



# Just DDance

Are you ready to dance?

*Likelion*

**Final\_Project**

# Director & Directing.

Just DDance는 프로젝트 계획, 개발, 배포를 마무리로 프로젝트를 진행했습니다.  
우리 팀원들을 소개 합니다.

## 팀원소개\_



김채현 팀장



김동영



서진원



주소리

# Contents.

목차\_

#1 프로젝트 목표 & 배경

#2 유사 프로그램

#3 서비스 개발 과정

#4 회고 및 리뷰



# Goal & Background.

모션 인식을 통한 안무 연습 및 피드백 제공 서비스

바쁜 일상 속 쉽고 간편하게 집에서 춤을 배우거나, 춤으로 운동하고 싶은 사람들에게 즉각적 자세 교정 피드백을 통해 도움을 주는 서비스 구현

## 목표 & 배경\_



COVID-19의 영향으로 실내 활동의 중요성이 대두 된 상황에서, 실내에서 즐길 수 있는 여가 활동의 다양성 확보를 위한 일환 중 하나로 운동을 도와주는 서비스에서 아이디어를 얻어 시작하게 되었습니다.

장기자랑 등을 위해 춤을 연습하는 경우, 일반적으로 안무 영상을 보며 따라하는 방식으로 연습을 하게 됩니다. 그러나 단순히 영상을 보며 따라하는 방법은 결국 안무를 모두 스스로 암기할 수밖에 없습니다. 또한 이후, 본인의 춤추는 영상을 보면서 역시 스스로 틀린 부분을 찾고 고치는 방식으로 이뤄지게 됩니다.

그래서 본 프로젝트는 원본 안무 영상과 본인의 안무 영상을 비교하며 틀린 부분을 직접 찾아야 하는 번거로움을 줄여주고, 특정 동작에서 어떤 부분을 어떻게 고치면 좋을지 피드백을 해주는 서비스를 기획했습니다.



# Similar Services.

현재 상용화되어 있는 유사 서비스에 대해서 소개합니다.

## Just Dance



센서를 손에 장착하고, 그 센서로만 동작을 감지하여 유사도를 통해 점수를 매기는 서비스

## RingFit



모션 감지 및 동작 감지 기기를 이용하여 운동을 게임처럼 할 수 있는 서비스

## 유사 서비스



# Similar Services.

현재 상용화되어 있는 유사 서비스에 대해서 소개합니다.

## Fitness Boxing

복싱을 이용해 플레이어를 운동시키는 트레이닝 게임이다  
모션 인식이 가능한 한 쌍의 조이콘의 특성을 살려, 양 손에 하나 씩 쥐고 복싱의 동작을 익히게 된다.



## Microsoft Kinect

Kinect라는 카메라 센서를 통해 게임을 할 수 있는 서비스



## 유사 서비스

# Overview of the Just DDance

모션 인식을 통한 안무 연습과 음악 추천 시스템 개발

개요\_

**DATASET**

**LANDMARKS**

**LIBRARIES**

**SKELETON**

**CRAWLING**

# DATASET

# LIBRARIES

유튜브 안무 영상 크롤링


 MediaPipe





# Libraries.

## 라이브러리\_

 **MediaPipe** : 두 명 이상의 사람이 등장하면 객체 검출 성능이 떨어짐  
=> 기본적으로 한 명이 나오는 사람에 대한 영상을 사용

한 명의 사람만 등장한다는 가정으로 프로젝트를 진행했다.



- openpose

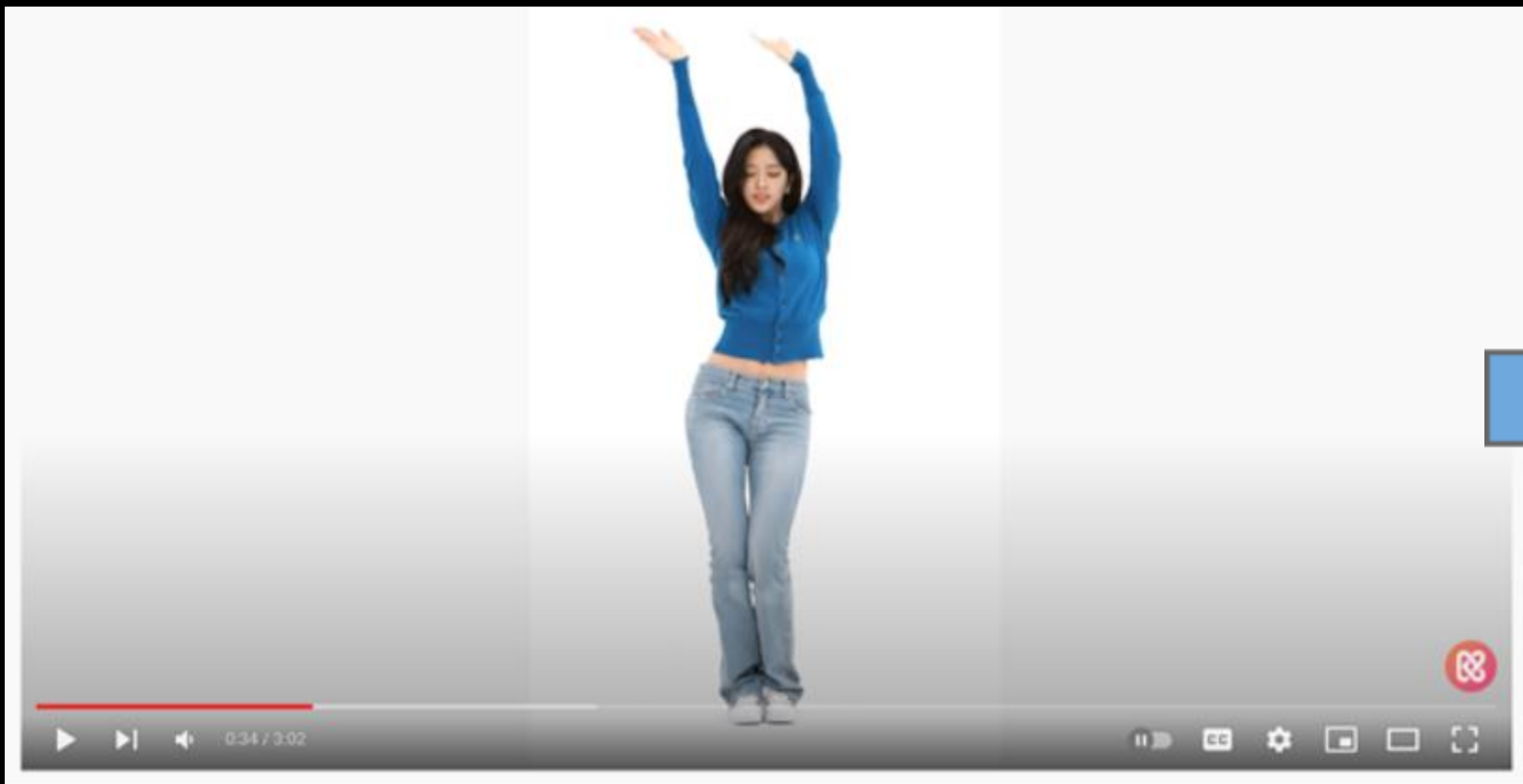
: 이용, 여러 사람의 포즈 추정, YOLO 모델 사용 처리하는 방법도 시도 -> 실패

# Collecting data.

## Crawling.

pytube 라이브러리를 이용한 유튜브 영상 크롤링

# 크롤링\_



```
def download_video(self):
    self.__save_dance_name()
    url = input(f"{self.__dance_name}의 안무 영상 링크: ")
    if not os.path.exists(self.__video_download_path): os.mkdir(self.__video_download_path)
    yt = pytube.YouTube(url).streams.filter(res="720p").first()
    yt.download(output_path=self.__video_download_path, filename=self.__dance_name+".mp4")
```



[안방1열 직캠  
4K] 르세라핌 김  
채원 FEARLESS  
(LE SSERAFIM ...



[안방1열 직캠  
4K] 아이브 안유  
진 LOVE DIVE  
(IVE YUJIN Fan...



[주간아 직캠]  
IVE YUJIN -  
LOVE DIVE (아이  
브 유진 - 러브 ...



[주간아 직캠] LE  
SSERAFIM KIM  
CHAEWON -  
FEARLESS (르...

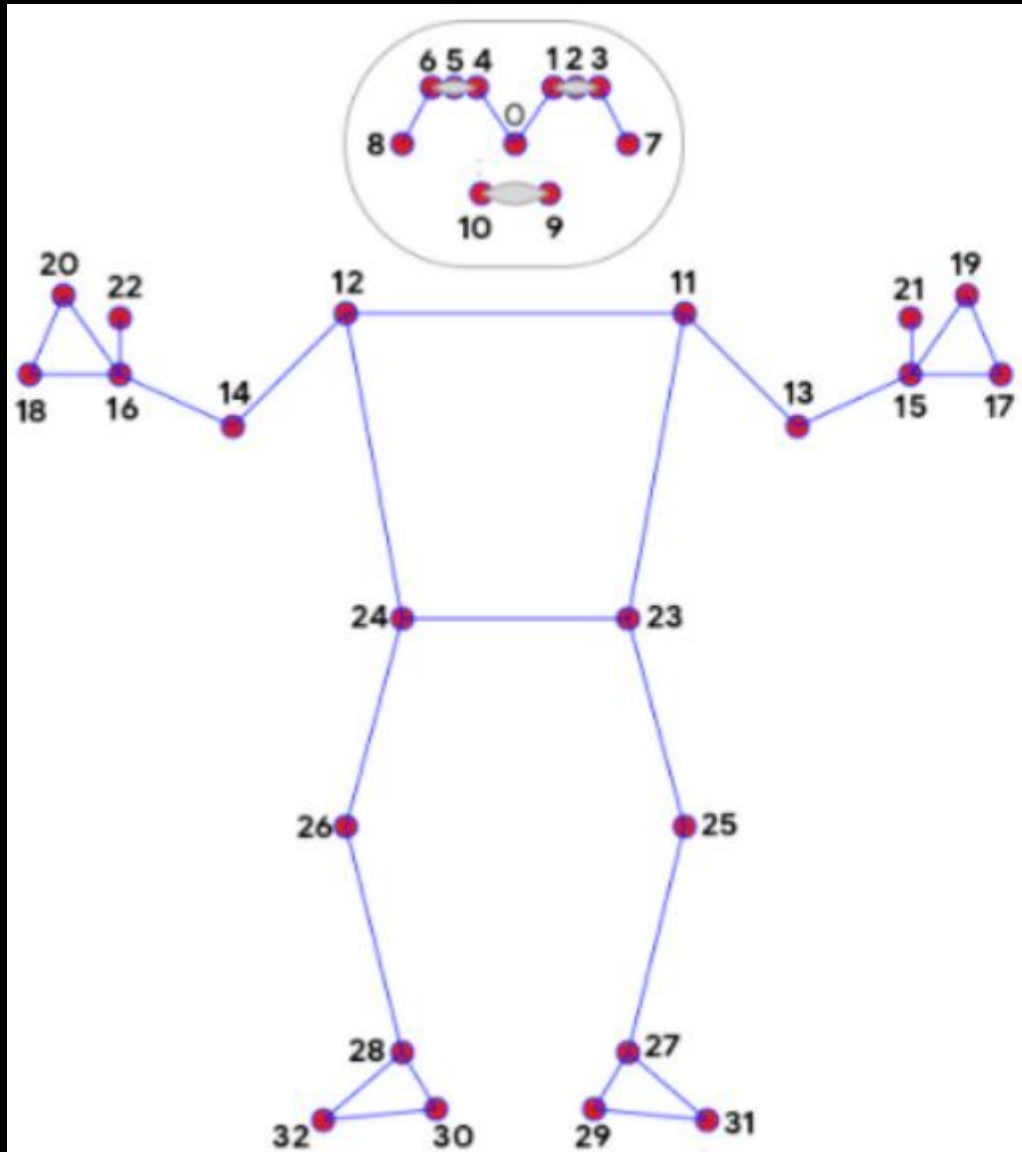


YMCA  
Dance.mp4

# Landmarks.

- 디텍션 모델을 활용해 영상의 프레임 단위로 키포인트를 추출해 저장
- Pose 키포인트 33개 존재, 영상의 각 프레임에 대해 모든 Pose 키포인트 수집 후 json 형식으로 저장

## 랜드마크\_



- |                    |                      |
|--------------------|----------------------|
| 0. nose            | 17. left_pinky       |
| 1. left_eye_inner  | 18. right_pinky      |
| 2. left_eye        | 19. left_index       |
| 3. left_eye_outer  | 20. right_index      |
| 4. right_eye_inner | 21. left_thumb       |
| 5. right_eye       | 22. right_thumb      |
| 6. right_eye_outer | 23. left_hip         |
| 7. left_ear        | 24. right_hip        |
| 8. right_ear       | 25. left_knee        |
| 9. mouth_left      | 26. right_knee       |
| 10. mouth_right    | 27. left_ankle       |
| 11. left_shoulder  | 28. right_ankle      |
| 12. right_shoulder | 29. left_heel        |
| 13. left_elbow     | 30. right_heel       |
| 14. right_elbow    | 31. left_foot_index  |
| 15. left_wrist     | 32. right_foot_index |
| 16. right_wrist    |                      |





# Landmarks.

- 디텍션 모델을 활용해 영상의 프레임 단위로 키포인트를 추출해 저장
- Pose 키포인트 33개 존재, 영상의 각 프레임에 대해 모든 Pose 키포인트 수집 후 json 형식으로 저장

랜드마크\_

```
try: keypoint_dict_pose.append({str(idx): [lmk.x, lmk.y, lmk.z] for idx, lmk in enumerate(results.pose_landmarks.landmark)})  
except: pass
```

```
with open(self.__keypoints_path+"/"+self.__dance_name+"_keypoints.json", "w") as keypoints:  
    json.dump(keypoint_dict_pose, keypoints)
```

# Landmarks.

- 디텍션 모델을 활용해 영상의 프레임 단위로 키포인트를 추출해 저장
- Pose 키포인트 33개 존재, 영상의 각 프레임에 대해 모든 Pose 키포인트 수집 후 json 형식으로 저장

랜드마크\_

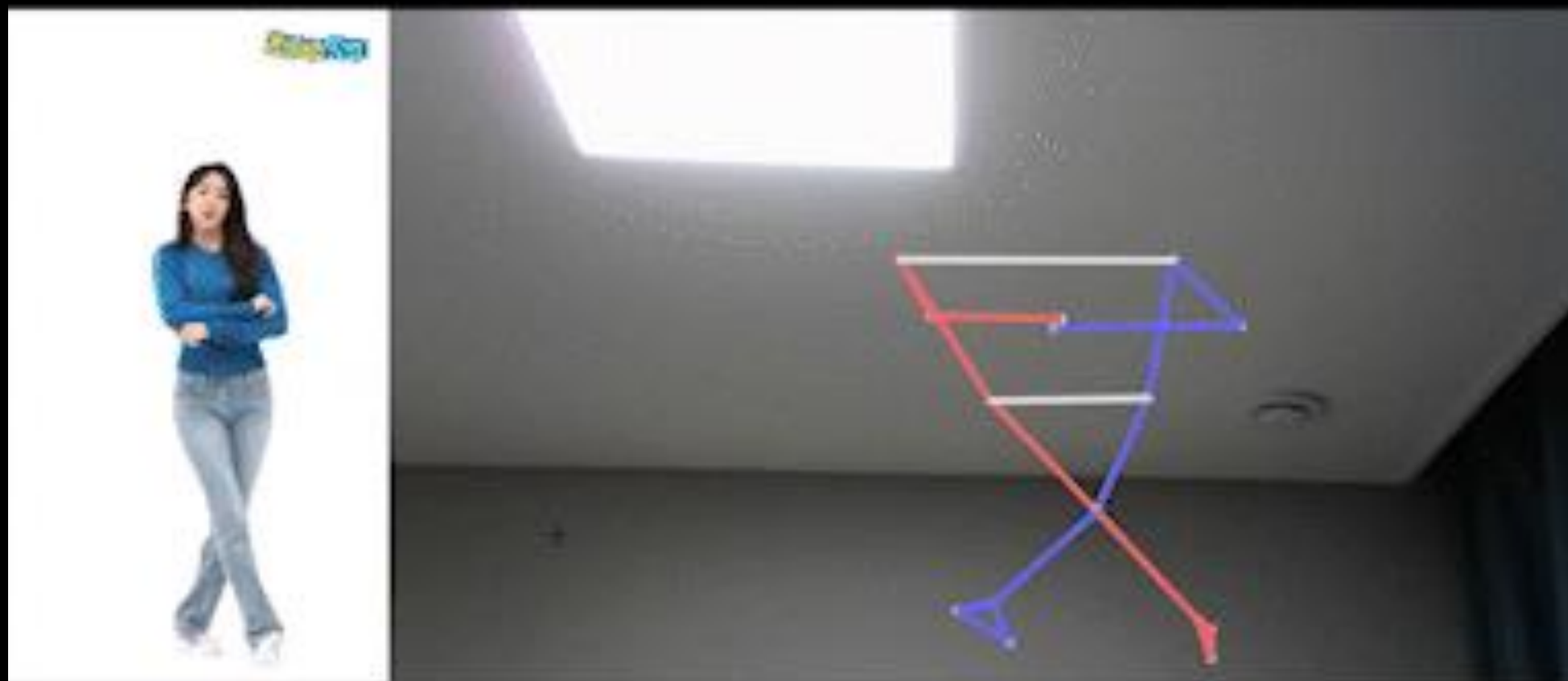
```
try:
    keypoint_dict_pose.append({str(idx):
        [lmk.x, lmk.y, lmk.z] for idx, lmk in enumerate(results.pose_landmarks.landmark)})
    keypoint_dict_left_hand.append({str(idx):
        [lmk.x, lmk.y, lmk.z] for idx, lmk in enumerate(results.left_hand_landmarks.landmark)})
    keypoint_dict_right_hand.append({str(idx):
        [lmk.x, lmk.y, lmk.z] for idx, lmk in enumerate(results.right_hand_landmarks.landmark)})
except:
    pass

keypoint_dict = {"pose": keypoint_dict_pose,
                 "left_hand": keypoint_dict_left_hand,
                 "right_hand": keypoint_dict_right_hand}
```

# Scaling & Output.

Skeleton.

스케일링 및 출력\_





# Scaling & Output.

## Skeleton.

## 스케일링 및 출력\_

```
try: user_input = {str(idx): [lmk.x, lmk.y, lmk.z] for idx, lmk in enumerate(user_results.pose_landmarks.landmark)}  
except: pass
```

```
try:  
    # get coors MARGIN  
    cors_margin = self.__get_margin([user_input["0"], user_input["23"], user_input["24"]], [dance_cors[dance_cors_frames][0], dance_cors[dance_cors_frames][23], dance_cors[dance_cors_frames][24]])  
    for pose_point in [11, 12, 13, 14, 15, 16, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32]:  
        x_cor_pose, y_cor_pose, z_cor_pose = \  
            int((dance_cors[dance_cors_frames][pose_point][0]+cors_margin[0])*user_image.shape[1]), \  
            int((dance_cors[dance_cors_frames][pose_point][1]+cors_margin[1])*user_image.shape[0]), \  
            int((dance_cors[dance_cors_frames][pose_point][2]+cors_margin[2])*1000)  
  
        cv2.circle(user_image, (x_cor_pose, y_cor_pose), 8, (244, 244, 244), cv2.FILLED)  
        skeletons[pose_point] = (x_cor_pose, y_cor_pose)  
    self.__draw_skeleton(user_image, skeletons)  
    dance_cors_frames +=1  
except: pass
```

# Scaling & Output.

## Skeleton.

## 스케일링 및 출력\_

```
def __draw_skeleton(self, image, skeleton):
    # 오른쪽 스켈레톤 (붉은색)
    cv2.line(image, skeleton[12], skeleton[14], (102, 102, 255), thickness=7, lineType=cv2.LINE_AA, shift=None) # 오/어깨 -> 오/팔꿈치
    cv2.line(image, skeleton[14], skeleton[16], (102, 102, 255), thickness=7, lineType=cv2.LINE_AA, shift=None) # 오/팔꿈치 -> 오/손목
    cv2.line(image, skeleton[12], skeleton[24], (102, 102, 255), thickness=7, lineType=cv2.LINE_AA, shift=None) # 오/어깨 -> 오/엉덩이
    cv2.line(image, skeleton[24], skeleton[26], (102, 102, 255), thickness=7, lineType=cv2.LINE_AA, shift=None) # 오/엉덩이 -> 오/무릎
    cv2.line(image, skeleton[26], skeleton[28], (102, 102, 255), thickness=7, lineType=cv2.LINE_AA, shift=None) # 오/무릎 -> 오/발목
    cv2.line(image, skeleton[28], skeleton[30], (102, 102, 255), thickness=7, lineType=cv2.LINE_AA, shift=None) # 오/발목 -> 오/뒷꿈치
    cv2.line(image, skeleton[30], skeleton[32], (102, 102, 255), thickness=7, lineType=cv2.LINE_AA, shift=None) # 오른발
    cv2.line(image, skeleton[28], skeleton[32], (102, 102, 255), thickness=7, lineType=cv2.LINE_AA, shift=None) # 오른발
    # 왼쪽 스켈레톤 (푸른색)
    cv2.line(image, skeleton[11], skeleton[13], (255, 102, 102), thickness=7, lineType=cv2.LINE_AA, shift=None) # 왼/어깨 -> 왼/팔꿈치
    cv2.line(image, skeleton[13], skeleton[15], (255, 102, 102), thickness=7, lineType=cv2.LINE_AA, shift=None) # 왼/팔꿈치 -> 왼/손목
    cv2.line(image, skeleton[11], skeleton[23], (255, 102, 102), thickness=7, lineType=cv2.LINE_AA, shift=None) # 왼/어깨 -> 왼/엉덩이
    cv2.line(image, skeleton[23], skeleton[25], (255, 102, 102), thickness=7, lineType=cv2.LINE_AA, shift=None) # 왼/엉덩이 -> 왼/무릎
    cv2.line(image, skeleton[25], skeleton[27], (255, 102, 102), thickness=7, lineType=cv2.LINE_AA, shift=None) # 왼/무릎 -> 왼/발목
    cv2.line(image, skeleton[27], skeleton[29], (255, 102, 102), thickness=7, lineType=cv2.LINE_AA, shift=None) # 왼/발목 -> 왼/뒷꿈치
    cv2.line(image, skeleton[29], skeleton[31], (255, 102, 102), thickness=7, lineType=cv2.LINE_AA, shift=None) # 왼발
    cv2.line(image, skeleton[27], skeleton[31], (255, 102, 102), thickness=7, lineType=cv2.LINE_AA, shift=None) # 왼발
    # 상체 스켈레톤 (회색)
    cv2.line(image, skeleton[11], skeleton[12], (224, 224, 224), thickness=5, lineType=cv2.LINE_AA, shift=None)
    cv2.line(image, skeleton[23], skeleton[24], (224, 224, 224), thickness=5, lineType=cv2.LINE_AA, shift=None)
```

# Scoring.

- 실제 안무연습 영상과 사용자가 춘 안무간 유사도를 측정하여 Scoring
- 유사도는 3차원 벡터(x, y, z)간의 L2 Norm을 이용

## 스코어링\_

### ▶ 적용한 Scoring 공식 (K값으로 난이도 조절)

$$Acc = \frac{K}{L2Norm + K}$$

### ▶ 구현 코드

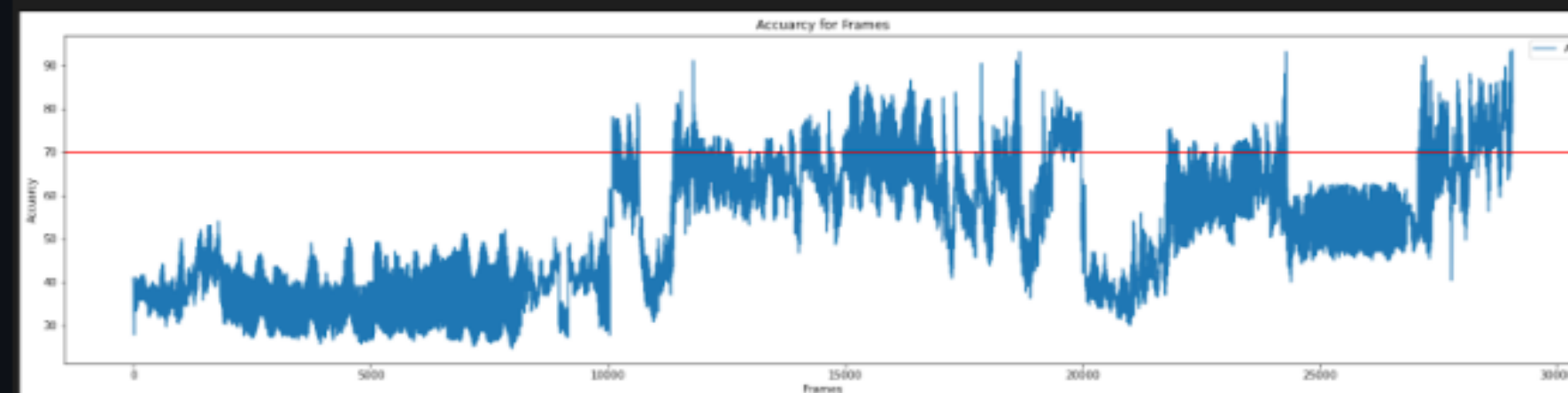
```
# L2 Norm
acc_per_frame.append(
    np.round(
        self.__const_k /
        (np.linalg.norm(
            [(x_cor_pose/user_image.shape[1]-cors_margin[0])-user_input[str(pose_point)][0],
            (y_cor_pose/user_image.shape[0]-cors_margin[1])-user_input[str(pose_point)][1],
            (z_cor_pose/1000-cors_margin[2])-user_input[str(pose_point)][2]]
        )
        + self.__const_k),
        2)
)
acc = np.mean(acc_per_frame)*100
```

### ▶ (필요시) 부위별 가중치 입력

$$ACC = \frac{K}{L2_{pose} + 0.3 * L2_{hand} + 0.1 * L2_{face} + K}$$

### ▶ Scoring 결과

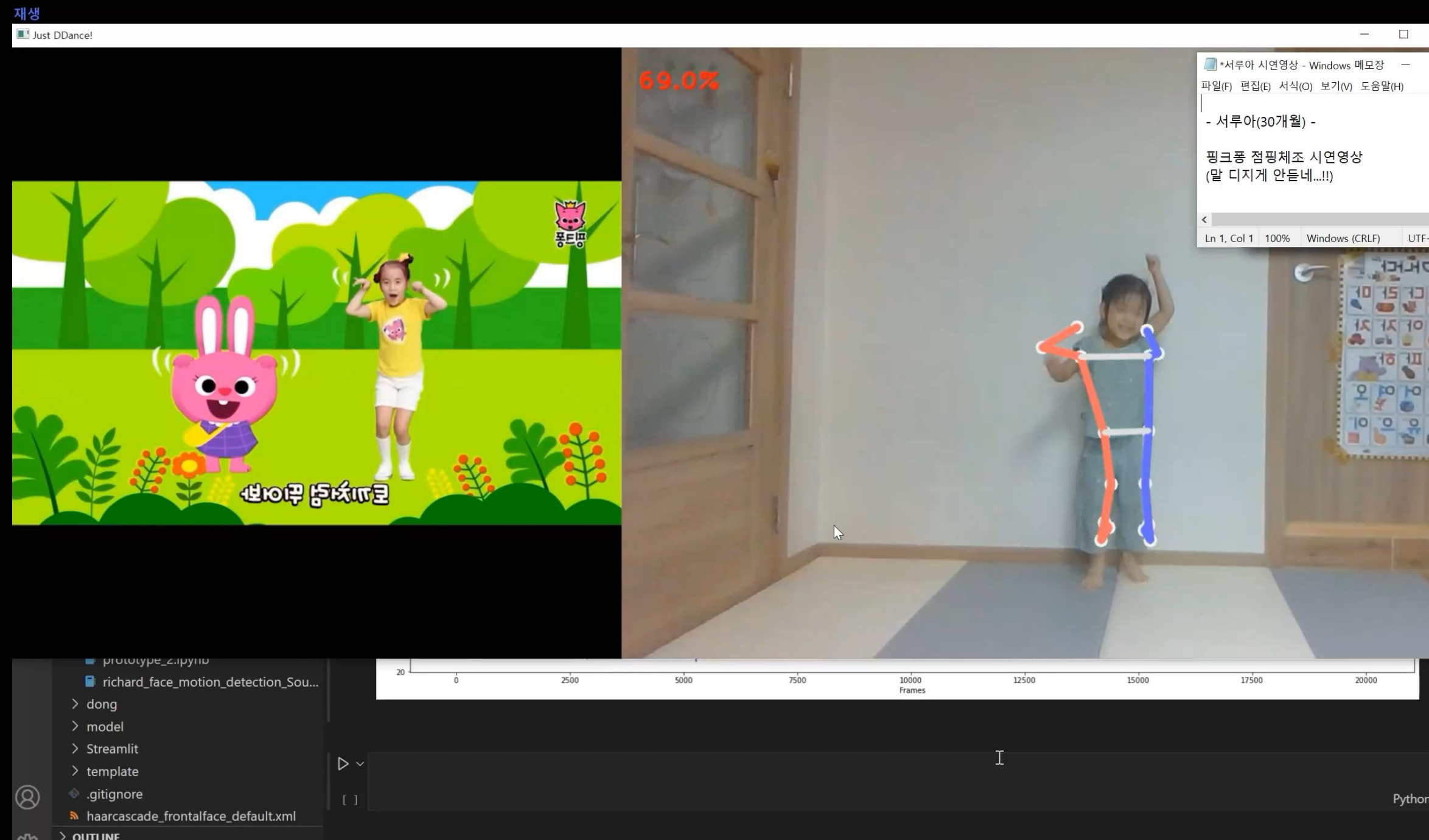
Max Acc: 93.5 Min Acc: 24.625 Avg. Acc: 51.56302269113049





# Testing.

## 테스트영상\_



# Testing.

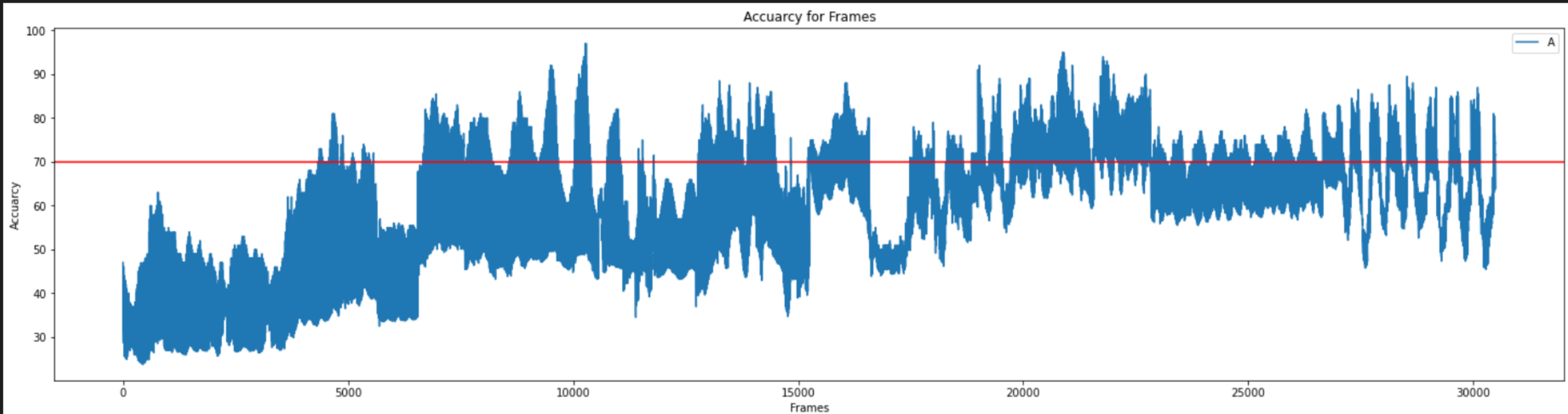
## 테스트영상\_

```
jd.print_dance_data()
```

✓ 0.2s

Python

Max Acc: 97.0    Min Acc: 23.812500000000004    Avg. Acc: 58.55732479958649



# Testing.

테스트영상\_



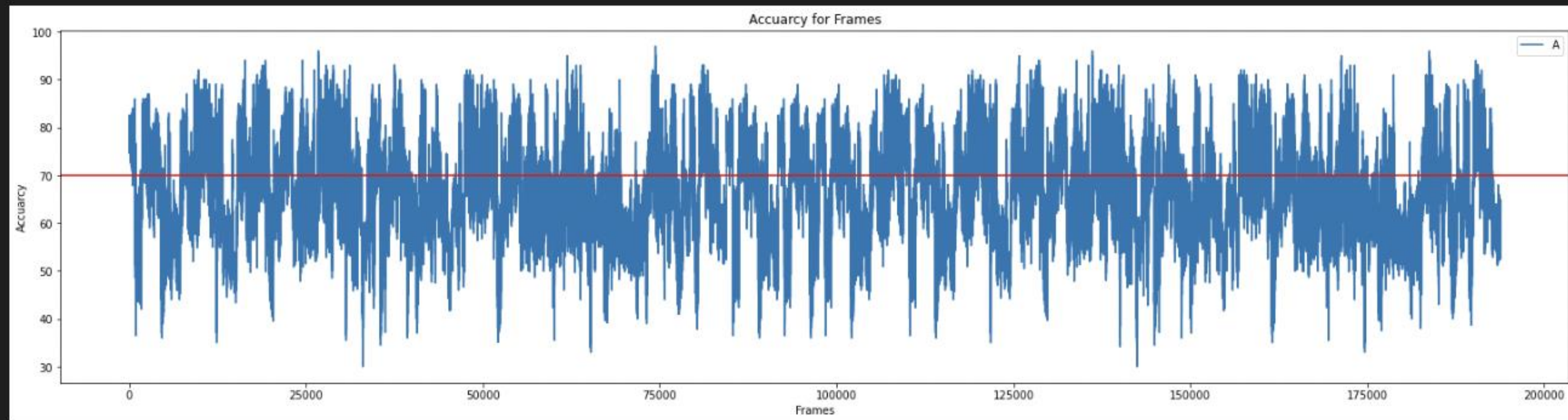


# Testing.

테스트영상\_

... Max Acc: 97.0 Min Acc: 30.0 Avg. Acc: 66.38342090437555

</>



# Testing.

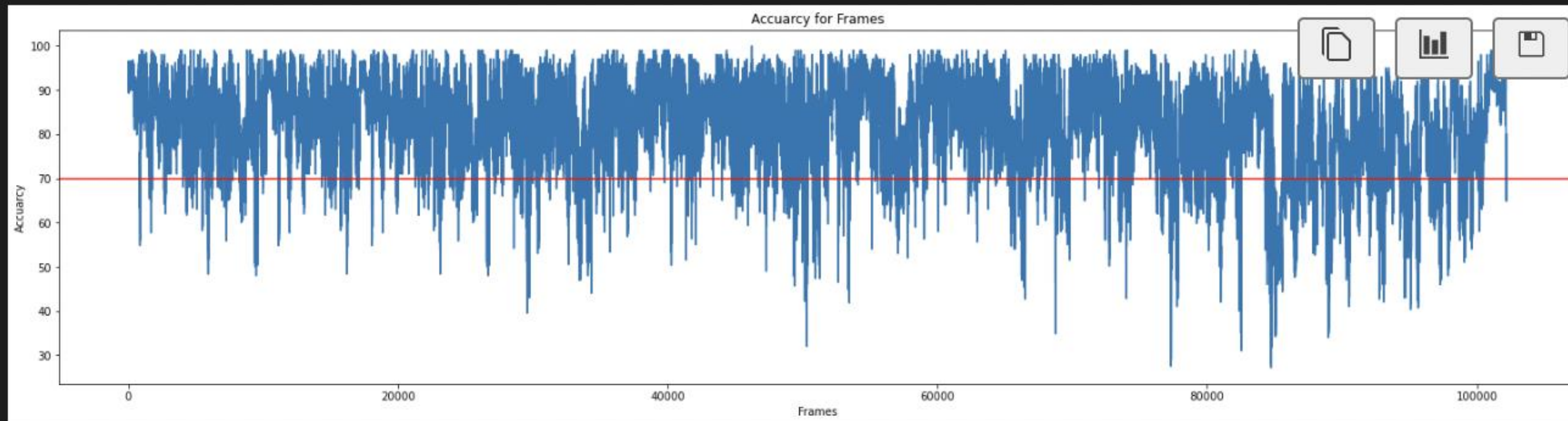
## 테스트영상\_



# Testing.

## 테스트영상\_

Max Acc: 100.0 Min Acc: 27.166666666666668 Avg. Acc: 81.47673553361903



# Retrospective.

## 회고 및 리뷰\_

구 분	한 계	개선 방안	개선 정도
프레임드랍	mediapipe 모델이 프레임을 강제로 드랍	프레임수를 직접 계산하여 강제로 프레임을 유지	일부 개선 (다만, 영상의 재생속도가 떨어짐)
스케일링	스켈레톤이 인식한 사람을 따라 다니지만, 스케일링이 되지 않음		
복수인원 등장	원하는 인물 외에 다른 인물을 인식하기도 함	Openpose 또는 YOLO와 mediapipe 함께 사용	적용하지 못함 (다른문제 야기..)
추천시스템 구현 * 사용자 안무를 입력받아 가장 유사한 안무 노래를 추천해주는 서비스	춤 유형별 안무 데이터 확보, 데이터에 맞는 딥러닝 모델을 구현하지 못함	사전에 구현하려는 기능을 구체화하고 가능성 등을 충분히 감안하여 기획 필요	포기
실시간 피드백 * 정확도를 높이기 위한 자세 개선 기능	구현하지 못함		시간 여건상 포기



# Improving Frame drop.

## 프레임드랍개선\_

```
pTime = 0
FPS = 댄스 영상.get(cv2.CAP_PROP_FPS)

while 웹캠이 실행되는 동안:
    cTime = time.time() - pTime

    if cTime > 1./FPS:
        pTime = time.time()
        출력
```

# Improving Scaling.

## 스케일링 개선\_

SUDO:

```
user_ratio = Distance Two Points
```

```
dance_ratio = Distance Two Points
```

```
ratio = user_ratio/dance_ratio
```

```
x_cor_pose, y_cor_pose = x_cor_pose*ratio, y_cor_pose*ratio
```



# Thank you!

감사합니다.

Just DDance

 LIKELION AI SCHOOL