DEADLIFT MONITORING USING COMPUTER VISION

ABSTRACT:

The dead-lift is a weight training exercise in which a loaded barbell or bar is lifted off the ground to the level of the hips, torso perpendicular to the floor, before being placed back on the ground. It is one of the three power lifting exercises, along with the squat and bench press. Dead-lifting creates a large amount of torque at hips and low back. Poor technique due to excessive weight may create an imbalance in the distribution of load between these areas, quite commonly increasing load at the lumbar spine and increasing the risk of injury. Hence proper guidance of the exercise is to be implemented in order to prevent the trainee from any health issues. The implementation is done using machine learning models and computer vision.

PROJECT OBJECTIVE:

The primary goal of the project is to assist the trainee to do proper dead-lifting without any misleading actions. The important features of the project are listed below:

- The model tries to detect the motion of the trainee using computer vision
- Detection of the structural framework of the trainee
- Calculation of the count of reps taken
- Probability calculation to indicate the risk of improper posture
- User friendly interface for the ease of use

TECHNOLOGY AND FRAMEWORK USED:

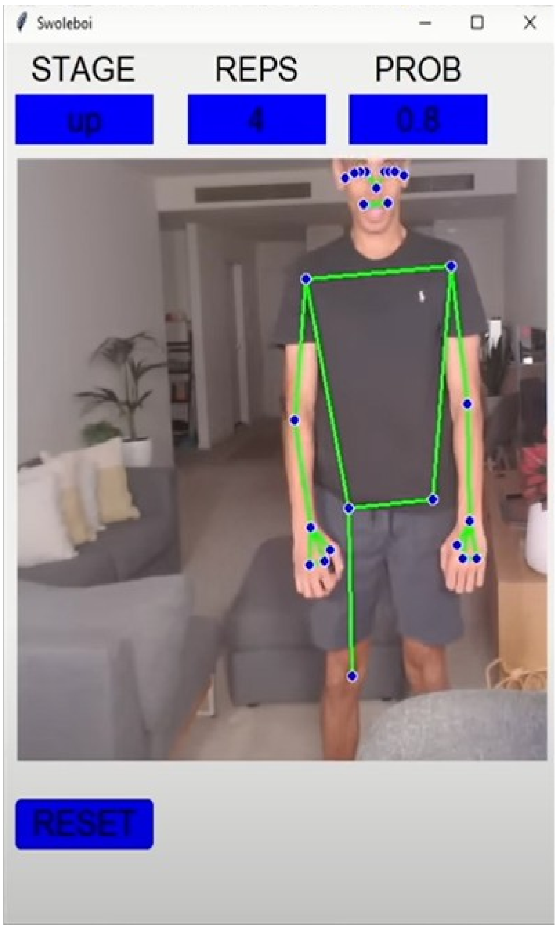
- Python
- Cv2 and PIL for computer vision and image processing
- Numpy and pandas for computations
- Mediapipe for prototyping perception pipelines
- Pickle for implementing dead-lift framework
- Tkinter for user interface

CODING:

```
import tkinter as tk
import customtkinter as ck

import pandas as pd
import numpy as np
import pickle

import mediapipe as mp
import cv2
from PIL import Image, ImageTk

from landmarks import landmarks

window = tk.Tk()
window.geometry("480x700")
window.title("Swoleboi")
ck.set_appearance_mode("dark")

classLabel = ck.CTkLabel(window, height=40, width=120, text_font=("Arial", 20),
text_color="black", padx=10)
classLabel.place(x=10, y=1)
classLabel.configure(text='STAGE')
counterLabel = ck.CTkLabel(window, height=40, width=120, text_font=("Arial", 20),
text_color="black", padx=10)
counterLabel.place(x=160, y=1)
counterLabel.configure(text='REPS')
probLabel  = ck.CTkLabel(window, height=40, width=120, text_font=("Arial", 20),
text_color="black", padx=10)
probLabel.place(x=300, y=1)
probLabel.configure(text='PROB')
classBox = ck.CTkLabel(window, height=40, width=120, text_font=("Arial", 20),
text_color="white", fg_color="blue")
classBox.place(x=10, y=41)
classBox.configure(text='0')
counterBox = ck.CTkLabel(window, height=40, width=120, text_font=("Arial", 20),
text_color="white", fg_color="blue")
counterBox.place(x=160, y=41)
counterBox.configure(text='0')
probBox = ck.CTkLabel(window, height=40, width=120, text_font=("Arial", 20),
text_color="white", fg_color="blue")
```

```
probBox.place(x=300, y=41)
probBox.configure(text='0')


def reset_counter():
    global counter
    counter = 0

button = ck.CTkButton(window, text='RESET', command=reset_counter, height=40,
width=120, text_font=("Arial", 20), text_color="white", fg_color="blue")
button.place(x=10, y=600)

frame = tk.Frame(height=480, width=480)
frame.place(x=10, y=90)
lmain = tk.Label(frame)
lmain.place(x=0, y=0)

mp_drawing = mp.solutions.drawing_utils
mp_pose = mp.solutions.pose
pose = mp_pose.Pose(min_tracking_confidence=0.5, min_detection_confidence=0.5)

with open('deadlift.pkl', 'rb') as f:
    model = pickle.load(f)

cap = cv2.VideoCapture(0)
current_stage = ''
counter = 0
bodylang_prob = np.array([0,0])
bodylang_class = ''

def detect():
    global current_stage
    global counter
    global bodylang_class
    global bodylang_prob

    ret, frame = cap.read()
    image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    results = pose.process(image)
    mp_drawing.draw_landmarks(image, results.pose_landmarks,
mp_pose.POSE_CONNECTIONS,
        mp_drawing.DrawingSpec(color=(106,13,173), thickness=4, circle_radius = 5),
        mp_drawing.DrawingSpec(color=(255,102,0), thickness=5, circle_radius = 10))

    try:
        row = np.array([[res.x, res.y, res.z, res.visibility] for res in
results.pose_landmarks.landmark]).flatten().tolist()
        X = pd.DataFrame([row], columns = landmarks)
        bodylang_prob = model.predict_proba(X)[0]
        bodylang_class = model.predict(X)[0]

        if bodylang_class =="down" and bodylang_prob[bodylang_prob.argmax()] > 0.7:
            current_stage = "down"
        elif current_stage == "down" and bodylang_class == "up" and
bodylang_prob[bodylang_prob.argmax()] > 0.7:
            current_stage = "up"
            counter += 1

    except Exception as e:
        print(e)

    img = image[:, :460, :]
    imgarr = Image.fromarray(img)
    imgtk = ImageTk.PhotoImage(imgarr)
    lmain.imgtk = imgtk
    lmain.configure(image=imgtk)
    lmain.after(10, detect)

    counterBox.configure(text=counter)
    probBox.configure(text=bodylang_prob[bodylang_prob.argmax()])
    classBox.configure(text=current_stage)

detect()
window.mainloop()
```

OUTPUT:



FUTURE SCOPE:

In the perspective of improving the quality of user experience, the can be an voise assistant in guiding the user with the rep count, alarming the user not to stay in a particular posture that might cause risk, etc. An exclusive training manual can also be added/

RESULT:

Thus the trained ML model can be used for assisting the trainee for a better experience of smooth exercising and uplifting the health perspectives. The implementation is straightforward and cost-efficient as the model can work on multiple supportive devices like webcams, raspberry pi, etc.