line 29, in __init__
    self.rgb_model_backbone, self.rgb_model_branch = load_inference_model(self.rgb_model_backbone, self.rgb_model_branch, opt.rgb_
  File "../MOC_utils/model.py", line 84, in load_inference_model
    branch.load_state_dict(state_dict, strict=False)
  File "/root/miniconda3/envs/MOC/lib/python3.5/site-packages/torch/nn/modules/module.py", line 719, in load_state_dict
    self.__class__.__name__, "\n\t".join(error_msgs)))
RuntimeError: Error(s) in loading state_dict for MOC_Det:
        size mismatch for branch.hm.0.weight: copying a param of torch.Size([256, 1920, 3, 3]) from checkpoint, where the shape is
        size mismatch for branch.mov.0.weight: copying a param of torch.Size([256, 1920, 3, 3]) from checkpoint, where the shape i
        size mismatch for branch.mov.2.bias: copying a param of torch.Size([60]) from checkpoint, where the shape is torch.Size([1
        size mismatch for branch.mov.2.weight: copying a param of torch.Size([60, 256, 1, 1]) from checkpoint, where the shape is
(MOC) root@c8802d0806ad:~/MOC-Detector/src/vis#
```

- 참고

CRCV | Center for Research in Computer Vision at the University of Central Florida

There will be a workshop in ICCV'13 with UCF101 as its main competition benchmark: The First International Workshop on Action Recognition with Large Number of Classes. Click here to check the published results on UCF101 (updated October 17, 2013) UCF101 is an action

https://www.crcv.ucf.edu/data/UCF101.php



학습 데이터

- 입력 데이터
    - 동영상을 K초 단위로 분할한 그냥 이미지 (ex. K = 7)
    - K초 단위로 분할한 Flow 이미지(위의 이미지와 1:1 매칭)
    - 라벨링 파일(.pkl)

라벨링

- 행동인식(flow 영상)
- 바운딩 박스(.pkl)

```
"""
Abstract class for handling dataset of tubes.

Here we assume that a pkl file exists as a cache. The cache is a dictionary with the following keys:
    labels: list of labels
    train_videos: a list with nsplits elements, each one containing the list of training videos
    test_videos: idem for the test videos
    nframes: dictionary that gives the number of frames for each video
    resolution: dictionary that output a tuple (h,w) of the resolution for each video
    gttubes: dictionary that contains the gt tubes for each video.
            Gttubes are dictionary that associates from each index of label, a list of tubes.
            A tube is a numpy array with nframes rows and 5 columns, <frame number> <x1> <y1> <x2> <y2>.
"""
```

- gttubes : 분할한 영상에 대한 프레임별 이미지에 대한 바운딩 박스 정보
  - 구조 : <label name>/<dir name> : {label idx: [frameNum, Xmin, Ymin, Xmax, Ymax]}

    ```
    {'Skiing/v_Skiing_g06_c07':
        {16: [array([[  1., 199.,  61., 239., 125.],
                     [  2., 199.,  61., 239., 125.],
                     [  3., 190.,  65., 231., 126.],
                     ...,
                     [298., 166.,  61., 211., 120.],
                     [299., 162.,  61., 207., 120.],
                     [300., 162.,  61., 207., 120.]], dtype=float32)]},
    ...}
    ```

  - sample(확인결과)