LAB: Exercise Posture Assistance System

Date: 2022.06.16

Name: ChangMin An JiWoo Yang

Github: Link

Demo Video: Link

I. Introduction

In this LAB, we start a project that tells us how to correct our posture in real time when we exercise at the gym. This program is basically limited to the "lat-pull-down" movement. These days, as interest in health increases due to COVID-19, interest in health increases, and people who exercise alone also increase. However, if we exercise alone, it is difficult to recognize whether you are exercising in an accurate posture, and as a result, a problem that is prone to muscle imbalance is found. To solve this problem, we try to create system that identifies each joint of a person and measures the balance according to both slopes of the upper body joint to give feedback on the balance between the two forces. The tutorial is run by visual studio code(VS code), loading web cam or video source, and processing images in real time using OpenCV.

II. Requirement

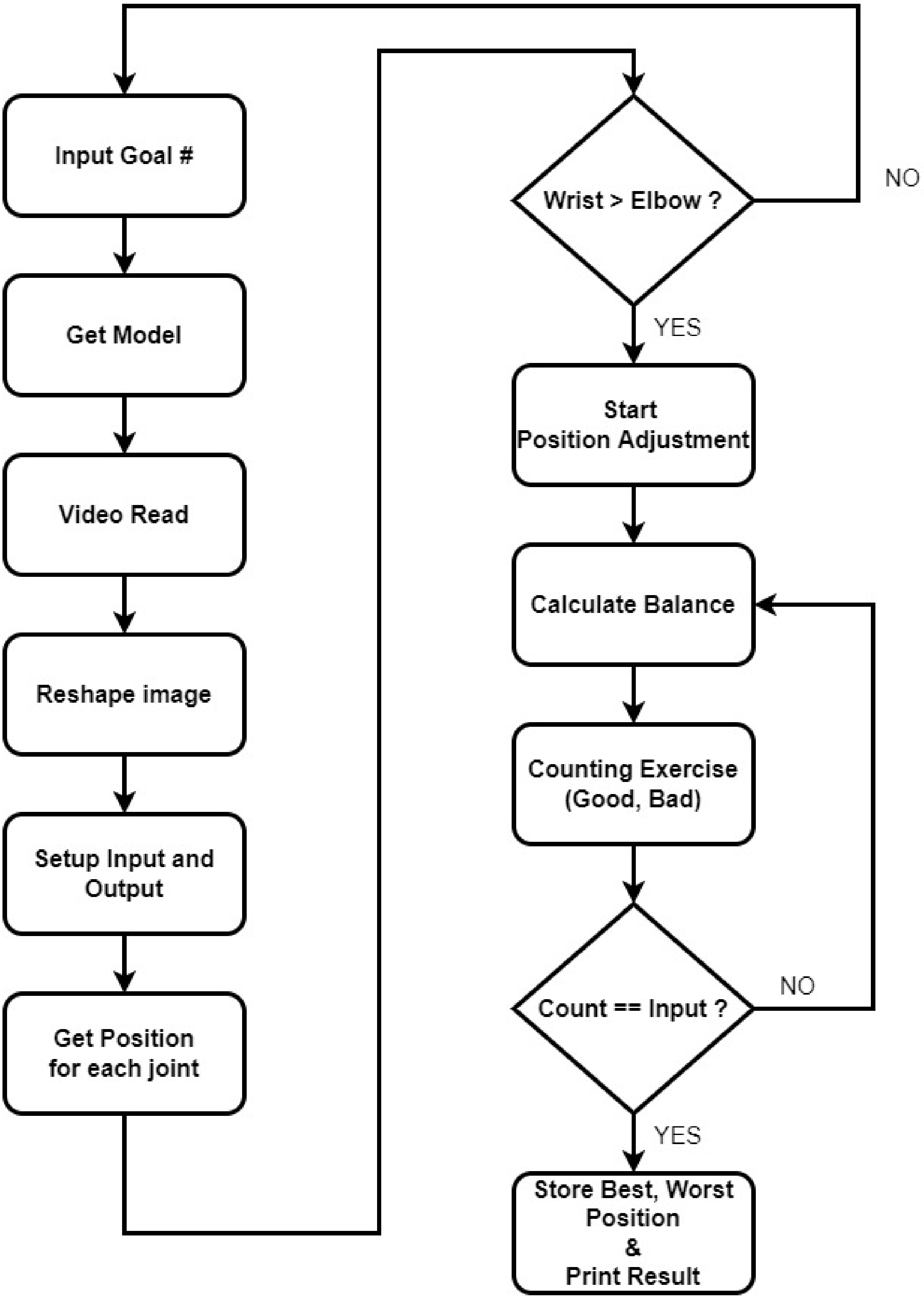
Hardware

- Logitech C922 pro Webcam
- Lat Pull Down Machine

Software

- Python 3.9.12
- Tensorflow 2.9.1
- numpy 1.21.5
- OpenCV 4.5.5
- MoveNet

III. Flow Chart



IV. Procedure

1. Setup

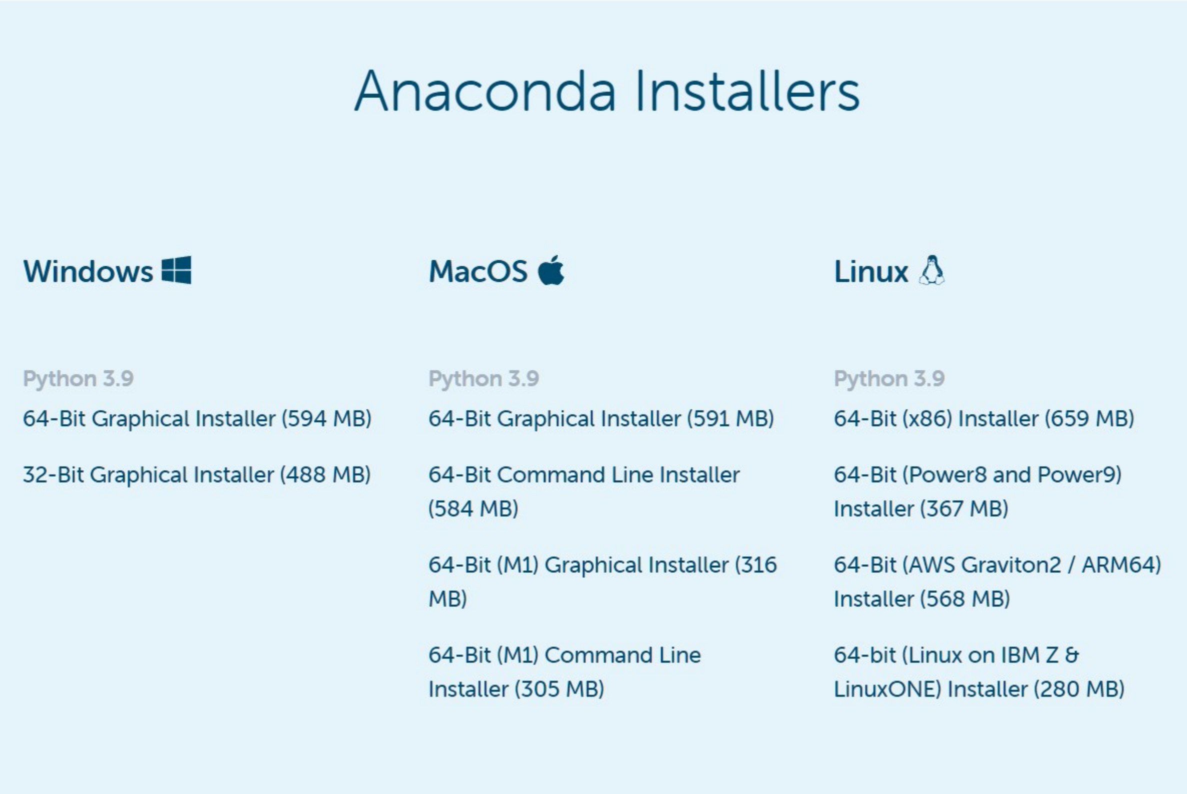
First, installation is carried out using Anaconda Prompt to build the environment. It is important to install something suitable for each version using anaconda to build it to enable image processing.

2. Installation

2-1. Install Anaconda

Anaconda : Python and libraries package installer.

Click here [Download the installer on window](#) to Windows 64-Bit Graphical Installer



Anaconda Installers

Windows

Python 3.9

64-Bit Graphical Installer (594 MB)

32-Bit Graphical Installer (488 MB)

MacOS

Python 3.9

64-Bit Graphical Installer (591 MB)

64-Bit Command Line Installer (584 MB)

64-Bit (M1) Graphical Installer (316 MB)

64-Bit (M1) Command Line Installer (305 MB)

Linux

Python 3.9

64-Bit (x86) Installer (659 MB)

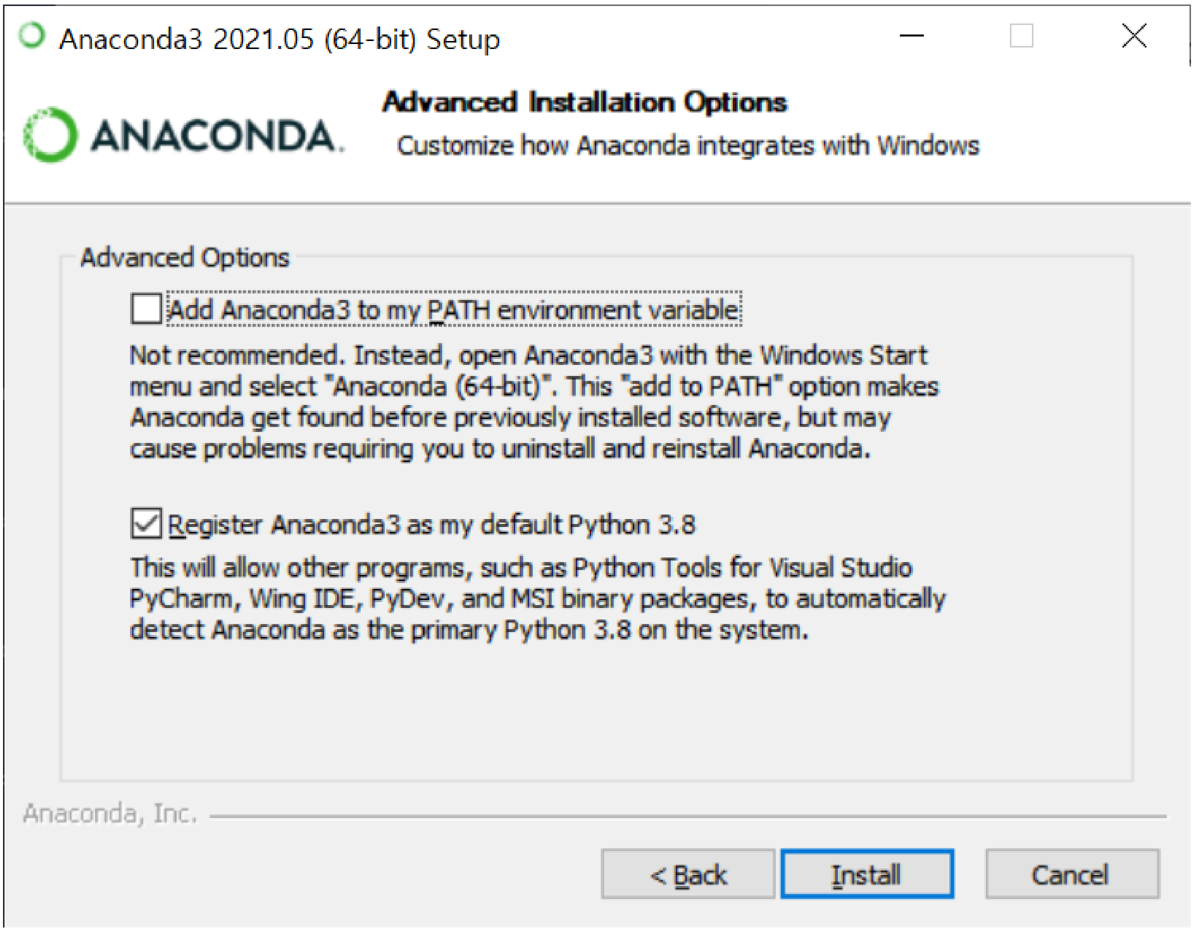
64-Bit (Power8 and Power9) Installer (367 MB)

64-Bit (AWS Graviton2 / ARM64) Installer (568 MB)

64-bit (Linux on IBM Z & LinuxONE) Installer (280 MB)

Follow the following steps

- Double click the installer to launch.
- Select an install for "Just Me"(recommended)
- Select a destination folder to install Anaconda and click the Next button.
- Do NOT add Anaconda to my PATH environment variable
- Check to register Anaconda as your default Python.



Anaconda3 2021.05 (64-bit) Setup

Advanced Installation Options

Customize how Anaconda integrates with Windows

Advanced Options

☐ Add Anaconda3 to my PATH environment variable

Not recommended. Instead, open Anaconda3 with the Windows Start menu and select "Anaconda (64-bit)". This "add to PATH" option makes Anaconda get found before previously installed software, but may cause problems requiring you to uninstall and reinstall Anaconda.

☑ Register Anaconda3 as my default Python 3.8

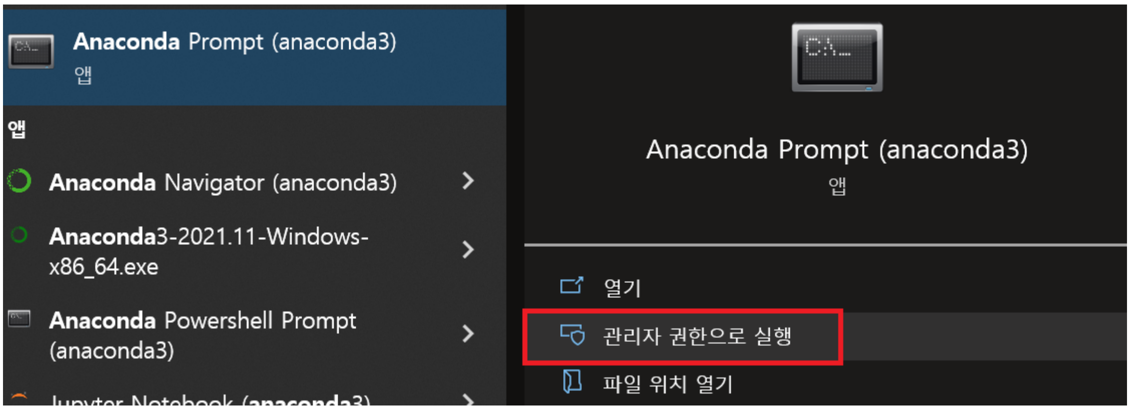
This will allow other programs, such as Python Tools for Visual Studio PyCharm, Wing IDE, PyDev, and MSI binary packages, to automatically detect Anaconda as the primary Python 3.8 on the system.

2-2. Install Python

Python 3.9

Python is already installed by installing Anaconda. But, we will make a virtual environment for a specific Python version.

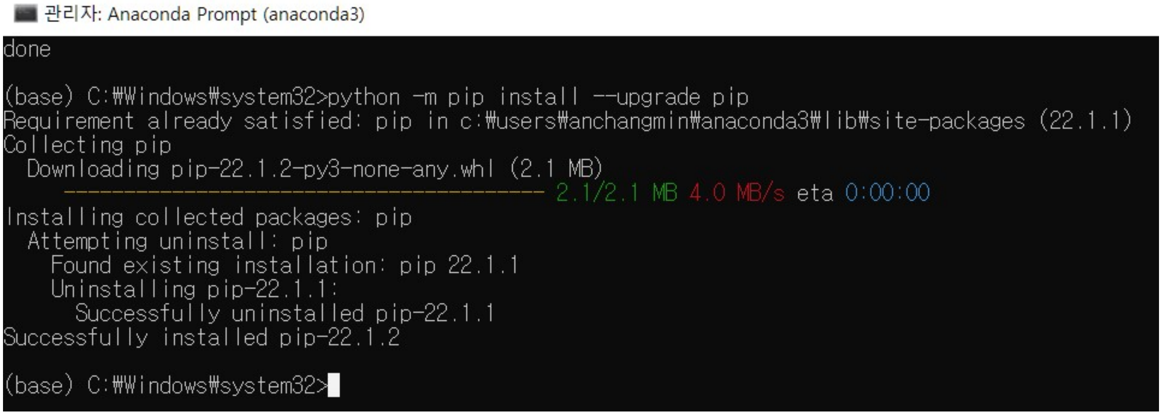
- Open Anaconda Prompt(admin mode)



Anaconda Prompt (anaconda3)
앱
앱
Anaconda Navigator (anaconda3)
Anaconda3-2021.11-Windows-x86_64.exe
Anaconda Powershell Prompt (anaconda3)
Jupyter Notebook (anaconda3)
Anaconda Prompt (anaconda3)
앱
열기
관리자 권한으로 실행
파일 위치 열기

- First, update **conda** and **pip** (If the message "Press Y or N" appears, press Y and Enter key)

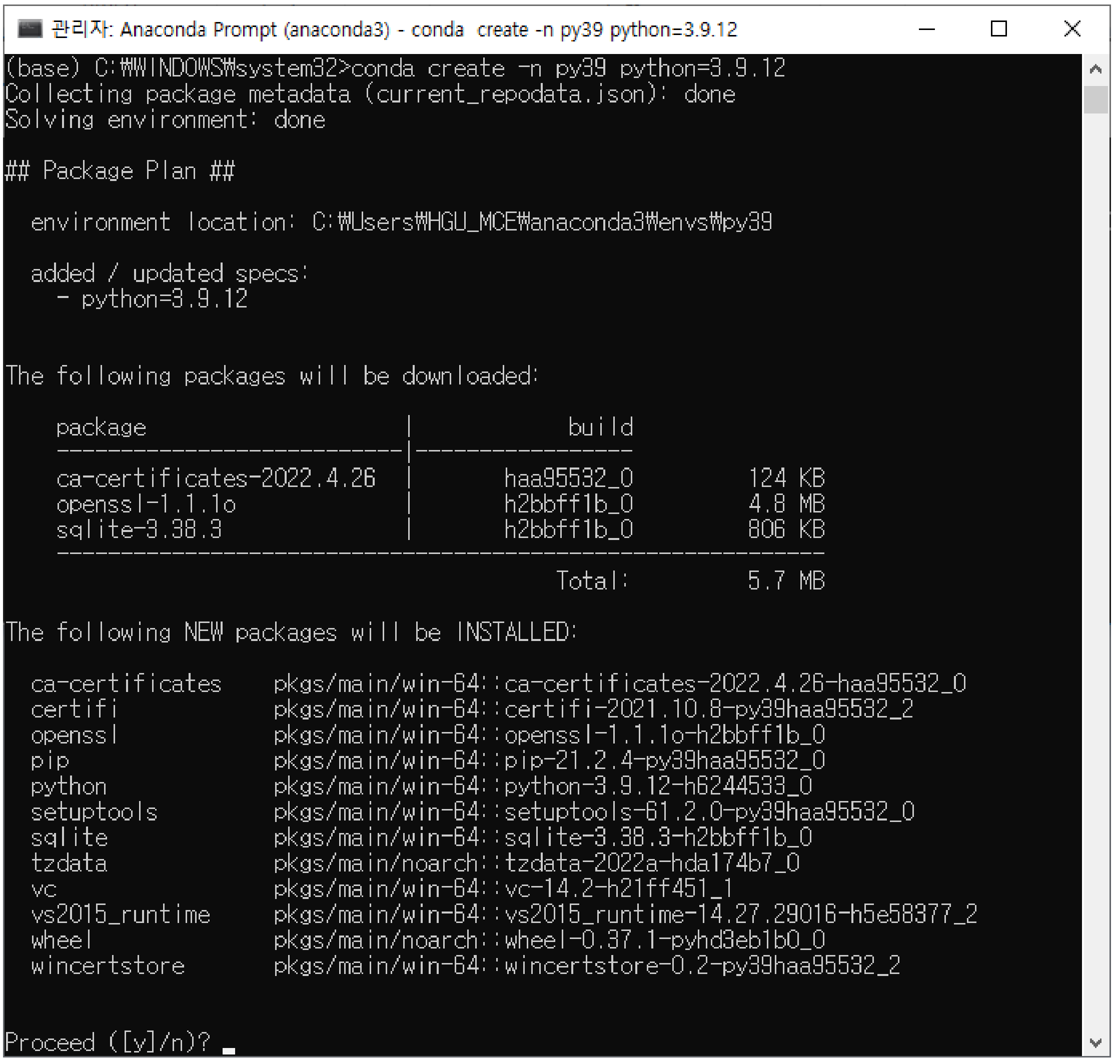
```
# Conda Update
conda update -n base -c defaults conda

# pip Update
python -m pip install --upgrade pip
```



```
done

(base) C:₩Windows₩system32>python -m pip install --upgrade pip
Requirement already satisfied: pip in c:₩users₩anchangmin₩anaconda3₩lib₩site-packages (22.1.1)
Collecting pip
  Downloading pip-22.1.2-py3-none-any.whl (2.1 MB)
     ---------------------------------------- 2.1/2.1 MB 4.0 MB/s eta 0:00:00
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 22.1.1
    Uninstalling pip-22.1.1:
      Successfully uninstalled pip-22.1.1
Successfully installed pip-22.1.2

(base) C:₩Windows₩system32>
```

- Then, Create virtual environment for Python 3.9, Name the $ENV as `py39`. If you are in base, enter `conda activate py39`

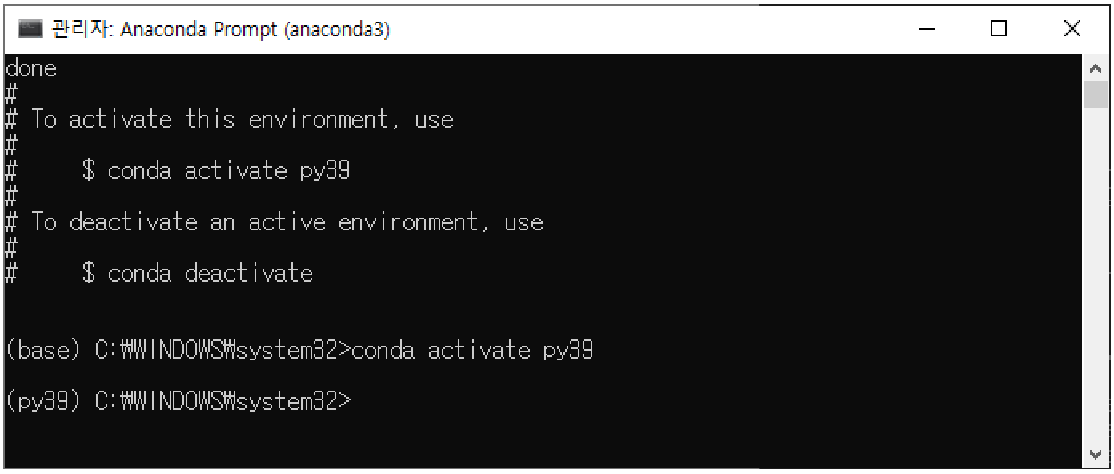
```
# Install Python 3.9
conda create -n py39 python=3.9.12
```


관리자: Anaconda Prompt (anaconda3) - conda create -n py39 python=3.9.12

```
(base) C:₩WINDOWS₩system32>conda create -n py39 python=3.9.12
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:₩Users₩HGU_MCE₩anaconda3₩envs₩py39

  added / updated specs:
    - python=3.9.12


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    ca-certificates-2022.4.26  |       haa95532_0         124 KB
    openssl-1.1.1o             |       h2bbff1b_0         4.8 MB
    sqlite-3.38.3              |       h2bbff1b_0         806 KB
    ------------------------------------------------------------
                                           Total:         5.7 MB

The following NEW packages will be INSTALLED:

  ca-certificates    pkgs/main/win-64::ca-certificates-2022.4.26-haa95532_0
  certifi            pkgs/main/win-64::certifi-2021.10.8-py39haa95532_2
  openssl            pkgs/main/win-64::openssl-1.1.1o-h2bbff1b_0
  pip                pkgs/main/win-64::pip-21.2.4-py39haa95532_0
  python             pkgs/main/win-64::python-3.9.12-h6244533_0
  setuptools         pkgs/main/win-64::setuptools-61.2.0-py39haa95532_0
  sqlite             pkgs/main/win-64::sqlite-3.38.3-h2bbff1b_0
  tzdata             pkgs/main/noarch::tzdata-2022a-hda174b7_0
  vc                 pkgs/main/win-64::vc-14.2-h21ff451_1
  vs2015_runtime     pkgs/main/win-64::vs2015_runtime-14.27.29016-h5e58377_2
  wheel              pkgs/main/noarch::wheel-0.37.1-pyhd3eb1b0_0
  wincertstore       pkgs/main/win-64::wincertstore-0.2-py39haa95532_2


Proceed ([y]/n)?
```

- After installation, activate the newly created environment

```
# Activate py39
conda activate py39
```



```
done
#
# To activate this environment, use
#
#     $ conda activate py39
#
# To deactivate an active environment, use
#
#     $ conda deactivate

(base) C:₩WINDOWS₩system32>conda activate py39

(py39) C:₩WINDOWS₩system32>
```

2-3. Install Libs

Install Numpy, OpenCV, Jupyter (opencv-python MUST use 4.5.5 NOT 4.6.0)

```
conda activate py39
conda install -c anaconda seaborn jupyter
pip install opencv-python==4.5.5.64
```

2-4. Install Visual Studio Code

Follow: How to Install VS Code

Also, read about

- How to program Python in VS Code

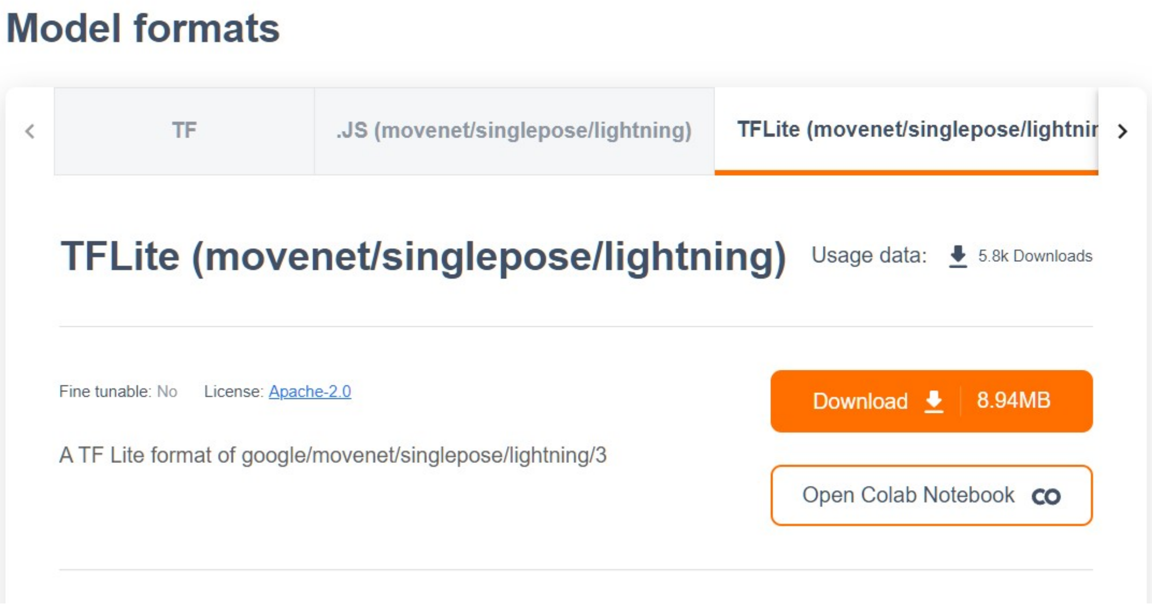
2-5. Install TensorFlow

- **TensorFlow** - DL library, developed by Google.

```
conda activate py39
pip install tensorflow==2.9.1
```

3. Download

- Download MoveNet model: TFLite model link **(Must download in local folder (include main.py))**



Model formats

‹
TF
.JS (movenet/singlepose/lightning)
TFLite (movenet/singlepose/lightnir
›

TFLite (movenet/singlepose/lightning)

Usage data: 5.8k Downloads

Fine tunable: No License: Apache-2.0

A TF Lite format of google/movenet/singlepose/lightning/3

Download 8.94MB
Open Colab Notebook

4. library you need

```
import tensorflow as tf
import numpy as np
import cv2 as cv
from cv2 import *
from cv2.cv2 import *
from tkinter import *
```

5. Global Variable

5-1. Main Variable

Definition Body Parts: The output of the deep learning model we use is location information for 17 joints. The position of each joint has an order. The order is as follows.

[nose, left eye, right eye, left ear, right ear, left shoulder, right shoulder, left elbow, right elbow, left wrist, right wrist, left hip, right hip, left knee, right knee, left ankle, right ankle]

Since we use the positions of the shoulders, elbows, and wrists on both sides in this application, we defined them as follows.

Definition of body edges: Each joint is tied together to draw a skeleton model.

Thresholding

- **CONFIDENCE_THRESHOLD:** Acceptable confidence for each joint position
- **CORRECT_RECOGNIGION:**
- **START_THRESHOLD:**
- **START_COUNT_THRESH:** Define the minimum number of frames for the correct posture

User input: This is a definition for setting the number of user's target exercise in the system.

Flag

- **system_Flag:** It is a flag that controls the start of processing.

- **start_Flag:** It is a flag that controls the start of processing for counting the number of workouts.
- **finish_Flag:** It is a flag indicating that the exercise is over.
- **tk_Flag:** It is a flag that determines whether or not to receive new input from the user.
- **up_down_Flag:** It is a flag necessary to count the exercise.

For counting: Definitions for the number of good and bad exercises, stack calculations, average values, conditions for starting an exercise, and frame values.

For Balance: Definitions for the balance value, the minimum balance value, and the maximum balance value.

```
#===============================================#
#                Global Variable                #
#===============================================#

# Color Definition (BGR)
WHITE               = (255, 255, 255)
RED                 = (  0,   0, 255)
GREEN               = (  0, 255,   0)
PINK                = (184,  53, 255)
YELLOW              = (  0, 255, 255)
BLUE                = (255,   0,   0)
BLACK               = (  0,   0,   0)
PURPLE              = (255, 102, 102)

# Font Definition
USER_FONT       = FONT_HERSHEY_DUPLEX
fontScale       = 1
fontThickness   = 2

# Definition Body Parts
LEFT_SHOULDER       = 5
RIGHT_SHOULDER      = 6
LEFT_ELBOW          = 7
RIGHT_ELBOW         = 8
LEFT_WRIST          = 9
RIGHT_WRIST         = 10

# Definition of body edges
EDGES = {
    (LEFT_SHOULDER,LEFT_ELBOW): 'm',
    (LEFT_ELBOW,LEFT_WRIST): 'm',
    (RIGHT_SHOULDER,RIGHT_ELBOW): 'c',
    (RIGHT_ELBOW,RIGHT_WRIST): 'c',
    (LEFT_SHOULDER,RIGHT_SHOULDER): 'y',
}

# Thresholding
CONFIDENCE_THRESHOLD    = 0.2   # For minimum confidence of output
CORRECT_RECOGNIGION     = 5     #

START_THRESHOLD         = 0.3   # For value of good pose
START_COUNT_THRESH      = 30    # For start counting

# User input
inputCount      = 11 # For user input
```

```
# Flag
system_Flag     = False                        # For start system
start_Flag      = False                        # For start counting process
finish_Flag     = False                        # For finish workout
tk_Flag         = False                        # For new user input
up_down_Flag    = [False, False, False, False] # 0: down_Flag, 1: up_Flag, 2:
left_down_Flag, 3: right_down_Flag

# For counting
counting_List   = [0, 0, 0, 0, 0.0, 0.0, 0] # 0: Good count, 1: bad count, 2:
left_E_ystack, 3: right_E_ystack, 4: left_E_avg, 5: right_E_avg 6: count_frame
start_Count     = 0                         # For starting setting

# 0:set Count, 1: startCount, 2: userSet
offset_Text = ""

# For Balance
balance_List = [0.0, 10.0, 0.0] # 0: balance, 1: good, 2: bad


# Frame count
frame_Num           = 0         # for processing
position_FrameList  = [0, 0]    # 0: worst / 1: Best

# Video naming
Video = "DEMO.mp4"
```

6. Definition Function

6-1. Processing

Get position of 17 each joint

- Since the position of the joint obtained from the model is relative to the frame, the frame is entered as an input to the function, and the keypoint that has information about the joint points and the minimum confidence are input. As output, 'shape' with the positions of all joints for the frame and 'posebuf' with position information for joints exceeding the minimum confidence can be obtained.

```
def Get_Shape_PoseBuf(frame, keypoints, edge, confidence_threshold):
   # Rendering
    y, x, c = frame.shape
    _shaped = np.squeeze(np.multiply(keypoints, [y,x,1]))

    _poseBuf = []
    for edge, color in EDGES.items():

        p1, p2 = edge
        y1, x1, c1 = _shaped[p1]
        y2, x2, c2 = _shaped[p2]

        if(c1 > confidence_threshold) & (c2 > confidence_threshold):
            # Buffer Pose Coordinate
            _poseBuf.append([[p1, int(x1), int(y1)],[p2, int(x2), int(y2)]])
    # Return
    return _poseBuf, _shaped
```

Draw connecting

- This is a function that can draw a skeleton model for a joint.

```
def Draw_Connecting(frame, _shaped, edge, confidence_threshold):
    for edge, color in EDGES.items():

        p1, p2 = edge
        y1, x1, c1 = _shaped[p1]
        y2, x2, c2 = _shaped[p2]

        if(c1 > confidence_threshold) & (c2 > confidence_threshold):
            cv.line(frame, (int(x1), int(y1)), (int(x2), int(y2)),BLUE, 2)
            cv.circle(frame, (int(x1), int(y1)), 4, GREEN, -1)
            cv.circle(frame, (int(x2), int(y2)), 4, GREEN, -1)
```

Start position

- If the value obtained by subtracting the calculated slopes between the shoulder and wrist joints is within the allowable range for a certain frame, a text indicating that the user is in the correct posture is given and start_Flag is turned on (start_Flag = True). Otherwise, feedback text about the posture is given.

```
# Get Start Flag and start_count for counting, offset_Text and gage bar
def Start_Postion_Adjustment(_frame, _poseBuf, _startCount, _startFlag,
 _offsetText):
    if _startCount >= START_COUNT_THRESH:
        _startFlag      = True

    if _startFlag == False:
        if len(_poseBuf) == CORRECT_RECOGNIGION:
            # print(pose_Buf)
            for i in range(len(_poseBuf)):
                if _poseBuf[i][0][0] == LEFT_SHOULDER:
                    x1_left = _poseBuf[i][0][1]
                    y1_left = _poseBuf[i][0][2]
                elif _poseBuf[i][1][0] == LEFT_WRIST:
                    x2_left = _poseBuf[i][1][1]
                    y2_left = _poseBuf[i][1][2]
                elif _poseBuf[i][0][0] == RIGHT_SHOULDER:
                    x1_right = _poseBuf[i][0][1]
                    y1_right = _poseBuf[i][0][2]
                elif _poseBuf[i][1][0] == RIGHT_WRIST:
                    x2_right = _poseBuf[i][1][1]
                    y2_right = _poseBuf[i][1][2]
            slope_left      = abs(round(float((y2_left-y1_left)/(x2_left-
x1_left+0.000001)), 3))
            slope_right     = abs(round(float((y2_right-y1_right)/(x2_right-
x1_right+0.000001)), 3))
```

```
            slope_offset    = abs(slope_left - slope_right)

            # Counting
            if y1_left / y2_left > 1.5 and y1_right / y2_right > 1.5:
                # _skeletoneFlag = True
                if slope_offset < START_THRESHOLD:
                    _startCount += 1
                    _offsetText = "Collect Position! Stop it"
                elif slope_offset >= START_THRESHOLD:
                    if slope_right > slope_left:
                        _offsetText = "Move your hands to the RIGHT"
                    elif slope_right < slope_left:
                        _offsetText = "Move your hands to the LEFT"
                    _startCount = 0
                if _startCount !=0:
                    cv.rectangle(_frame, (100, 60), (100+13*_startCount, 90),
GREEN, -1)
                    cv.rectangle(_frame, (100, 60), (100+13*30, 90), BLACK, 3)

    return _startFlag, _startCount, _offsetText
```

Calculate balance

- The balance is calculated using the relative proportions of the positions of both wrists.

```
def Calculate_Balance(_shaped, _balance):
    Left_w_y    = 0.0
    Right_w_y   = 0.0
    Left_w_y, _, Left_w_c       = _shaped[9]  # Left wrist
    Right_w_y, _, Right_w_c     = _shaped[10] # Right wrist
    # Valance > 0 : left wrist under the right wrist
    # Valance < 0 : right wrist under the left wrist
    if(Left_w_y != 0) & (Right_w_y != 0) & (Left_w_c > CONFIDENCE_THRESHOLD) &
(Right_w_c > CONFIDENCE_THRESHOLD):
        _balance = Left_w_y/Right_w_y - 1.0
    return _balance
```

Count workout

- Because there is a noise value, the positions of the elbows on both sides are accumulated for 10 frames each, and the flag is determined based on the average value.
- The condition for counting is when one or both elbows go down below the shoulder and then rise again.
- If both elbows go down and the balance value at that time is within the allowable range, the correct posture count is performed.
- Otherwise, if one elbow goes down or both elbows go down but the balance value exceeds the allowable range, a bad posture count is counted.
- Finally, when the correct posture count is equal to the target number of exercise, finish_Flag is turned on (finish_Flag = True).

```
# Count the workout number
```

```
def Count_Workout(_frame, _shaped, _inputCount, _finishFlag, _countList,
_balanceList, _updownflag):
    _countList[6] += 1

    Left_E_y, _, _          = _shaped[7]    # Left Elbow
    Right_E_y, _, _         = _shaped[8]    # Right Elbow

    _countList[2]    = _countList[2] + Left_E_y # left-Stack
    _countList[3]    = _countList[3] + Right_E_y # right-Stack


    if _countList[6]%10 == 0:
        _countList[6]       = 0
        _countList[4]       = _countList[2]/10.0      # Left_E_avg
        _countList[5]       = _countList[3]/10.0      # Right_E_avg
        _countList[2]       = 0                       # left-Stack initialize
        _countList[3]       = 0                       # right-Stack initialize
        # 0: down_Flag, 1: up_Flag, 2: left_down_Flag, 3: right_down_Flag,
4:left_up_Flag, 5: right_up_Flag
        # up/down flag control
        if _updownflag[0] == False: # 일단 내려갔다 라는 것에 대한 조건
            if _updownflag[2] == False: # 왼쪽이 내려갔는지
                if _countList[4]>shaped[LEFT_SHOULDER][0]:
                    _updownflag[2] = True
                    # Find Worst pose
                    if abs(_balanceList[0]) > abs(_balanceList[2]):
                        _balanceList[2] = _balanceList[0]
                        imwrite('WorstPose.jpg', _frame)
            if _updownflag[3] == False: # 오른쪽이 내려갔는지
                if _countList[5]>shaped[RIGHT_SHOULDER][0]:
                    _updownflag[3] = True
                    # Find Worst pose
                    if abs(_balanceList[0]) > abs(_balanceList[2]):
                        _balanceList[2] = _balanceList[0]
                        imwrite('WorstPose.jpg', _frame)

            if _updownflag[2] == True and _updownflag[3] == True: # 둘다 내려갔는지
                _updownflag[0] = True
                # Find Worst pose
                if abs(_balanceList[0]) < abs(_balanceList[1]):
                    _balanceList[1] = _balanceList[0]
                    imwrite('BestPose.jpg', _frame)
                print("Two hand Down")

        if _updownflag[0] == False:
            if _updownflag[3] == True:
                if _countList[5]<shaped[RIGHT_SHOULDER][0]:
                    _countList[1]+=1
                    _updownflag[3] = False
                    _updownflag[2] = False
            if _updownflag[2] == True:
                if _countList[4]<shaped[LEFT_SHOULDER][0]:
                    _countList[1]+=1
                    _updownflag[2] = False
                    _updownflag[3] = False
        else:
            if _countList[5]<shaped[RIGHT_SHOULDER][0] and _countList[4]
<shaped[LEFT_SHOULDER][0]:
```

```
                if _balanceList[0] < -0.2 or _balanceList[0] > 0.2:
                    _countList[1]+=1
                else:
                    _countList[0]+=1
                _updownflag[0] = False
                _updownflag[2] = False
                _updownflag[3] = False

    # For finish Flage
    if _countList[0] == _inputCount:
        _finishFlag = True

    return _finishFlag, _countList, _balanceList, _updownflag, _countList[6]
```

Reset all parameter

- Reset all parameter when system Flag is off(value = False)

```
def ResetPram():
    # System flag
    global system_Flag, start_Flag, finish_Flag, up_down_Flag, counting_List,
start_Count, offset_Text, balance_List, frame_Num, inputCount, frame_Num2,
position_FrameList

    # System flag
    system_Flag     = False
    start_Flag      = False
    finish_Flag     = False

    # For workout count
    # 0: down_Flag, 1: up_Flag, 2: left_down_Flag, 3: right_down_Flag,
4:left_up_Flag, 5: right_up_Flag
    up_down_Flag    = [False, False, False, False]

    # 0: Good count, 1: bad count, 2: left_E_ystack, 3: right_E_ystack, 4:
left_E_avg, 5: right_E_avg 6: count_frame
    counting_List   = [0, 0, 0, 0, 0.0, 0.0, 0]

    # For starting setting
    # 0:set Count, 1: startCount, 2: userSet
    start_Count = 0
    offset_Text = ""

    # Balance_List
    # 0: balance, 1: good, 2: bad)
    balance_List = [0.0, 10.0, 0.0]

    # Frame count
    frame_Num           = 0         # for processing
    position_FrameList  = [0, 0]    # 0: worst / 1: Best
```

6-2. Show Text

Show text

- Show text for all count, good count, bad count, balance, feedback and finish

```python
def show_Text(_img, _Balance, _count, _countBad, _flag):
    if _flag == False:
        # All workout count
        TEXT_allCount          = f"All count = {_count+_countBad}"
        # Good workout count
        TEXT_goodCount          = f"Good count = {_count}/{inputCount}"
        # Bad workout count
        TEXT_bedCount       = f"Bad count = {_countBad}"
        # Balance value
        TEXT_Bal_value      = f"Balance = {round(_Balance,3)}"
        # Balance Feedback
        if _Balance <-0.2:
            TEXT_Bal_Fed    = f"FeedBack: Push down left side"
            color = RED
        elif _Balance >0.2:
            TEXT_Bal_Fed    = f"FeedBack: Push down right side"
            color = RED
        else:
            TEXT_Bal_Fed    = f"FeedBack: Good pos bro!!."
            color = WHITE

        textSize, _ = cv.getTextSize(TEXT_Bal_Fed, USER_FONT, fontScale, fontThickness)

        # Print Text
        cv.rectangle(_img, (_img.shape[0]-530, 0), (_img.shape[0], 200), BLACK, -1)
        cv.putText(_img, TEXT_allCount, (_img.shape[0]-530, textSize[1]), USER_FONT, fontScale, WHITE, fontThickness)
        cv.putText(_img, TEXT_goodCount, (_img.shape[0]-530, textSize[1]+40), USER_FONT, fontScale, WHITE, fontThickness)
        cv.putText(_img, TEXT_bedCount, (_img.shape[0]-530, textSize[1]+80), USER_FONT, fontScale, WHITE, fontThickness)
        cv.putText(_img, TEXT_Bal_value, (_img.shape[0]-530, textSize[1]+120), USER_FONT, fontScale, color, fontThickness)
        cv.putText(_img, TEXT_Bal_Fed, (_img.shape[0]-530, textSize[1]+160), USER_FONT, fontScale, color, fontThickness)
    else:
        TEXT_finish = f"Finish!! Good Job brother~~."

        textSize, _ = cv.getTextSize(TEXT_finish, USER_FONT, fontScale, fontThickness)
        cv.rectangle(_img, (int(_img.shape[0]/2.0)-int(textSize[0]/2.0)-5, int(_img.shape[1]/2)-textSize[1]-5), (int(_img.shape[0]/2.0)+int(textSize[0]/2.0)+5, int(_img.shape[1]/2)+5), BLACK, -1)
        cv.putText(_img, TEXT_finish, (int(_img.shape[0]/2.0)-int(textSize[0]/2.0), int(_img.shape[1]/2)), USER_FONT, fontScale, RED, fontThickness)
```

Show start text

- The text output of the **Start_Postion_Adjustment** function is displayed on the image.

```
def show_Start_text(_img, _text):
    cv.putText(_img, _text, (100, 50), USER_FONT, fontScale, WHITE,
fontThickness)
```

7. Main Code

Model Interpreter Definition

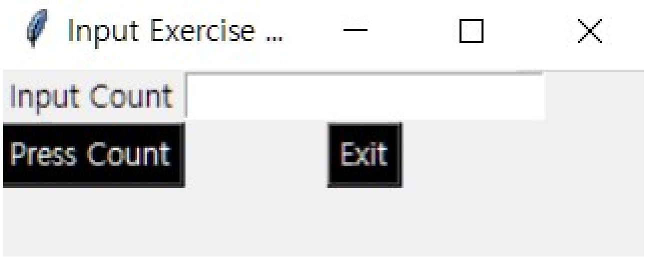
- Load the model
- The model file must be the same as the path where the corresponding code is located.

```
interpreter = tf.lite.Interpreter(model_path = 'lite-
model_movenet_singlepose_lightning_3.tflite')
interpreter.allocate_tensors()
```

TKinter for user count

- When starting for the first time, this is a code that allows you to input the number of target workouts.

```
# TKinter for USER Count
tk = Tk()
tk.title('Input Exercise Count')
tk.geometry("200x100")

def Input_Count():
    global set_List
    global inputCount
    A = int(entry1.get())
    inputCount = A

label1 = Label(tk, text='Input Count').grid(row=0, column=0)
entry1 = Entry(tk)
entry1.grid(row=0,column=1)

btn = Button(tk, text='Press Count', bg='black', fg='white',
command=Input_Count).grid(row=1,column=0)
exit_button = Button(tk, text='Exit', bg='black', fg='white',
command=tk.destroy).grid(row=1,column=1)

tk.mainloop()
```



Open the Video & Recording Video Configuration

- Open the video we using and do ready for recording

```
# cv.VideoCaputure(0) -> notebook cam
# cv.VideoCaputure(1) -> another cam connecting with my notebook
# cv.VideoCapture(filename.mp4) -> Video
cap = cv.VideoCapture("DEMO.mp4")

# Recording Video Configuration
w = round(cap.get(CAP_PROP_FRAME_WIDTH))
h = round(cap.get(CAP_PROP_FRAME_HEIGHT))
fps = cap.get(CAP_PROP_FPS)
fourcc = VideoWriter_fourcc(*'DIVX')
out = VideoWriter('output.avi', fourcc, fps, (w,h))
delay = round(1000/fps)

if (cap.isOpened() == False): # if there is no video we can open, print error
  print("Not Open the VIDEO")
```

Start the system by while()

- Processing is performed for each frame of the video using a while().
- First, when the user wants to receive input again (press 'r' key -> tk_Falg = True), a section to receive input again was placed at the beginning of while().
- And use the function (cv.getTickCount()) to get the time to measure the FPS.
- Get a frame from video.

```
#================== While Loop =================#
while cap.isOpened():
    # When you press the 'r' botton -> restart
    if tk_Flag == True:
        tk = Tk()
        tk.title('Input Exercise Count')
        tk.geometry("200x100")

        label1 = Label(tk, text='Input Count').grid(row=0, column=0)
        entry1 = Entry(tk)
        entry1.grid(row=0,column=1)

        btn = Button(tk, text='Press Count', bg='black', fg='white',
command=Input_Count).grid(row=1,column=0)
        exit_button = Button(tk, text='Exit', bg='black', fg='white',
command=tk.destroy).grid(row=1,column=1)
        tk.mainloop()
        ResetPram()
        tk_Flag = False

    frame_Num += 1

    # Start Window Time
    startTime = cv.getTickCount()

    # Video Read
    ret, frame = cap.read()
    if ret == False:
        print("Video End")
        break
```

Resize the frame, Setup input detail and output detail, and Input to model and get output

- Resized to fix frame size to 1080x1080.
- Since the input size of the deep learning model we use is 192x192, we change the frame to 192x192 to put it as an input.

  (Using **resize_with_pad()**)
- You need to know the information of the input tensor and the output tensor in order to transmit and receive data, so setup is done.
- Input to model and get output

```
    # Reshape image
    frame       = resize(frame, dsize = (1080, 1080), interpolation=INTER_LINEAR)
    img         = frame.copy()
    img         = tf.image.resize_with_pad(np.expand_dims(img, axis=0), 192, 192)
    input_image = tf.cast(img, dtype=tf.float32)

    #Setup input and Output
    input_details   = interpreter.get_input_details()   # receive information of input tensor
    output_details  = interpreter.get_output_details()  # receive information of output tensor

    # input to model
    interpreter.set_tensor(input_details[0]['index'], np.array(input_image))
    interpreter.invoke()
    # Get output to model
    keypoints_with_scores = interpreter.get_tensor(output_details[0]['index'])
```

Main code

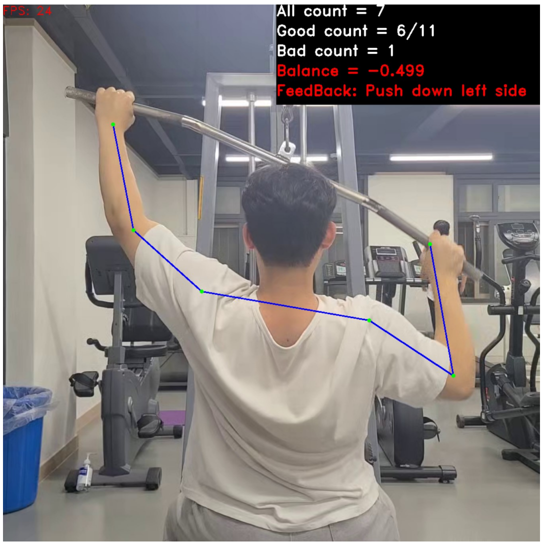
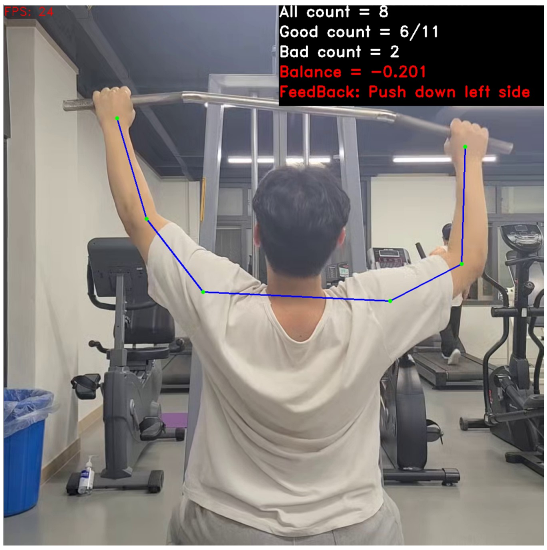
- We bring about the joint information we want to use.

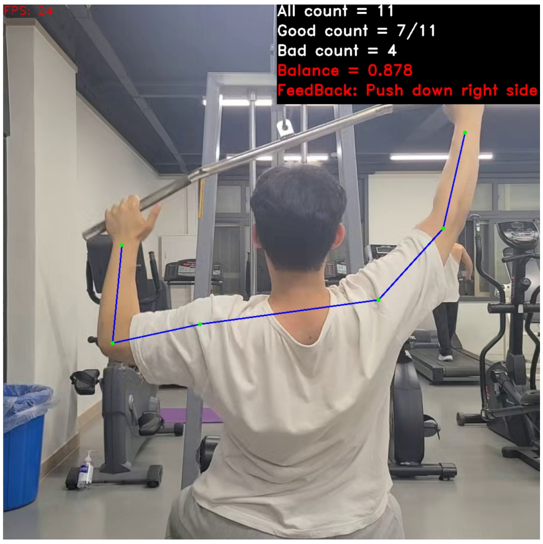
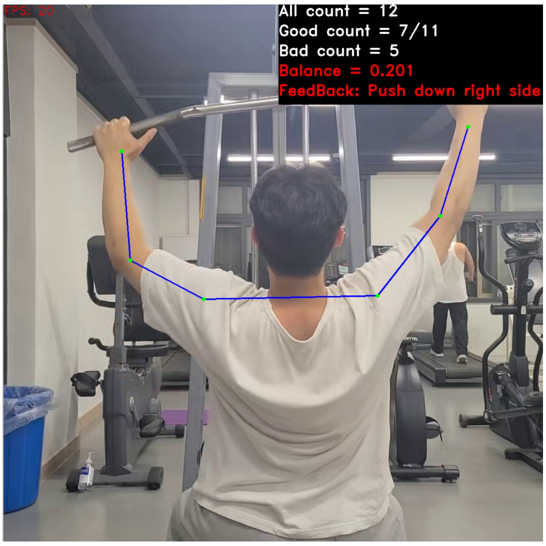
  (Wrist, Elbow, and Shoulder)

- To adjust the flag to start the system, adjust the flag depending on whether the elbow is above or below the shoulder position.

  (flag off(**system_Flag = False**) if the elbow position is below the shoulder / flag off(**system_Flag = True**) if the elbow position is above the shoulder))
- If the system flag is turned on (**system_Flag == True**), the finish flag is checked, and if it is off, the function to adjust the start flag is executed.
- If the flag is turned on through the function (**start_Flag == True**), the balance is calculated and the exercise count starts.
- Otherwise, the feedback on the starting posture adjustment is output as text.
- When the exercise done with the correct posture is equal to the target number of exercises, the finish flag is turned on (**finish_Flag == True**).
- When the finish flag is turned on, the system flag is turned on and the exercise result window appears.

```
    # =================== START POINT =================== #
    pose_Buf = []
    pose_Buf, shaped = Get_Shape_PoseBuf(frame, keypoints_with_scores, EDGES, 
CONFIDENCE_THRESHOLD)

    if shaped[RIGHT_WRIST][0]>shaped[RIGHT_ELBOW][0] and shaped[LEFT_WRIST]
[0]>shaped[LEFT_ELBOW][0]: # 쉴 때의 조건
        system_Flag = False
    else:
        system_Flag = True

    if system_Flag == True:
        # Get Start Flag and start_count for counting, offset_Text
        if finish_Flag == False:
            start_Flag, start_Count, offset_Text = 
Start_Postion_Adjustment(frame, pose_Buf, start_Count, start_Flag, offset_Text)
            if start_Flag == False:
                show_Start_text(frame, offset_Text)

        # draw skeleton
        Draw_Connecting(frame, shaped, EDGES, CONFIDENCE_THRESHOLD)

        # Start count processing
        if start_Flag == True and finish_Flag == False:
            balance_List[0] = Calculate_Balance(shaped, balance_List[0])
            finish_Flag, counting_List, balance_List, up_down_Flag, 
counting_List[6] = Count_Workout(frame, shaped, inputCount, finish_Flag, 
counting_List, balance_List, up_down_Flag)

            show_Text(frame, balance_List[0], counting_List[0], 
counting_List[1], finish_Flag)
        # Finish Flag
        elif finish_Flag == True:
            show_Text(frame, balance_List[0], counting_List[0], 
counting_List[1], finish_Flag)
            system_Flag = False

        # Press Esc to Exit, Stop Video to 's'
        k = cv.waitKey(5) & 0xFF
        if k == 27:
            break
        elif k == ord('s'):
            cv.waitKey()
        elif k == ord('r'):
            tk_Flag = True

        # Time Loop End
        endTime = cv.getTickCount()

        # FPS Calculate
        FPS = round(getTickFrequency()/(endTime - startTime))

        # FPS Text
        FPS_Text = f"FPS: {FPS}"
        putText(frame, FPS_Text, (0, 20), USER_FONT, 0.8, RED)

        cv.imshow('MoveNet Lightning', frame)
        resizeWindow('MoveNet Lightning', 1080, 1080)
    else:
        if finish_Flag == False:
            ResetPram()
        else:
            break
        # Press Esc to Exit, Stop Video to 's'
        k = cv.waitKey(5) & 0xFF
        if k == 27:
            break
        elif k == ord('s'):
            cv.waitKey()
        elif k == ord('r'):
            tk_Flag = True

        # Time Loop End
        endTime = cv.getTickCount()

        # FPS Calculate
        FPS = round(getTickFrequency()/(endTime - startTime))

        # FPS Text
        FPS_Text = f"FPS: {FPS}"
        putText(frame, FPS_Text, (0, 20), USER_FONT, 0.8, RED)

        cv.imshow('MoveNet Lightning', frame)

    # # Record Video
    out.write(frame)

cap.release()
cv.destroyAllWindows()
out.release()
```

Left	Bad Count
 FPS: 24 All count = 7 Good count = 6/11 Bad count = 1 Balance = -0.499 FeedBack: Push down left side	 FPS: 24 All count = 8 Good count = 6/11 Bad count = 2 Balance = -0.201 FeedBack: Push down left side

Right	Bad Count
 FPS: 24 All count = 11 Good count = 7/11 Bad count = 4 Balance = 0.878 FeedBack: Push down right side	 FPS: 20 All count = 12 Good count = 7/11 Bad count = 5 Balance = 0.201 FeedBack: Push down right side

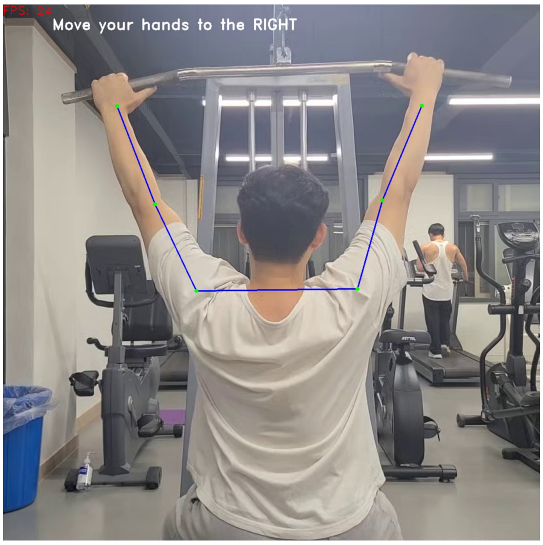
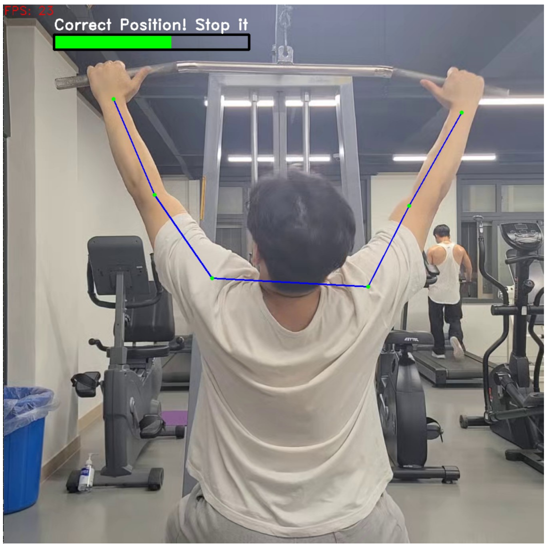
8. Show the result of workout

- show the result(worst pose, best pose, good pose count, bed pose count, and all count)

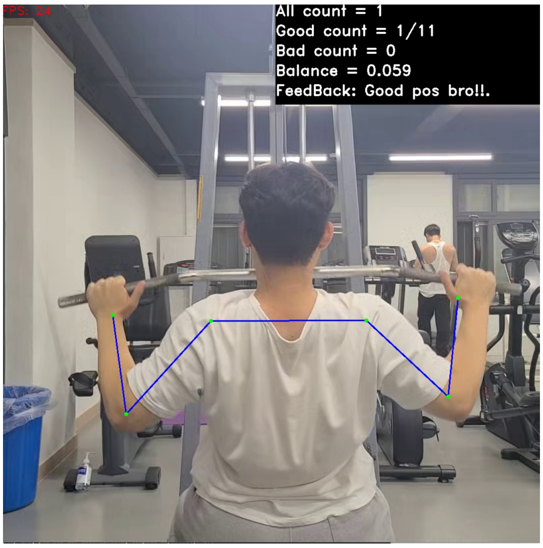
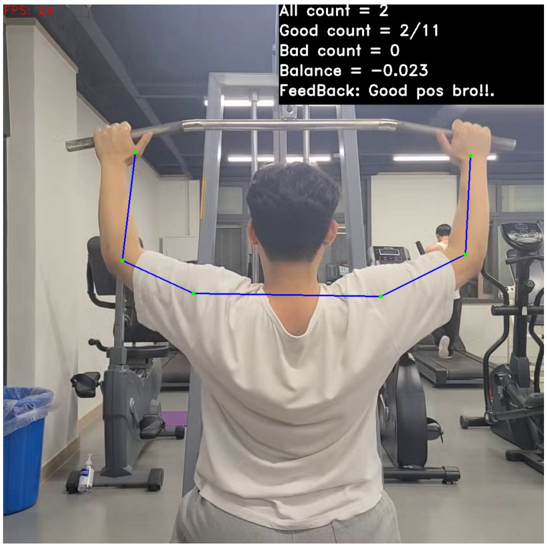
```
# ==============================================================================
#
# ================================ Final result report ==========================
#
# ==============================================================================
#
if finish_Flag == True: # when finish flag is on
    # image stack
    best_image      = cv.imread('BestPose.jpg')
    best_image      = cv.resize(best_image, (0,0), None, .5, .5)
    worst_image     = cv.imread('WorstPose.jpg')
    worst_image     = cv.resize(worst_image, (0,0), None, .5, .5)
    pose_result     = np.vstack((best_image, worst_image))
    result_paper    = np.zeros_like(pose_result)
    workout_result  = np.hstack((result_paper, pose_result))

    # Put text
    # Make text for result image
    TEXT_GOOD       = f"Best Pose(Balance {abs(round(balance_List[1],3))})"
    TEXT_BED        = f"Worst Pose(Balance {abs(round(balance_List[2],3))})"

    # Make text for result report
    TEXT_RESULT1         = f"======================"
    TEXT_RESULT          = f"---------Result---------"
    TEXT_RESULT2         = f"======================"
    TEXT_GOOD_COUNT     = f"Count about Good Pose = {counting_List[0]}"
    TEXT_BED_COUNT      = f"Count about Bed Pose = {counting_List[1]}"
    TEXT_COUNT          = f"Count about All Pose =
{counting_List[0]+counting_List[1]}"

    TEXT_RATIO          = f"Performace ratio ="

    # Parameter for position
    Size_GOOD, _    = cv.getTextSize(TEXT_GOOD, USER_FONT, fontScale,
fontThickness)
    Size_BED, _     = cv.getTextSize(TEXT_BED, USER_FONT, fontScale,
fontThickness)
    width           = best_image.shape[0] # best image width
    height          = best_image.shape[1] # best image height
    x_GOOD          = Size_GOOD[0]
    y_GOOD          = Size_GOOD[1]
    x_BED           = Size_BED[0]
    y_BED           = Size_BED[1]

    # Draw ract and Put Text for image
    cv.rectangle(workout_result, (width, 0), (width*2, y_GOOD+13), WHITE, -1)
    cv.rectangle(workout_result, (width, height), (width*2, y_GOOD+height+13),
WHITE, -1)
    cv.putText(workout_result, TEXT_BED, (width+int(width/2)-int(x_BED/2),
y_GOOD+height+5), USER_FONT, fontScale, BLACK, fontThickness)
    # Put Text for result report
    cv.putText(workout_result, TEXT_RESULT1, (0, y_GOOD), USER_FONT, fontScale,
RED, fontThickness)
    cv.putText(workout_result, TEXT_RESULT, (0, y_GOOD*2), USER_FONT, fontScale,
RED, fontThickness)
    cv.putText(workout_result, TEXT_RESULT2, (0, y_GOOD*3), USER_FONT,
fontScale, RED, fontThickness)

    cv.putText(workout_result, TEXT_COUNT, (0, y_GOOD*4), USER_FONT, fontScale,
WHITE, fontThickness)
    cv.putText(workout_result, TEXT_GOOD_COUNT, (0, y_GOOD*6), USER_FONT,
fontScale, WHITE, fontThickness)
    cv.putText(workout_result, TEXT_BED_COUNT, (0, y_GOOD*8), USER_FONT,
fontScale, WHITE, fontThickness)


    cv.imshow('Final result of your workout', workout_result)
    waitKey()

    cv.destroyAllWindows()
```

V. Result

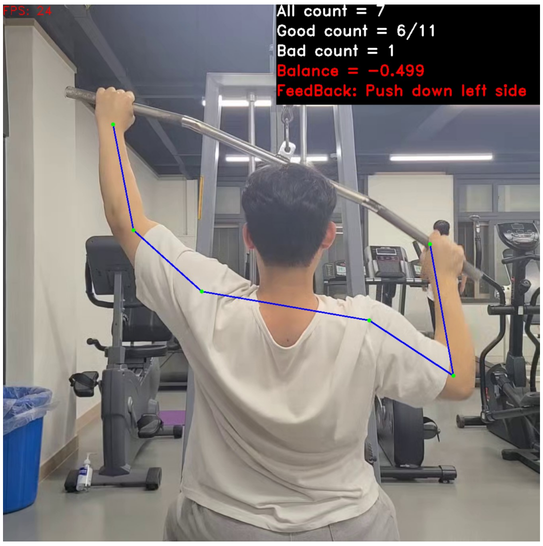
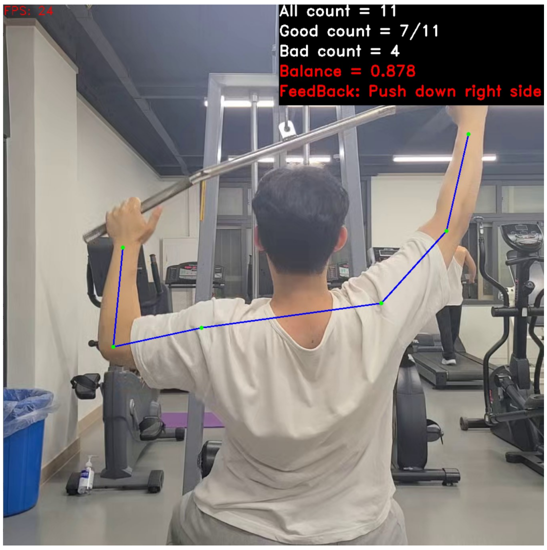
1. Adjust Correct Starting Position

Adjust	Complete
 FPS: 24 Move your hands to the RIGHT	 FPS: 23 Correct Position! Stop it

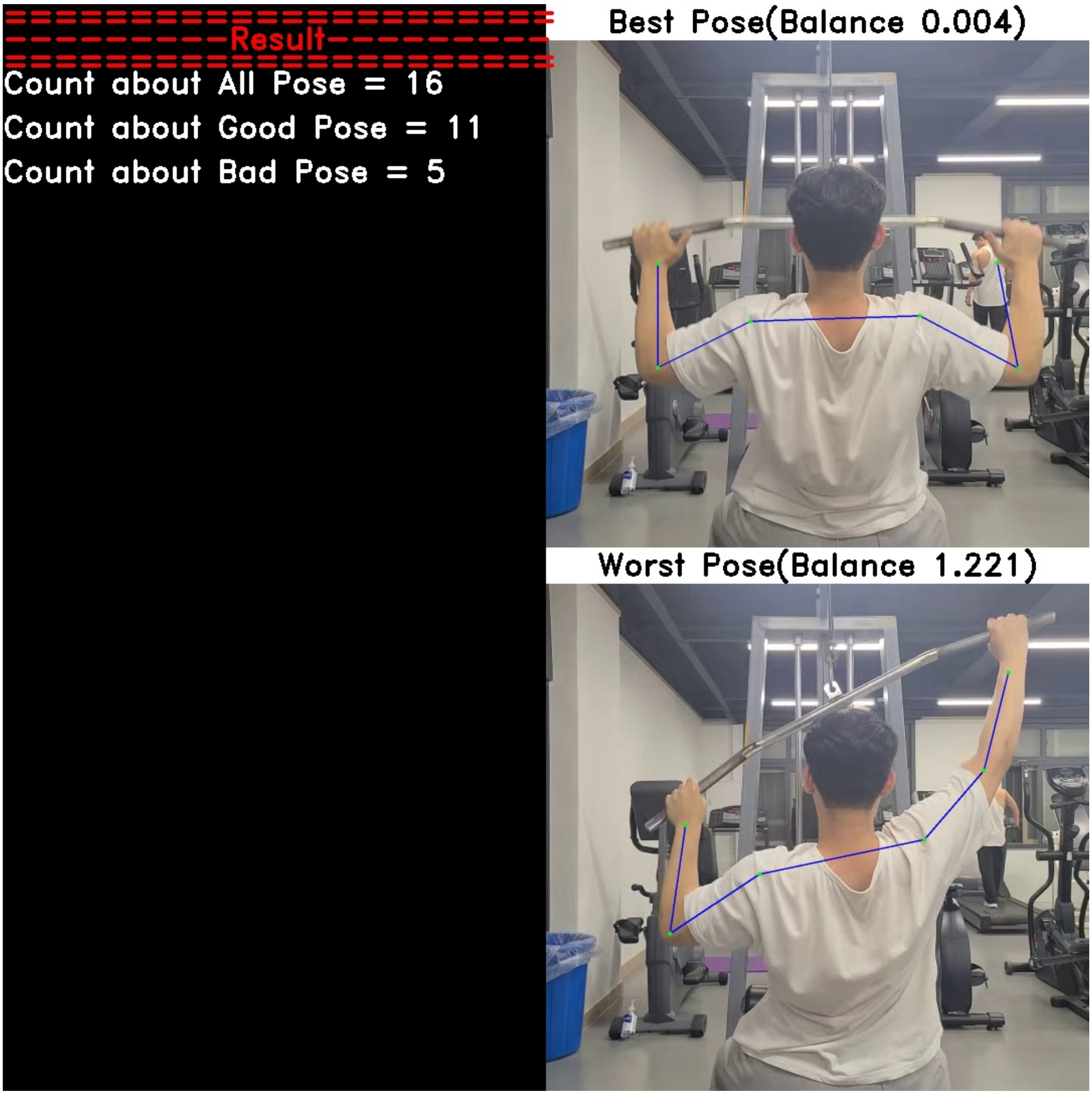
2. Exercising

Contraction	Relaxation
 FPS: 24 All count = 1 Good count = 1/11 Bad count = 0 Balance = 0.059 FeedBack: Good pos bro!!.	 FPS: 24 All count = 2 Good count = 2/11 Bad count = 0 Balance = −0.023 FeedBack: Good pos bro!!.

3. Unbalance

Left	Right
 FPS: 24 All count = 7 Good count = 6/11 Bad count = 1 Balance = -0.499 FeedBack: Push down left side	 FPS: 24 All count = 11 Good count = 7/11 Bad count = 4 Balance = 0.878 FeedBack: Push down right side

4. Show Result



```
==========================
-----------Result----------
==========================
Count about All Pose = 16
Count about Good Pose = 11
Count about Bad Pose = 5
```

Best Pose(Balance 0.004)

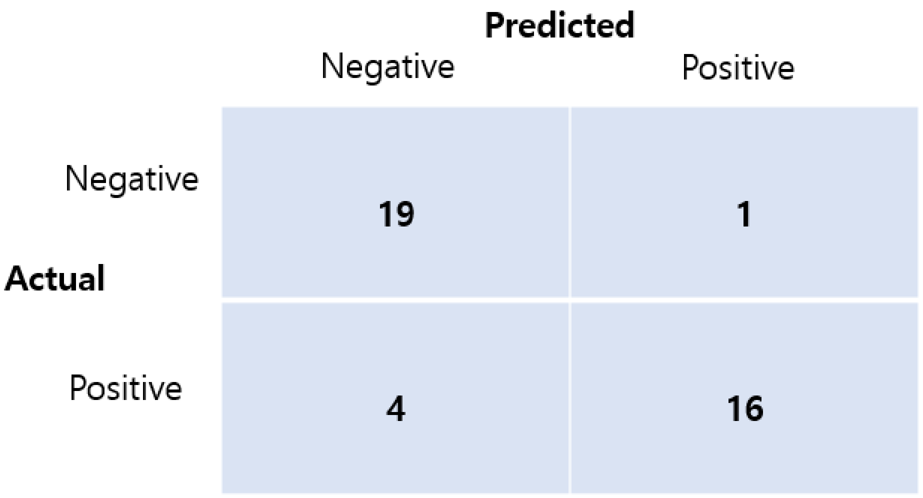
Worst Pose(Balance 1.221)

VI. Evaluation

Since we used the pre-trained model, we analyzed the algorithm we implemented, not the analysis of the model itself. The adjust correct starting position part and the experimenting part were largely divided and analyzed.

- **Adjust Correct Starting Position**

  For evaluation, another Lat-Pull Down machine tested "Adjust Correct Starting Position" 20 times per person and 40 times in total. In this case, Positive means Correct Position, and Negative means a state in which movement to right or left is required. Accordingly, the heat map is as follows, and based on this, Accuracy, Precision, and Recall are analyzed.



		Predicted	
		Negative	Positive
Actual	Negative	19	1
	Positive	4	16

  - Accuracy: 87.5%

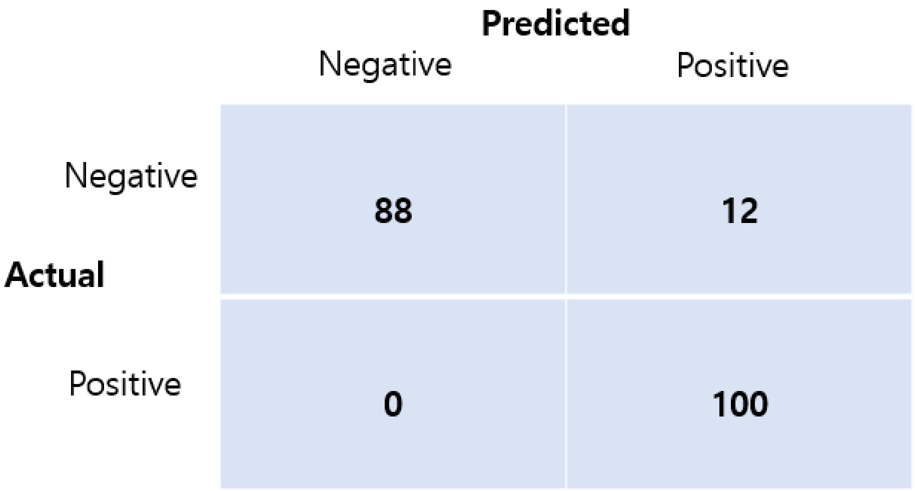
  - Precision: 94.1%

  - Recall: 80.0%

  Looking at the above results, Recall is lowered, which means that negative is frequently recognized (FN) when positive. In other words, it can be seen that the threshold value should be adjusted so that it can be clearly recognized as positive when it is positive.

- **Workout Counting**

  This time, an experiment on "Workout Counting" is conducted 20 times per person, 5 sets, and a total of 200 times. At this time, Positive means exercising in the right posture, and Negative means exercising in the wrong posture. The heat map accordingly is as follows, and based on this, Accuracy, Precision, and Recall are analyzed.



		Predicted	
		Negative	Positive
Actual	Negative	88	12
	Positive	0	100

- Accuracy: 94.0%

- Precision: 89.3%

- Recall: 100.0%

Looking at the above results, the precision is lowered, which means that there are many cases (FP) that are perceived as positive when negative. In the experiment, a mirror is present and the precision is lowered due to the recognition of the person in the mirror. In other words, when using this program, it should be executed in an environment where there is nothing else that can be recognized as a person other than the surrounding me.

VII. Reference

- MoveNet Colab: https://github.com/nicknochnack/MoveNetLightning
- Pose Estimation with Fastest Deep Learning Model Tutorial
- Installation Guide - Y.K.Kim Deep-Learning-Image-Processing Installation Guide