KUBIG
DATA SCIENCE & AI

인공지능 요가 선생

Team | YOGIS

18기 원준혁, 19기 김은우, 이서현, 이소희

KUBIG
DATA SCIENCE & AI

CONTENTS

01. Pose Estimation

Pose Estimation이란 ?

주요 신체 관절의 공간적 위치를 식별하여
사람의 포즈나 자세를 추론하는 것을 목표로 하는 **task**

(a) Input Image

(e) Parsing Results

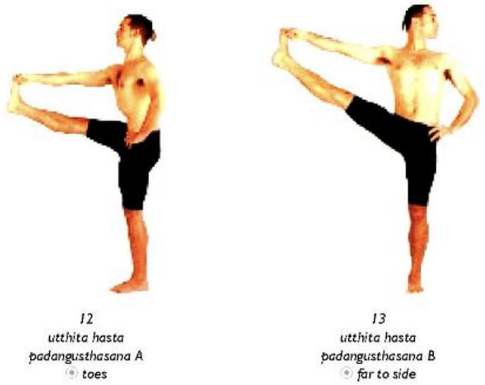
KUBIG
DATA SCIENCE & AI

01. Intro

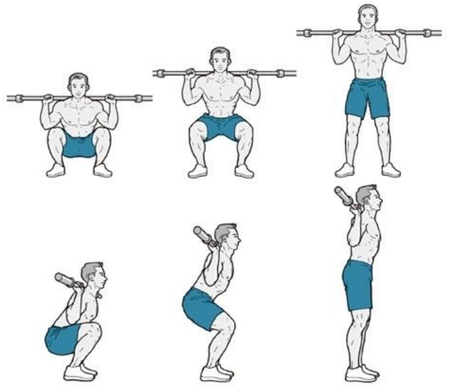
01. Why Yoga?

홈트 관심 증가

비슷해보이지만 다른 **yoga pose**
(classification만으로는 분류가 어려움)



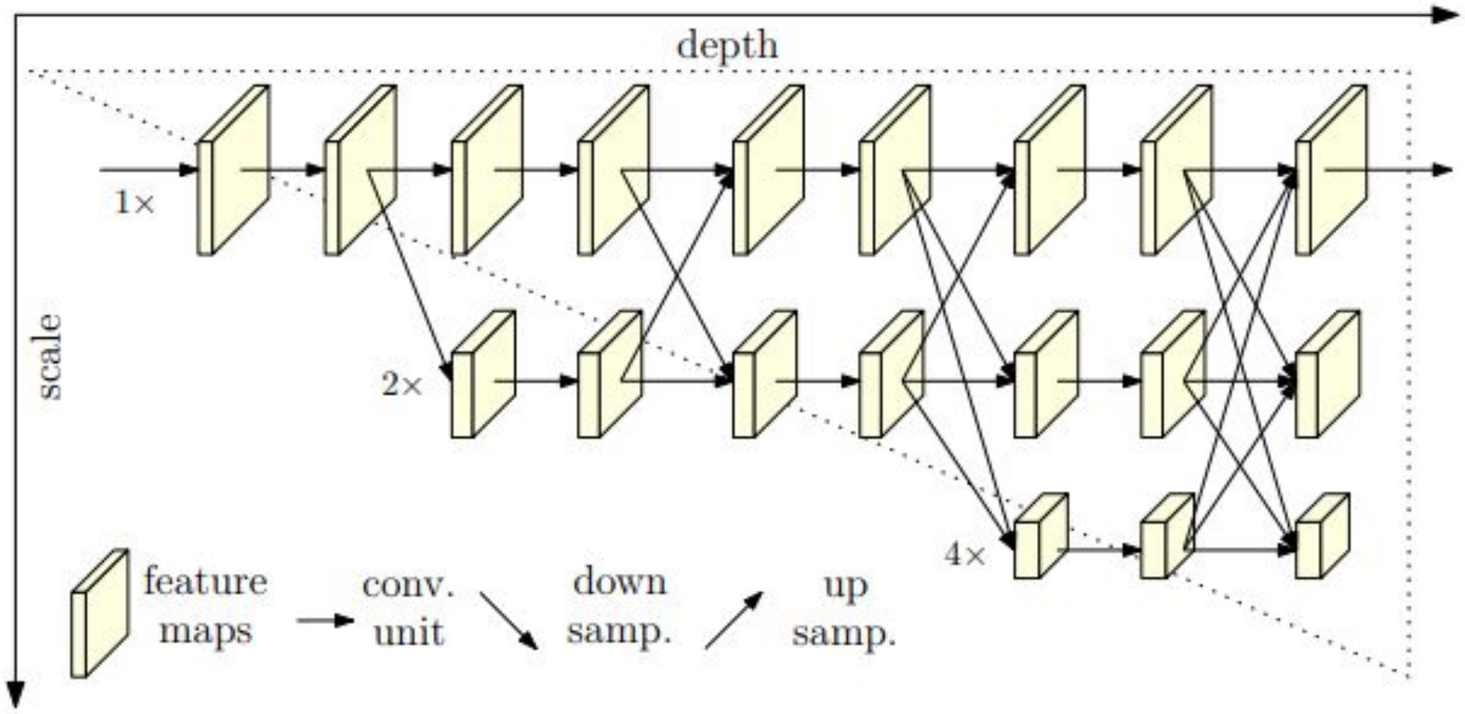
Reproducibility
(헬스 등 다른 분야에도 적용가능)



01. Related works

HRNet

고해상도의 feature map를 최대한 살려서 관절 feature 추출 하는 방식



01. Related works

RTMPose: Real-Time Multi-Person Pose Estimation based on MMPose

CSPNeXT를 백본으로 사용하고, SimCC기반 알고리즘을 사용한 coordinate classification방식

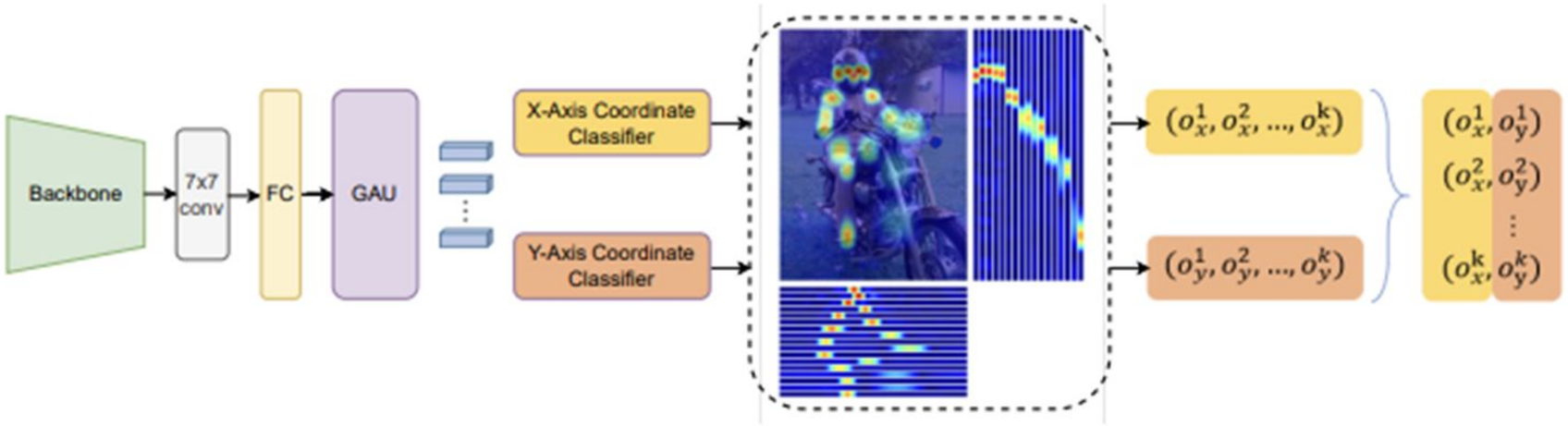


Figure 2. The overall architecture of RTMPose, which contains a convolutional layer, a fully-connected layer and a Gated Attention Unit (GAU) to refine K keypoint representations. After that 2d pose estimation is regarded as two classification tasks for x-axis and y-axis coordinates to predict the horizontal and vertical locations of keypoints.

KUBIG
DATA SCIENCE & AI

02. OpenPose

02. OpenPose

OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields

Part Affinity Fields(PAFs)라는 **비모수적 표현(nonparametric representation)**을 이용해 각 신체 부위를 연결시키는 **bottom-up** 방식

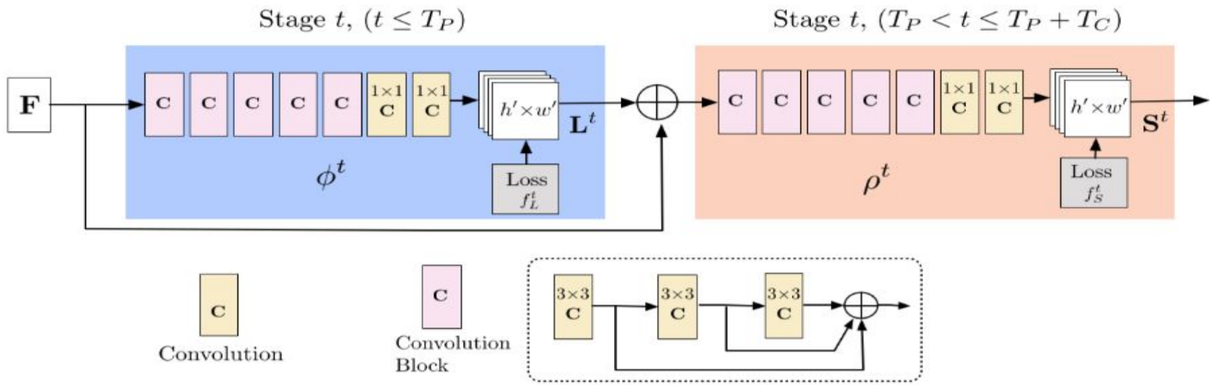


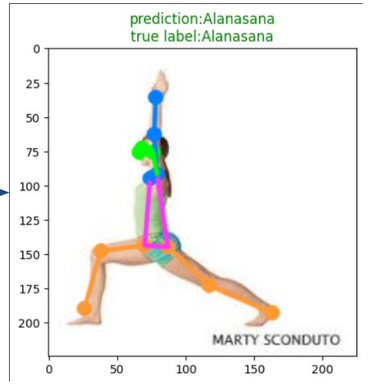
Fig. 3: Architecture of the multi-stage CNN. The first set of stages predicts PAFs L t , while the last set predicts confidence maps S t . The predictions of each stage and their corresponding image features are concatenated for each subsequent stage. Convolutions of kernel size 7 from the original approach [3] are replaced with 3 layers of convolutions of kernel 3 which are concatenated at their end.

- Realtime에서도 high accuracy 달성
- 사람 수에 상관 없이 part를 연결하는 greedy parsing algorithm
- **OpenPose**라는 오픈소스 라이브러리 배포

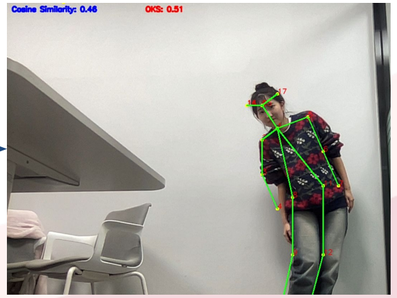
02. Overall Model Structure

Openpose
(Freezed)

Pose Estimation



Feedback



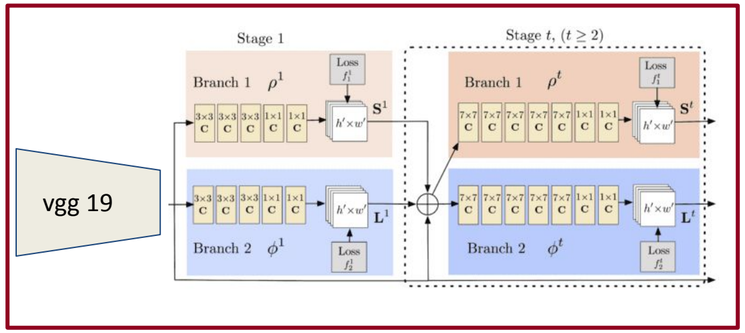
KUBIG
DATA SCIENCE & AI

03. Pose Estimation

03. Pose Estimation Overall Pipeline

Yoga Pose Label

vgg 19



Freezed

Keypoint Features
(54 dimension)

Classifier

Yoga Pose Label

03. Data

Yoga Pose Image Classification Dataset

(107개 Folder 중 13개 Folder만 사용해서 학습 진행)

HxWx3

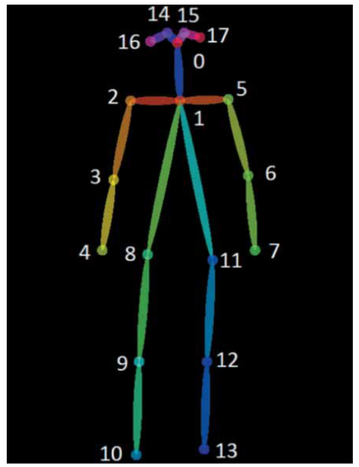
Resize (256x256)
Pixel Normalization
[0, 255] ->[0.0, 1.0]
Channel Normalization
(channel−mean)/std

Open Pose 를 활용한 Keypoint Detect
(x, y , confidence score)

detected_keypoints (list):
(x, y, confidence) tuple

Detect된 Keypoint -> 벡터변환
(54 dimension)

1D Vector



coco based (18 keypoints)

03. Classifier Performance

Input : 54 dimension 1D vector (18개 관절에 대해 x, y, confidence score를 변환한것)

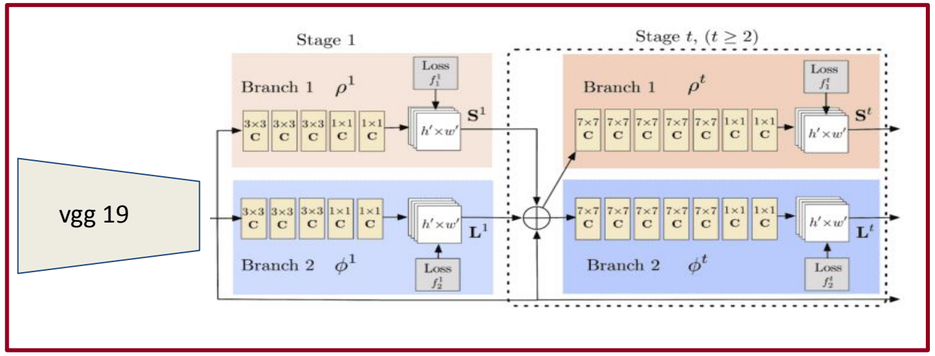
Output : Classified Pose (13개 중 하나로 분류)

KNN	Random Forest	MLP	MLP
Accuracy : 74.29%	**Accuracy : 71.43%**	2개의 linear layer, BatchNorm1d, Drpout, ReLU **Accuracy : 55.21%**	5개의 linear layer, BatchNorm1d, Drpout, ReLU **Accuracy : 75.69%**

03. Pose Estimation Pipeline (with MLP)

256x256x3
Yoga Pose Label

vgg 19



Freezed

Keypoint Features
(54 dimension)

fc1

fc2

fc3

fc4

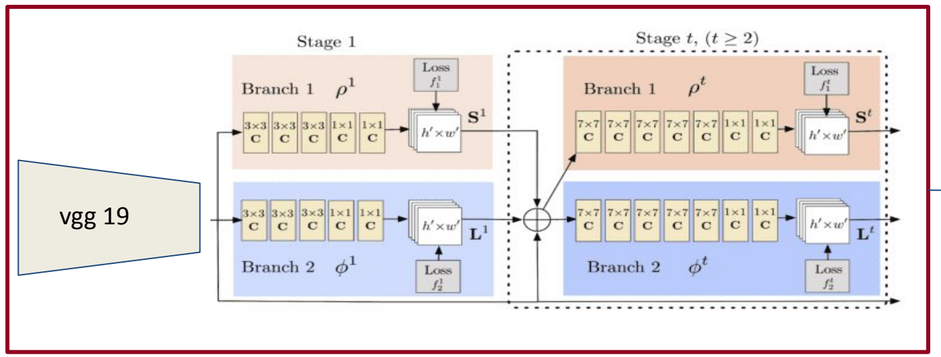
fc5

ReLU

Yoga Pose Label(13 classes)

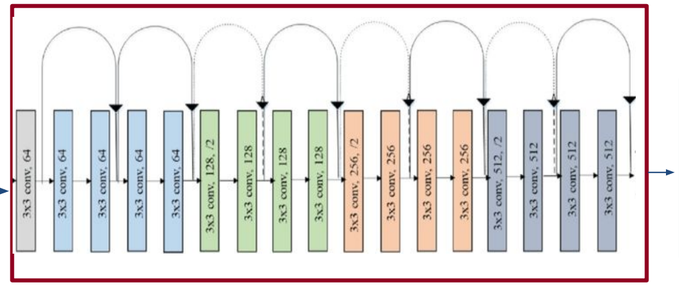
03. Enhanced Model (combining Image features)

256x256x3
Yoga Pose Label



Freezed

image



ResNet18
(pretrained)

nn.identity

512

Image Dim Reducer

128

Keypoint Features
(54 dimension)

fc1

fc2

fc3 ->dim 1024

Concat Feature

fc4

ReLU

Yoga Pose Label(13 classes)

03. Enhanced classifier Performance

KNN	Random Forest	MLP	MLP
Accuracy : 74.29%	**Accuracy : 71.43%**	2개의 linear layer, BatchNorm1d, Drpout, ReLU **Accuracy : 55.21%**	5개의 linear layer, BatchNorm1d, Drpout, ReLU **Accuracy : 75.69%**

Performance Enhanced

Input : 54 dimension 1D vector + Image(256x256)

Output : Classified Pose (13개 중 하나로 분류)

ResNet + MLP
Accuracy : 93.75%

03. Enhanced classifier Performance

Pose estimation Result

KUBIG
DATA SCIENCE & AI

04. Feedback

04. Objective

Image-Image

Openpose에서 검출된
keypoint 및 평가 지표를 활용한
구체적인 피드백 제공

Video-Image

Openpose을 기반으로 한 skeleton
생성 및 평가 지표를 활용하여
실시간 피드백 제공

04. Evaluation Metrics

거리 기반 : OKS(Object Keypoint Similarity)

주석되는 관절의 추정 좌표와
정답 좌표의 유사성 평균

$$OKS = \frac{\sum_i \exp(-d_i^2/2s^2k_i^2)\delta(v_i > 0)}{\sum_i \delta(v_i > 0)}$$

각도 기반 : Cosine Similarity

두 벡터 간 코사인값을 이용하여
측정된 벡터 간 유사한 정도

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

04. Image-Image

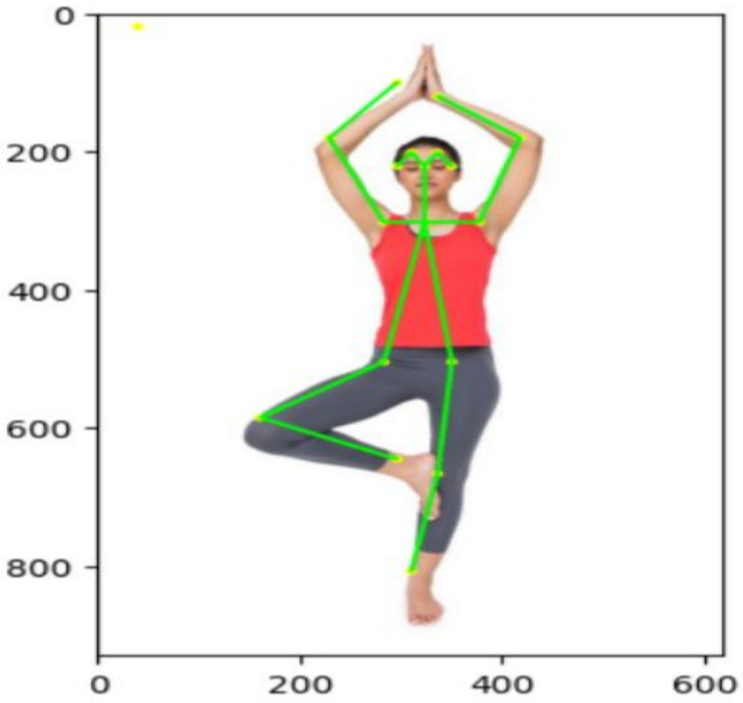
1. 정답 이미지 선정

Tree

Warrior2

04. Image-Image

2. 이미지 keypoint 및 skeleton검출(Tree)

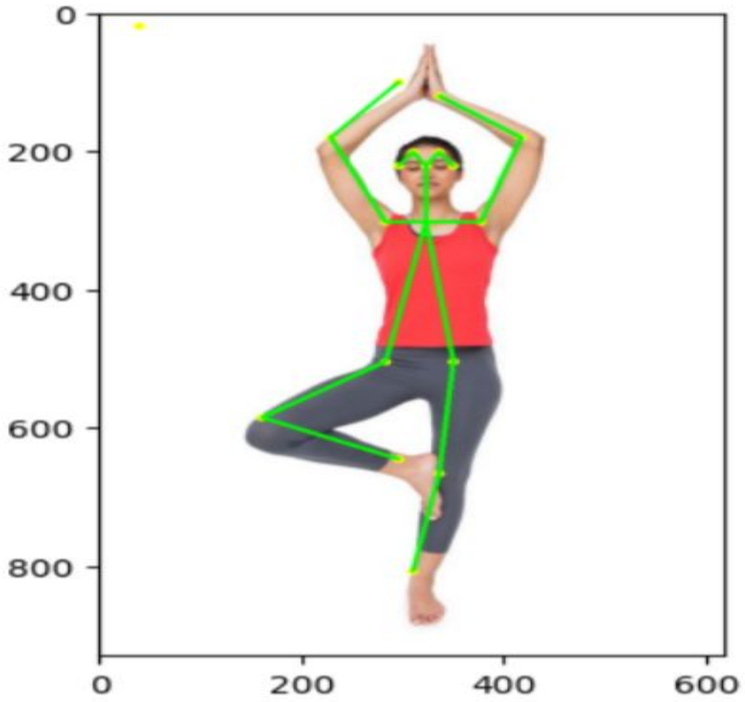


Ground Truth

User

04. Image-Image

3. 피드백 기준 선정



Ground Truth

자세 정확도 (Accuracy)	OKS, Cosine similarity
Open pose confidence	검출된 keypoint 수
관절, keypoint별 자세 평가	Angle, Distance 기반

04. Image-Image

3. 피드백 기준 선정 - 자세 평가 기준(Tree)

1) OKS, Cosine similarity : 특정 경계값 이상

2) 검출된 keypoint 수 : 13개 이상

3) 고개 : 지면과 수직, 어깨 : 지면과 수평

4) 왼쪽 다리 : 일직선으로 서있기

5) 오른쪽 다리 : 다리 접은 각도 정답과 유사

6) 양손 : 손이 서로 맞닿게

7) 팔 : 어깨-팔꿈치, 팔꿈치-손목 각도 정답과 유사

04. Image-Image

3. 피드백 기준 선정 - 자세 평가 기준(Warrior2)

1) OKS, Cosine similarity : 특정 경계값 이상

2) 검출된 keypoint 수 : 13개 이상

3) 고개 : 지면과 수직이면서 측면 바라보기

4) 팔, 어깨, 손목 : 지면과 평행하게 일직선

5) 겨드랑이 : 팔과 몸통 수직

6) 왼쪽 다리 : 몸통과 다리 각도 135도, 일직선

7) 오른 다리 : 무릎 각도 수직

04. Image-Image

4. 실험 및 경계값 조정

```
# 4. 오른 무릎 각도
if all(usr_kp[i] is not None for i in [8,9,10]):
  angle1=calculate_angle_tree(usr_x, usr_y, "8", "9", "10")
  angle2=calculate_angle_tree(gt_x, gt_y, "8", "9", "10")
  if abs(angle1-angle2)<15 :
    cond_4=True
  else :
    cond_4=False
else :
  cond_4=None
cond.append(cond_4)
```

오른 무릎 각도 설정

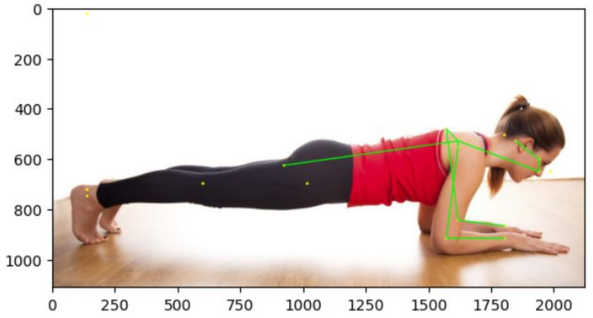
```
oks_coco = keypoint_similarity(resized_gt_kp2,
                               resized_usr_kp2,
                               sigmas=KPTS_OKS_SIGMAS_COCO,
                               areas=area)
value = oks_coco.item()
pose_pair_vectors_gt = get_pose_pair_vectors(gt_vectors, POSE_PAIRS_COCO)
pose_pair_vectors_usr = get_pose_pair_vectors(usr_vectors, POSE_PAIRS_COCO)
similarities = calculate_similarity_between_pose_pairs(pose_pair_vectors_gt, pose_pair_vectors_usr)
value2 = calculate_average_similarity(similarities)
count_none = sum(1 for kp in resized_usr_kp if kp is None)
cond=tree(resized_gt_kp, resized_usr_kp)
count_true = sum(1 for c in cond if c is True)
print("oks :", value)
print("cosine simlarity :", value2)
print("없는 키포인트 :", count_none)
print("맞는 자세 :", count_true)
if value <0.85 :
  print( "전반적인 자세가 많이 다르니 다시 준비해 오세요!")
elif calculate_average_cosine_similarity(resized_gt_kp, resized_usr_kp)<0.6 :
  print( "전반적인 자세가 많이 다르니 다시 준비해 오세요!")
elif count_none >= 5 :
     print( "키포인트가 너무 없으니 다시 준비해 오세요!")
elif count_true<4 :
     print( "고칠게 너무 많으니 다시 준비해 오세요!")
```

경계값 설정 및 피드백 예시

04. Image-Image

5. Result - Tree

Ground Truth



Bad Example

OKS 및 Cosine similarity가 경계값 미만

미검출된 Keypoint가 다수 존재

틀린 자세가 너무 많음

→

전반적인 자세 수정

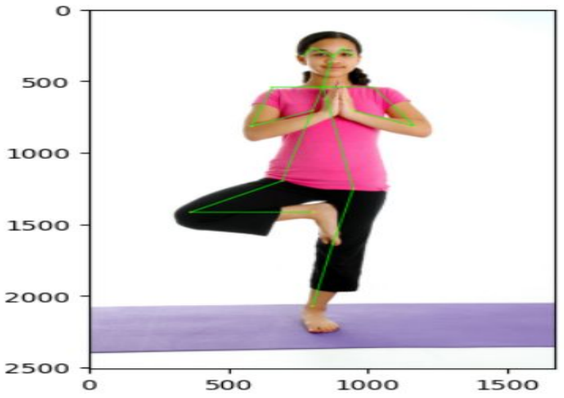
Feedback

```
oks : 0.3616127565540952
cosine simlarity : 0.18660015347728334
없는 키포인트 : 7
맞는 자세 : 0
전반적인 자세가 많이 다르니 다시 준비해 오세요!
```

04. Image-Image

5. Result - Tree

Ground Truth



Require modification

OKS, Cosine 유사도, keypoint 검출 만족

수정해야 할 자세 여럿

→

구체적인 Feedback

Feedback

```
oks : 0.8623756618933693
cosine simlarity : 0.6351362273036315
없는 키포인트 : 0
맞는 자세 : 5
Good! 고개가 바른 자세로 앞을 바라보고 있습니다.
Good! 어깨가 지면과 평행하며 일직선이네요.
Good! 왼쪽 다리가 일직선으로 잘 서 있네요.
Try Again, 오른발 자세를 조정해보도록 하세요.
Try Again, 두 손을 잘 맞닿아보세요.
Try Again, 오른쪽 어깨와 팔꿈치 각도를 조정해보도록 하세요.
Good! 오른쪽 손목과 팔꿈치 각도가 정답 자세와 일치합니다.
Try Again, 왼쪽 어깨와 팔꿈치 각도를 조정해보도록 하세요.
Good! 왼쪽 손목과 팔꿈치 각도가 정답 자세와 일치합니다.
```

04. Image-Image

5. Result - Tree

Ground Truth

Good Example

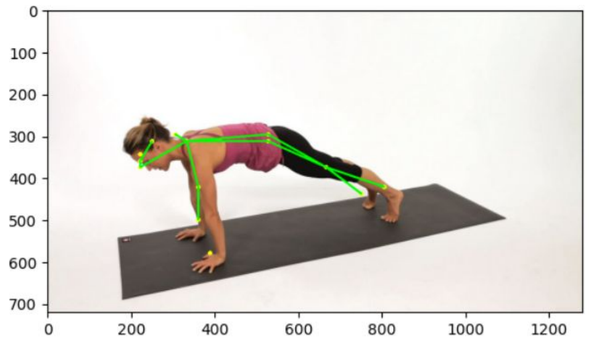
```
oks : 0.9400151800903828
cosine simlarity : 0.9990210640522932
없는 키포인트 : 0
맞는 자세 : 7
Good! 고개가 바른 자세로 앞을 바라보고 있습니다.
Good! 어깨가 지면과 평행하며 일직선이네요.
Good! 왼쪽 다리가 일직선으로 잘 서 있네요.
Try Again, 오른발 자세를 조정해보도록 하세요.
Try Again, 두 손을 잘 맞닿아보세요.
Good! 오른쪽 어깨와 팔꿈치 각도가 정답 자세와 일치합니다.
Good! 오른쪽 손목과 팔꿈치 각도가 정답 자세와 일치합니다.
Good! 왼쪽 어깨와 팔꿈치 각도가 정답 자세와 일치합니다.
Good! 왼쪽 손목과 팔꿈치 각도가 정답 자세와 일치합니다.
```

높은 평가지표
대부분의 자세가 기준에 부합(ground_truth 자세와 유사)

04. Image-Image

5. Result - Warrior2

Ground Truth



Bad Example

OKS 및 Cosine similarity가 경계값 미만

미검출된 Keypoint가 다수 존재

틀린 자세가 너무 많음

→ 전반적인 자세 수정

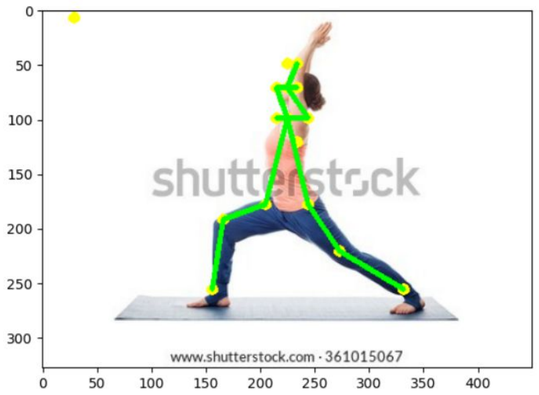
Feedback

```
oks : 0.37429231806801794
cosine simlarity : 0.057885686572290276
없는 키포인트 : 4
맞는 자세 : 2
전반적인 자세가 많이 다르니 다시 준비해 오세요!
```

04. Image-Image

5. Result - Warrior2

Ground Truth



Require modification

OKS, Cosine 유사도, keypoint 검출 만족

수정해야 할 자세 여럿

구체적인 Feedback

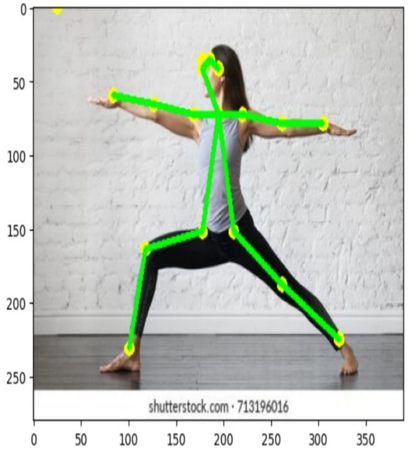
Feedback

```
oks : 0.8297477873942228
cosine simlarity : 0.8426145769506843
없는 키포인트 : 4
맞는 자세 : 2
Good! 고개가 측면을 바라보고 코와 귀가 수평에 가깝습니다.
양쪽 팔목 또는 팔꿈치 또는 어깨 키포인트 검출에 실패했습니다.
양쪽 팔목 또는 어깨 또는 엉덩이 키포인트 검출에 실패했습니다.
Try Again, 왼쪽 엉덩이, 무릎, 발목이 일직선상에 위치하게 하세요.
Good! 왼쪽 어깨, 엉덩이, 발목이 135도에 가깝습니다.
Try Again, 오른다리와 몸통, 무릎의 각도가 직각이 되게 하세요.
```

04. Image-Image

5. Result - Warrior2

Ground Truth



Good Example

```
oks : 0.8347966913508551
cosine simlarity : 0.8607301932270967
없는 키포인트 : 2
맞는 자세 : 4
Good! 고개가 측면을 바라보고 코와 귀가 수평에 가깝습니다.
Try Again, 양쪽 팔목, 팔꿈치, 어깨가 일직선상에 위치하게 하세요.
Good! 양쪽 팔과 몸통 사이 각도가 수직에 가깝습니다.
Good! 왼쪽 엉덩이, 무릎, 발목이 수평에 가깝습니다.
Good! 왼쪽 어깨, 엉덩이, 발목이 135도에 가깝습니다.
Try Again, 오른다리와 몸통, 무릎의 각도가 직각이 되게 하세요.
```

높은 평가지표
대부분의 자세가 기준에 부합(ground_truth 자세와 유사)

04. Video-Image

Process

1. 정답 이미지 선정

2. 이미지 keypoint 및 skeleton 검출

3. webcam 영상 skeleton 검출

4. Confidence score 실시간 제공

04. Video-Image

Video.mp4

```
gt_img = cv2.imread(gt_img_path)
gt_img, gt_kp = output_keypoints_with_lines_image(gt_img)
cv2.imshow("gt", gt_img)

# ============== using webcam ================
capture = cv2.VideoCapture(0) #카메라 정보 받아옴

while cv2.waitKey(1) <0:  #아무 키나 누르면 끝난다.
    hasFrame, frame = capture.read()

    if not hasFrame:
        cv2.waitKey()
        break

    frame, usr_kp = output_keypoints_with_lines_image(frame)

    # 코사인 유사도 계산
    gt_vectors, usr_vectors = get_valid_vectors(gt_kp, usr_kp)
    pose_pair_vectors_gt = get_pose_pair_vectors(gt_vectors, POSE_PAIRS_COCO)
    pose_pair_vectors_usr = get_pose_pair_vectors(usr_vectors, POSE_PAIRS_COCO)
    similarities = calculate_similarity_between_pose_pairs(pose_pair_vectors_gt, pose_pair_vectors_usr)
    cos_avg_similarity = calculate_average_similarity(similarities)

    # oks 계산
    KPTS_OKS_SIGMAS_COCO = torch.tensor([.26, .79, .72, .62, .79, .72, .62, 1.07, .87, .89, 1.07, .87, .89, .25, .25, 35, .35])/10.0
    resized_gt_kp = resize_keypoints(gt_img, gt_kp)
    resized_usr_kp = resize_keypoints(frame, usr_kp)
    resized_gt_kp2=resized_gt_kp.copy()
    resized_usr_kp2=resized_usr_kp.copy()

    ## oks 계산에 필요한 area 계산 ##
    gray = cv2.cvtColor(gt_img, cv2.COLOR_BGR2GRAY)
    ret, thresh = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)
    contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    for contour in contours:
        area = cv2.contourArea(contour)
    oks_coco = keypoint_similarity(resized_gt_kp2,
                                   resized_usr_kp2,
                                   sigmas=KPTS_OKS_SIGMAS_COCO,
                                   areas=area)

    # 실시간으로 유사도 표시
    # extracted keypoints minimum
    count_none = sum(1 for kp in resized_usr_kp if kp is None)
    if count_none >= 6:
        print("검출된 키포인트 수가 부족합니다. 다시 준비해오세요!")
    else:
        cv2.putText(frame, f"Cosine Similarity: {cos_avg_similarity:.2f}", (20, 20), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0), 2,
            lineType=cv2.LINE_AA)
        cv2.putText(frame, f"OKS: {oks_coco:.2f}", (300, 20), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2,
                    lineType=cv2.LINE_AA)
        cv2.imshow("Webcam", frame)

    if cv2.waitKey(1) & 0xFF == 27:
        break

capture.release()  #카메라 장치에서 받아온 메모리 해제
cv2.destroyAllWindows() #모든 윈도우 창 닫기
```

04. Video-Image

Result

조금 느리긴 하지만 **실시간**으로 스켈레톤 검출 및 **confidence score** 제공 가능!

인물이 정가운데 있지 않아 거리 기반인 **OKS값**은 image-image보다 작다

KUBIG
DATA SCIENCE & AI

05. Conclusion

05. Limitation

1. 관절 feature가 annotated된 Yoga 데이터 부재 및 데이터 수 부족

-> 더 많은 data set을 end-to-end 학습!

2. Openpose는 실시간이긴 하지만 반영 속도가 느리고 잘못된 keypoint도 존재

-> Rtmpose, Alphapose 등 더 최신화된 모델 공부 및 학습!

3. 이미지가 중앙에 없을 시 거리 기반 지표(OKS)의 정확도 하락

-> 이미지의 상대적 위치를 반영한 지표 계산!

4. 포즈 Feedback 시 임의의 기준으로 간단한 feedback만 진행

-> 보다 체계화된 기준으로 구체적인 feedback 제시 및 수치화!

05. Meaning

1. Openpose 모델의 구조 및 원리 이해

-> Pose estimation 과정에 대한 전반적인 내용 학습

2. Yoga Pose estimation, feedback 등 여러 task 수행

-> 실생활에 유용한 과제를 해결하는 과정에서 의미 있는 결과물 도출

3. 코딩 작업, 각종 토의 및 Contest 준비 과정

-> 팀원끼리 더 친해지고, cv 분야 프로젝트를 통한 실력 향상

KUBIG
DATA SCIENCE & AI

Thank You