FairMOT Embedding 조사

전제조건

- Tracking Service에 적용 가능유무
  - 매번 새로운 데이터에 대해서 학습을 할 수는 없다.
    - 준비된 모델로 해결을 해야한다.(학습 최소화)
  - 타겟에 대한 도메인은 존재하고, 그 도메인으로 학습할 수 있다고 하자.
    - ex. 특정 사람에 대한 트래킹이라면 그 사람을 제외한 일부 이미지 데이터에 대한 학습은 된 것이다.
- Embedding 적용해서 결과물이 사용가능한지?
  - 발표된 모델에 대한 논문에서 적용한 embedding은 뭔지? 어떤 과정을 거치는지? 더 보완할 수 있는지?

적용방안(FairMOT)

개발

- 기존 모델인 **Towards-Realtime-MOT**에서 아래와 같은 부분을 보완한 것으로 보임

> 요약 : When two person is close enough, the center points if the two person fall into same grid, the embedding features seem to be ambiguous

  - https://github.com/Zhongdao/Towards-Realtime-MOT/issues/132

이론

- [x] ~~Re-ID Embedding 작동 원리가 뭔지 알아내기~~

  논문 내용

  > ID 임베딩 브랜치의 목표는 서로 다른 객체를 구별 할 수있는 기능을 생성하는 것입니다. 이상적으로는 서로 다른 물체 간의 거리가 동일한 물체 간의 거리보다 커야합니다. 목표를 달성하기 위해 백본 기능 위에 128 개의 커널이 있는 컨볼 루션 계층을 적용하여 각 위치에 대한 ID 임베딩 기능을 추출합니다. 결과 기능 맵은 E ∈ R^128 × W × H 입니다. (x , y) 에있는 객체의 Re-ID 특성 E_x,y ∈ R^128 이 특성 맵에서 추출됩니다.

  핵심 키워드 : identity embedding feature

  정리

  이미지

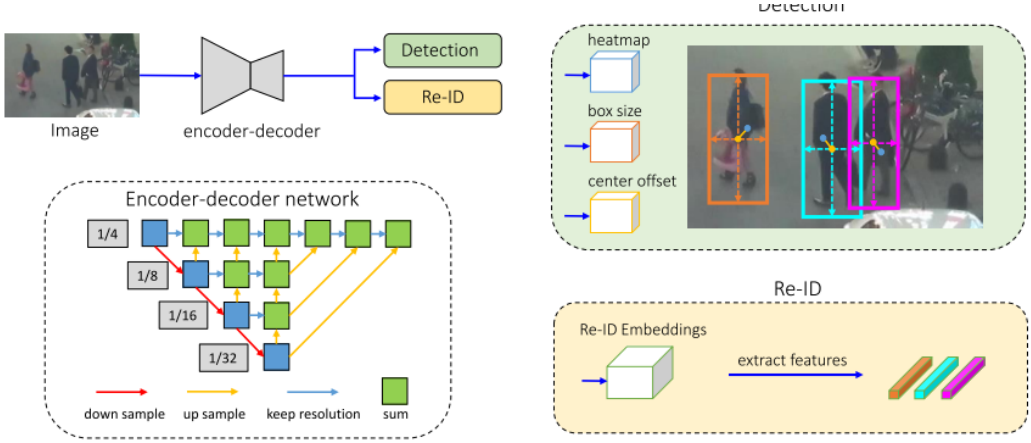

Fig. 2: Overview of our one-shot MOT tracker. The input image is first fed to an encoder-decoder network to extract high resolution feature maps (stride=4). Then we add two simple parallel heads for predicting bounding boxes and Re-ID features, respectively. The features at the predicted object centers are extracted for temporal bounding box linking.

Backbone(**DLA-34**) 특징 위에 128개 kernel이 포함된 conv_layer 적용해서 각 위치에 대한 Identity Embedding Feature 추출

참고

tistory 블로그

GCP ML Docs

코드

- Embedding 구현한 부분 찾기
  - src/lib/opts.py

```
222    if opt.task == 'mot':
223      opt.heads = {'hm': opt.num_classes,
224                   'wh': 2 if not opt.ltrb else 4,
225                   'id': opt.reid_dim}
226    if opt.reg_offset:
227      opt.heads.update({'reg': 2})
```

  - src/lib/models/networks/pose_dla_dcn.py

```
445    self.heads = heads
446    for head in self.heads:
447        classes = self.heads[head]
448        if head_conv > 0:
449          fc = nn.Sequential(
450              nn.Conv2d(channels[self.first_level], head_conv,
451                kernel_size=3, padding=1, bias=True),
452              nn.ReLU(inplace=True),
453              nn.Conv2d(head_conv, classes,
454                kernel_size=final_kernel, stride=1,
455                padding=final_kernel // 2, bias=True))
456        if 'hm' in head:
457          fc[-1].bias.data.fill_(-2.19)
458        else:
459          fill_fc_weights(fc)
460      else:
461        fc = nn.Conv2d(channels[self.first_level], classes,
462            kernel_size=final_kernel, stride=1,
463            padding=final_kernel // 2, bias=True)
```

```
464       if 'hm' in head:
465         fc.bias.data.fill_(-2.19)
466       else:
467         fill_fc_weights(fc)
468     self.__setattr__(head, fc)
```

- Embedding 거리비교
  - src/lib/tracker/matching.py

```
93    def embedding_distance(tracks, detections, metric='cosine'):
94
95        # :param tracks: list[STrack]
96        # :param detections: list[BaseTrack]
97        # :param metric:
98        # :return: cost_matrix np.ndarray
99
100
101       cost_matrix = np.zeros((len(tracks), len(detections)), dtype=np.float)
102       if cost_matrix.size == 0:
103           return cost_matrix
104       det_features = np.asarray([track.curr_feat for track in detections], dtype=np.float)
105       #for i, track in enumerate(tracks):
106           #cost_matrix[i, :] = np.maximum(0.0, cdist(track.smooth_feat.reshape(1,-1), det_features, metric))
107       track_features = np.asarray([track.smooth_feat for track in tracks], dtype=np.float)
108       cost_matrix = np.maximum(0.0, cdist(track_features, det_features, metric))  # Nomalized features
109       return cost_matrix
```

- 원본 GitHub Issue 참조
  - https://github.com/ifzhang/FairMOT/issues/156
  - https://github.com/ifzhang/FairMOT/issues/146

MCMOT Inference

적용 결과

MINIDRONE_Video_Dataset

> 드론 영상으로 찍은 벤치마킹 데이터셋(근거리, TopView)

출처 : https://www.epfl.ch/labs/mmspg/downloads/mini-drone/

Aa 구분	res_resdcn_18_2	res_resdcn_18_vis1	res_hrnet_18_deconv2
backbone	ResNet-18	ResNet-18	HRNet-18
train	CityPersons(근거리,SideView)	VisDrone(원거리,TopView)	CityPersons(근거리,SideView)
성능	중	하	상

YouTube(Seoul 4K 영상)

> 원거리, 근거리 / TopView, SideView

출처 : https://youtu.be/o70MzTHHNbI

Aa 구분	seoul_resdcn_18	seoul_resdcn_18_vis	seoul_hrnet_18_deconv
backbone	ResNet-18	ResNet-18	HRNet-18
train	CityPersons(근거리,SideView)	VisDrone(원거리,TopView)	CityPersons(근거리,SideView)
성능	근거리:중/원거리:하	근거리:하/원거리:중	근거리:상/원거리:중하

서비스 적용시 고려될 내용

- 도메인 데이터셋에 영향이 있음.
  - 예를 들면 학습한 데이터셋의 물체의 크기와 구성비율에 따른 영향
    - dataset1 : 자동차 블랙박스(근거리, 사이드뷰)
    - dataset2 : 드론 카메라(원거리, 탑뷰)
  - 추론할 데이터가 어디에 더 유사한지가 성능에 영향을 줌
- backbone으로 ResNet보다 HRNet이 더 성능이 좋음.
  - 특징으로 down-sampling / up-sampling 하면서 입력된 이미지에 대한 해상도를 유지해 나가는 방식이 성능향상에 도움이 되는 걸로 판단됨.

How to Inference?

`src/demo.py`

> 원본 소스코드인 FairMOT가 42줄 인데 반해, MCMOT는 223줄 까지(거의 5배) 늘렸다.그래도 I/O 데이터는 크게 다르지 않은 걸로 보인다.다만 Input에 대한 범위를 확장해서 단순한 이미지에 대한 결과만 받을 수 있게 한 것 같다.(임베딩 결과 비교 결과를 보려면 필요한 기능)

I/O

- Input
  - Video file or Image file
- Output
  - frame 별 바운딩 박스가 그려진 이미지
  - 위 이미지를 모은 동영상(.mp4) (또는 바운딩 박스 및 라벨 예측한 txt 파일)

Opt

필수설정

> 동영상 파일에 대한 추론을 기준으로 아래 설정을 안 하면 에러가 발생하게 됨

- `opt.input-video`
  - 입력할 video 파일 경로
  - ex

    ```
    --input-video ../videos/sample.mp4
    ```

- `opt.load_model`
  - pretrained model 파일 경로 입력
  - ex

    ```
    --load_model ../weights/mcmot-model.pth
    ```

- `opt.arch`
  - backbone model을 지정해 줘야 함(default='resdcn_18')
    - opt.load_model에 맞춰서 설정해야 함.

    - 'resdcn_18 |resdcn_34 | resdcn_50 | resfpndcn_34 | dla_34 | hrnet_32 | hrnet_18 | cspdarknet_53'
  - ex

```
--arch hrnet_18
```

- `opt.reid_cls_ids`
  - 오브젝트 클래스 라벨링(int)
  - 사용되는 모델의 라벨과 일치해야 함.
  - ex. '0,1,2,3,4'

선택설정

> 이미지 파일에 대한 추론을 고려하거나 여러 파일에 대한 추론으로 결과물 저장경로를 구분

- `opt.output-root`
  - 결과물 저장할 폴더경로 지정(default : '../results')
  - ex

```
--output-root ../results/sample
```

- `opt.input-img`
  - 입력할 이미지 폴더경로 또는 이미지 파일 리스트가 적힌 텍스트 파일 경로
  - ex

```
--input-img ../images/sample  # or ../datas/image_path.txt
```

- `opt.id_weight`
  - 1(default) : Detection & RE-IDA
  - 0(option) : Detection Only
- `opt.input_mode`
  - 'video'(default) : 비디오 파일 입력

```
61    data_loader = datasets.LoadVideo(opt.input_video, opt.img_size)  # load video as input
```

  - 'image_dir'(option) : 이미지 폴더 경로 또는 파일 경로

```
65    data_loader = datasets.LoadImages(opt.input_img, opt.img_size)  # load images as input
```

- 'img_path_list_txt'(option) : 이미지 경로 리스트(Detection 모드만 가능)

```
74    opt.id_weight = 0  # only do detection in this mode
75    with open(opt.input_img, 'r', encoding='utf-8') as r_h:
76        logger.info('Starting detection...')
77        paths = [x.strip() for x in r_h.readlines()]
78        print('Total {:d} image files.'.format(len(paths)))
79        data_loader = datasets.LoadImages(path=paths, img_size=opt.img_size)
```

Function

> 원본 FairMOT에서 demo() 하나만 있었지만, MCMOT에서는 기능을 확장한 만큼 함수도 run_demo()와 test_single() 두가지로 늘렸다.run_demo()는 메인 함수로 전반적인 상황에 맞게 inference를 할 수 있는 것으로 보이고, test_single()은 이미지 한장에 대한 특수한 경우에 사용할 수 있도록 만든 것으로 보임.

- `run_demo(opt)`
  - 'video', 'img_dir', 'img_file' 에 대한 데이터를 읽는다. 크게 video와 image로 구분할 수 있다.
  - Data Load

```
24    import lib.datasets.dataset.jde as datasets
"""
class LoadVideo
    :param path
    :param img_size=(1088, 608)
"""
```

  - Predict

```
# predict
25    from track import eval_seq, eval_imgs_output_dets
"""
case1. video & track => eval_seq(..., mode='track')
case2. video & detect => eval_seq(..., mode='detect')
case3. image => eval_imgs_output_dets() # Detection
"""
```

- `test_single(img_path, dev)`
  - img_path의 이미지 파일을 입력받아서 예측한 바운딩 박스가 그려진 이미지를 리턴 받는 함수

```
# ex
test_single(img_path='/mnt/diskb/even/MCMOT/src/00000.jpg', dev=torch.device('cpu')) # or 'gpu'
```

MCMOT Dataset

DataSet

- 아래와 같이 하나의 데이터 셋에 대해 크게 3가지 형식의 파일이 준비가 되어야 합니다.

VisDrone

메타정보

- 참조 : https://github.com/VisDrone/VisDrone2018-MOT-toolkit

Pre-processing

폴더 구성 및 라벨링 변환

`src/gen_dataset_visdrone.py`

- 주요 기능
  1. 폴더 구조에 맞게 데이터 위치 이동

```
VisDrone2019/
├── images
│   ├── train
│   └── val
└── labels_with_ids
    ├── train
    └── val
```

  2. label 좌표 변형

```
{object_category} {target_id} {x_center/image_width} {y_center/image_height} {bbox_width/image_width} {bbox_height/image_heig
```

     - object_category : 라벨과 매칭되는 정수
     - target_id : identity, 매 프레임별 같은 오브젝트에 부여된 정수

이미지 파일 경로 리스트

`src/gen_imgpath_list.py`

- 주요 기능
  1. 이미지 경로 종합된 텍스트 파일 생성
     - ex. src/data/visdrone.train

```
VisDrone2019/images/train/uav0000244_01440_v/0000638.jpg
VisDrone2019/images/train/uav0000244_01440_v/0000184.jpg
VisDrone2019/images/train/uav0000244_01440_v/0000239.jpg
...
```

데이터셋 설정 파일

`lib/cfg/visdrone.json`

- 경로 설정 해주면 됨

```
{
  "root":"/workspace/datasets",
  "train":
  {
      "visdrone":"./data/visdrone.train"
  },
  "test_emb":
```

```
  {
      "visdrone":"./data/visdrone.val"
  },
  "test":
  {
      "visdrone":"./data/visdrone.val"
  }
}
```