모션 키포인트 검출

CV분반_2팀 16기 신인섭 윤지현 임채명 17기 진유석

목차

Part 1,

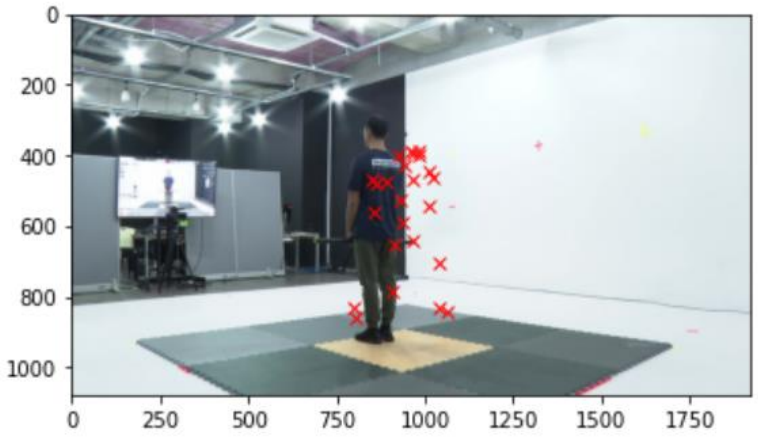
데이터 소개

Part 1,

데이터 소개 및 프로젝트 소개

```
[ ] train = pd.read_csv('train_df.csv')
    submission = pd.read_csv('sample_submission.csv')

    train.head(5)
```

	image	nose_x	nose_y	left_eye_x	left_eye_y	right_eye_x	right_eye_y
0	001-1-1-01-Z17_A-0000001.jpg	1046.389631	344.757881	1041.655294	329.820225	1059.429507	334.484230
1	001-1-1-01-Z17_A-0000003.jpg	1069.850679	340.711494	1058.608552	324.593690	1075.242111	325.593690
2	001-1-1-01-Z17_A-0000005.jpg	1084.475902	337.000008	1078.717997	323.757889	1095.648412	325.242119
3	001-1-1-01-Z17_A-0000007.jpg	1042.320047	361.452689	1037.907194	344.117804	1050.328382	353.913729
4	001-1-1-01-Z17_A-0000009.jpg	1058.046395	343.164191	1046.717997	331.703163	1058.132650	331.781079



데이터 설명

모션 이미지와 신체 24개의 부위에서 측정한 포인트

- train image: 4195장
- test image: 1600장
- train image의 이름: 4195*49
- keypoint 24지점의 x, y 좌표: 4195*49

프로젝트 설명

특정 운동 동작을 수행하고 있는 사람의 미리 지정된 각 신체 부위의 위치에서 측정한 데이터를 활용하여 motion keypoint detection 알고리즘 개발

Part 2,

전처리

Part 2,

전처리

error_list

train image에서 keypoint 검출이 제대로 되지 않은 이미지들이 존재하여
대회에서 지정한 이미지들을 기존 train set에서 제거해 줌

```
[ ]  error_list = [317, 869, 873, 877, 911, 1559, 1560, 1562, 1566, 1575]
     error_list += [1577, 1578, 1582, 1606, 1607, 1622, 1623, 1624, 1625]
     error_list += [1629, 3968, 4115, 4116, 4117, 4118, 4119, 4120, 4121]
     error_list += [4122, 4123, 4124, 4125, 4126, 4127, 4128, 4129, 4130]
     error_list += [4131, 4132, 4133, 4134, 4135, 4136, 4137, 4138, 4139]
     error_list += [4140, 4141, 4142, 4143, 4144, 4145, 4146, 4147, 4148]
     error_list += [4149, 4150, 4151, 4152, 4153, 4154, 4155, 4156, 4157]
     error_list += [4158, 4159, 4160, 4161, 4162, 4163, 4164, 4165, 4166]
     error_list += [4167, 4168, 4169, 4170, 4171, 4172, 4173, 4174, 4175]
     error_list += [4176, 4177, 4178, 4179, 4180, 4181, 4182, 4183, 4184]
     error_list += [4185, 4186, 4187, 4188, 4189, 4190, 4191, 4192, 4193, 4194]
```

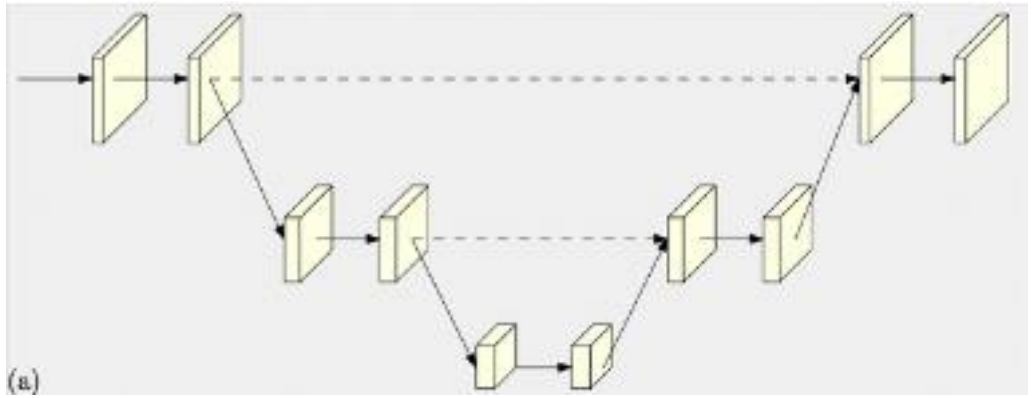
Part 3,

모델링 및 결과

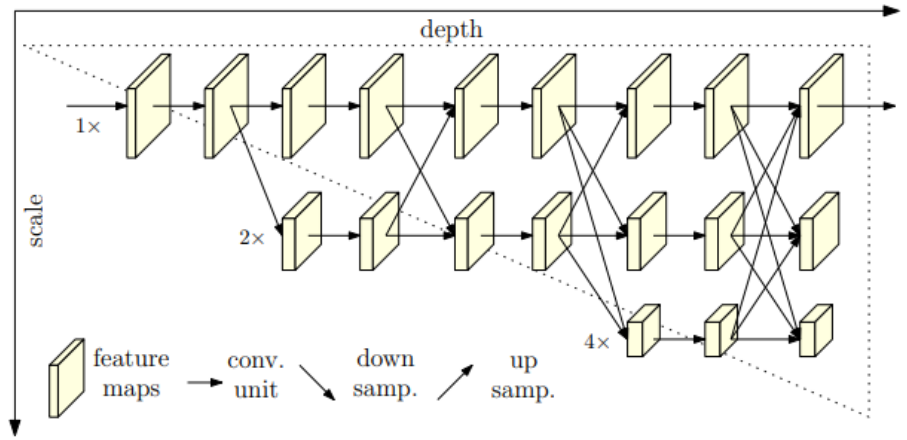
모델링

모델1. HRNet

이전 pose estimation model



HR Net



- DownSampling 과정에서 기존의 해상도는 유지한 상태로 **Parallel 하게 진행**이 된다.
- Parallel한 **Sub-network 간에 계속 정보 교환**.

➡ **다양한 해상도에서 전체적인 맥락과 국소적인 정보를 지속적으로 교환**

모델링

모델2. UDP Pose

기존 SOTA 모델들
인코딩, 디코딩 데이터 처리 방식 문제점

- Flipping strategy의 결과가 원래 추론 결과와 일치하지 않음
- 인코딩 디코딩 통계적인 오류

→ 포즈 추정 성능 저하

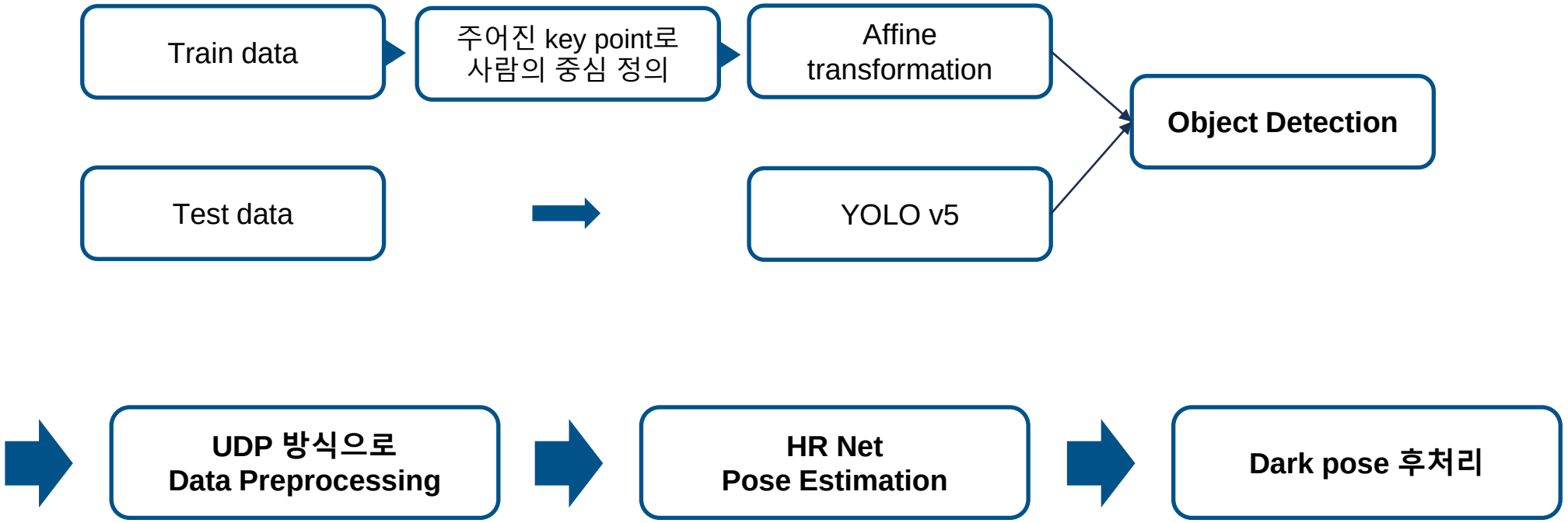
UDP(The Unbiased Data Processing)

- 데이터를 이산 공간(픽셀)이 아닌 연속적인 공간(픽셀 간격) 기반으로 처리
- 인코딩, 디코딩 수행시에 분류, 회귀 방식 결합

→ 기존의 모델의 성능 향상
특히 HR net에서 좋은 성능을 보임

모델링

모델2. UDP Pose



결과

모델2. UDP Pose

- Training 조건 변경

- **Epoch size** : 20, 30, 40, 50 → 30에서 가장 좋은 성능
- **Batch size**

Batch : 8

372.0159904255
278.1335910801

Batch : 16

277.8507982296
225.1213049702

Batch : 28

284.079399193
228.2797671704

- **Input size**

291.0409120932
240.8418759239

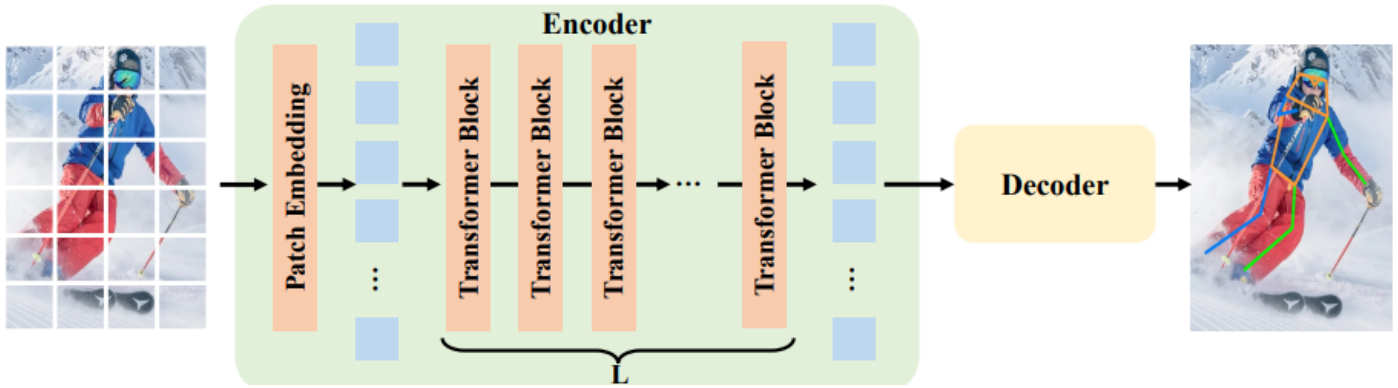
- **Loss function 변경(MSE → Cross Entropy)**

430.5397305337
312.6910296044

모델링

모델3. ViTPose

오직 transformer만을 이용하여 pose estimation을 수행



Simplicity
단순한 프레임워크로 구성
plain한 인코더와 단순한 디코더

Flexibility
Input/feature의 해상도가 달라도 적응 가능
단일 pose 데이터셋을 훈련하여
다중pose에 사용 가능

Scalability
모델의 크기 키우기 가능
Feature의 차원 쉽게 조절 가능

Transferability
Transfer 학습과 지식 증류에서 좋은 성능

Part 3,

모델링 및 프로젝트 시도

1. 시각화를 통해 학습 정도 즉각 확인

```
heatmap_vis = vis_heatmaps(outputs[0, :24, :, :].detach().cpu().numpy().astype(np.uint8))
target_vis = vis_heatmaps(target[0].detach().cpu().numpy()).astype(np.uint8)
check = cv2.hconcat([heatmap_vis, target_vis])
cv2.imwrite("Check.jpg", check)
cv2.imwrite("Train_Target_Heatmap.jpg",target_vis)
```

키포인트와 히트맵을 시각화할 수 있는 함수를 구현함으로써 모델의 학습 정도를 잘 파악하도록 함

- 250 epoch 학습 후의 예상 heatmap

prediction

ground truth

모델링 및 프로젝트 시도

2. 좌우 반전을 통한 Data Augmentation

Motion Keypoint Baseline을 참고하여 데이터 증강

: Keypoint detectio의 경우, image data augmentation시 x, y 좌표값을 함께 바꾸어 주어야 함

```
# 좌우 반전
def left_right_flip(images, keypoints):
    flipped_keypoints = []
    flipped_images = np.flip(images, axis=1)
    for idx, sample_keypoints in enumerate(keypoints):
        if idx%2 == 0:
            flipped_keypoints.append(480.-sample_keypoints)
        else:
            flipped_keypoints.append(sample_keypoints)

    # left_right_keypoints_convert
    for i in range(8):
        flipped_keypoints[2+(4*i):4+(4*i)], flipped_keypoints[4+(4*i):6+(4*i)] = flipped_keypoints[4+(4*i):6+(4*i)], flipped_keypoints[2+(4*i):4+(4*i)]
    flipped_keypoints[36:38], flipped_keypoints[38:40] = flipped_keypoints[38:40], flipped_keypoints[36:38]
    flipped_keypoints[44:46], flipped_keypoints[46:48] = flipped_keypoints[46:48], flipped_keypoints[44:46]

    return flipped_images, flipped_keypoints
```

모델링 및 프로젝트 시도

3. ViTPose 모델 fine-tuning

기존 ViTPose 모델

- 17개의 keypoint 검출
- Skeleton 함수 구현

▶▶▶

- Skeleton 함수 구현 X → 좌표 찍어 저장
- 마지막 classification layer의 class label을 24개로 변경

```
self.class_labels = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
                     13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23]
```

- 마지막 Cov2D layer의 최종 channel을 24개로 변경

```
model.keypoint_head.final_layer = nn.Conv2d(256, 24, kernel_size=(1, 1), stride=(1, 1))
```

모델링 및 프로젝트 시도

3. ViTPose 모델 fine-tuning

- 모델 학습 진행

1. 시각화를 통해 학습 정도 즉각 확인
2. 좌우 반전을 통한 Data Augmentation
3. 192*256에서 pretrained 되어 resize 진행

```
for i, (input, target, joints) in enumerate(train_loader):
    #print(input.shape, target.shape, joints.shape)
    input = torch.Tensor(input).cuda()
    outputs = model(input)
    #print(outputs.shape)
    outputs = F.interpolate(outputs[:, :24, :, :], size=(256, 192), mode='bilinear', align_corners=True)
    target = target.cuda(non_blocking=True)
```

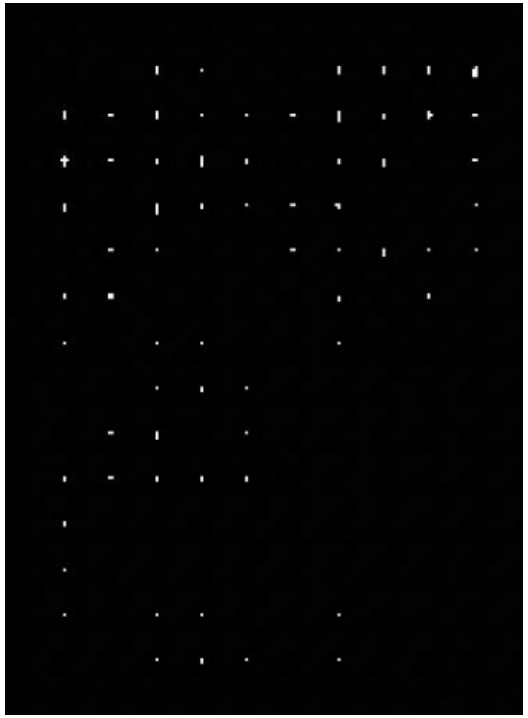
- ViTPose-H 모델의 가중치로 fine-tuning

```
import configs.ViTPose_base_coco_256x192 as b_cfg
import configs.ViTPose_large_coco_256x192 as l_cfg
import configs.ViTPose_huge_coco_256x192 as h_cfg

cfg = h_cfg

with open('config.yaml', 'r', encoding='utf-8') as f:
    cfg_yaml = yaml.full_load(f)

for k, v in cfg_yaml.items():
    cfg.__setattr__(k, v)

model = ViTPose(cfg=cfg)
pretrained = torch.load('vitpose-h-multi-coco.pth')
model.load_state_dict(pretrained)
print(model)
```

Part 4,

결과

- ViTPose-H 모델 fine-tuning 결과



- 다른 여러 시도들

이전 layer 수정 시도:
ViTPose 모델 자체가 너무 무거워 주어진 학습 환경에서 학습 불가능
ViTPose-B/L 모델을 사용하여 fine-tuning 시도:
마찬가지로 학습 불가능/성능 좋지 않음

》》 ViTPose 모델 자체가 파라미터 수가 많은 무거운 모델이어서 주어진 학습 환경의 제한 받은 것으로 예측됨

- 최종 모델 성능_HRNet

Part 4,

프로젝트 소감

신인섭

key point estimation 분야에 대해 여러 모델들을 비교하면서 공부할 수 있어서 좋았습니다.
시간이 있었으면 Single human에 최적화된 Efficient pose 모델로도 해보고
기존 HRnet 결과와 비교해보고 싶습니다.

윤지현

ViTPose 모델의 구현에는 성공하였지만 다른 모델들을 사용하여 성능을 높이는 시도를 해보지 못해 아쉽습니다.

임채명

test 데이터에서 bbox를 추출하기 위해 yolov5를 사용했는데, 다른 모델을 사용하거나
앙상블해보지 못해 아쉽습니다.

진유석

Key point estimation 분야를 처음 해 보았는데, 여러 모로 부족한 점이 많은 것을 느낀 프로젝트였습니다. 다음에 기회가 된다면 지금 배운 것을 바탕으로 더 좋은 결과를 내 보도록 하겠습니다.

「

감사합니다

」