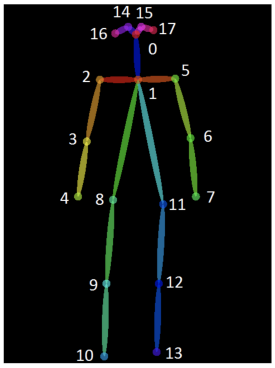
DEADLIFT DETECTION

INTRODUCTION

- Deadlift Detection app in python uses the machine learning along with the cv2 to detect the body movement of the object by using the landmarks.



- Here pickle file is used to serialize python object structures , which refers to the process of converting an object in the memory to the byte stream that can be stored in a binary file on the disk.
- When we load it back to a python program , this binary file can be de-serialized back to a python object.
- The usage of this app is many of the fitness organizations like Cultfit , HRX can use this to track the movements of their client and can make gym automate by integrating this with the machine itself.

DEPENDENCIES REQUIRED

Tkinter

- Library in python which is used to create GUI's in Python.
- This is used in many machine learning library which is used by the cv2 here we will use this to create the frame and labels like sets , reps and probability which will be tracked by our model during and exercise.
- By using this library we will create our GUI consisting of labels , buttons , camera frame for tracking.

tkinter - Python interface to Tcl/Tk

Source code: Lib/tkinter/__init__.py The package ("Tk interface") is the standard Python interface to the Tcl/Tk GUI toolkit. Both Tk and are available on most Unix platforms, including macOS, as well as on Windows systems. Running from the command line should open a window demonstrating a simple Tk interface, letting you know that

https://docs.python.org/3/library/tkinter.html

Customtkinter

- This is based on the tkinter library which is already present in python.
- This is used to create the modern looking GUI's in Python.

customtkinter

Skip to main content Warning Some features may not work without JavaScript. Please try enabling it if you encounter problems. Create modern looking GUIs with Python Download the file for your platform. If you're not sure which to choose, learn more about installing packages.

https://pypi.org/project/customtkinter/#description

Pandas

- It is software library written for the python programming language.
- It is use for the data manipulations and analysis.
- It offers data structures and operations for manipulating numerical tables and time series.

pandas

pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

https://pandas.pydata.org/

Numpy

- It is a library for the python programming language , adding support for large multi-dimensional arrays and matrices.
- It has a large collection of high level mathematical functions to operate on these arrays.

NumPy

Why NumPy? Powerful n-dimensional arrays. Numerical computing tools. Interoperable. Performant. Open source.

https://numpy.org/

Pickle

- It is the process whereby a Python object hierarchy is converted into a byte stream.

pickle - Python object serialization

Source code: Lib/pickle.py The pickle module implements binary protocols for serializing and de-serializing a Python object structure. "Pickling" is the process whereby a Python object hierarchy is...

https://docs.python.org/3/library/pickle.html

Mediapipe

- It is a cross platform library developed by Google that provide amazing ready-to-use ML solutions for computer vision tasks.
- Open CV library in python is a computer vision library that is widely used for :
  - Image analysis.

- Image processing.
- Image detection.
- Image Recognition etc.

Face and Hand Landmarks Detection using Python - Mediapipe, OpenCV - GeeksforGeeks

In this article, we will use python library to detect face and hand landmarks. We will be using a Holistic model from mediapipe solutions to detect all the face and hand landmarks. We will be also seeing how we can access different landmarks of the face and hands which can be used for different computer vision applications such as sign

https://www.geeksforgeeks.org/face-and-hand-landmarks-detection-using-python-mediapipe-opencv/

Cv2

- It is a library of Python bindings designed to solve computer vision problems.
- cv2.imread()
  - This method loads an image form the specified file.
  - If the image cannot be read (because of missing file , improper permission, unsupported or invalid format) then this method returns an empty matrix.

opencv-python

Pre-built CPU-only OpenCV packages for Python. Check the manual build section if you wish to compile the bindings from source to enable additional modules such as CUDA. If you have previous/other manually installed (= not installed via pip) version of OpenCV installed (e.g.

https://pypi.org/project/opencv-python/

PIL

- It is python imaging Library.
- It is the original library that enabled Python to deal with images.
- It was discontinued in 2011 that it only supports Python 2.
- To use its developer's own description.
  - Pillow is the friendly PIL fork that kept the library alive and includes support for Python 3.

Python: Pillow (a fork of PIL) - GeeksforGeeks

Python Imaging Library (expansion of PIL) is the de facto image processing package for Python language. It incorporates lightweight image processing tools that aids in editing, creating and saving images. Support for Python Imaging Library got discontinued in 2011, but a project named pillow forked the original PIL project and

https://www.geeksforgeeks.org/python-pillow-a-fork-of-pil/

Landmarks

- It is an additional file which contains all the landmarks in the form of an array these are used to mark the object body landmarks like :
  - Ears.
  - Eyes.
  - Mouth.
  - Nose etc.

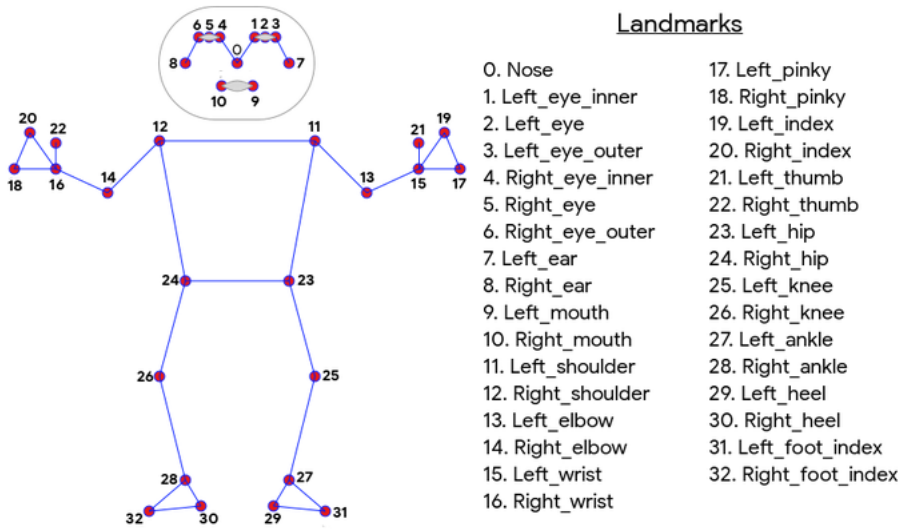


Figure 1. Landmarks

- You can read more about it here ⬇️ :

Pose detection | ML Kit | Google Developers

This API is offered in beta. It may be changed in backward-incompatible ways and is not subject to any SLA or deprecation policy. The ML Kit Pose Detection API is a lightweight versatile solution for app developers to detect the pose of a subject's body in real time from a continuous video or static image.

https://developers.google.com/ml-kit/vision/pose-detection

Google Developers

BUILDING PROJECT

SETTING UP THE GUI

- In this Step we will be using the tkinter to create the GUI and custom tkinter is used to add the label and there is the reset button which is created by using the custom tkinter.
- First of all we will use the tkinter to create an window and in this window we will add the geometry and title and then we will use the custom tkinter to set the favorite appearance mode.

```
window =  tk.Tk()
window.geometry("480x700")
window.title("Deadlift Detection")
ck.set_appearance_mode("dark")
```

- Now after doing this we will add the class label and inside that there are many criteria which we are following and also we are setting the height , width and text font and size and also we are giving it the color and padding on the x axis and then we are placing it by using the `place` function in which we are mentioning the coordinates in x and y axis and then we are configuring it with the text and using the `configure` function.
- Similarly do this for all the `counter label` , `probLabel` .

```
classLabel = ck.CTkLabel(window, height=40, width=120, 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, 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, font=(
```

```
    "Arial", 20), text_color="black", padx=10)
probLabel.place(x=300, y=1)
probLabel.configure(text='PROB')
```

- Now we will do this for the classBox , counterBox and ProbBox which we will create in the black color.
- The text color we will do is white.
- You can clearly see this below ⬇️

```
classBox = ck.CTkLabel(window, height=40, width=120, font=(
    "Arial", 20), text_color="white", fg_color="black")
classBox.place(x=10, y=41)
classBox.configure(text='0')
counterBox = ck.CTkLabel(window, height=40, width=120, font=(
    "Arial", 20), text_color="white", fg_color="black")
counterBox.place(x=160, y=41)
counterBox.configure(text='0')
probBox = ck.CTkLabel(window, height=40, width=120, font=(
    "Arial", 20), text_color="white", fg_color="black")
probBox.place(x=300, y=41)
probBox.configure(text='0')
```

- Now we will reset the counter and inside this we will create counter as global and initialize the counter with the value of 0 and this we will trigger with the button which we will create in our next step.

```
def reset_counter():
  global counter
  counter = 0
```

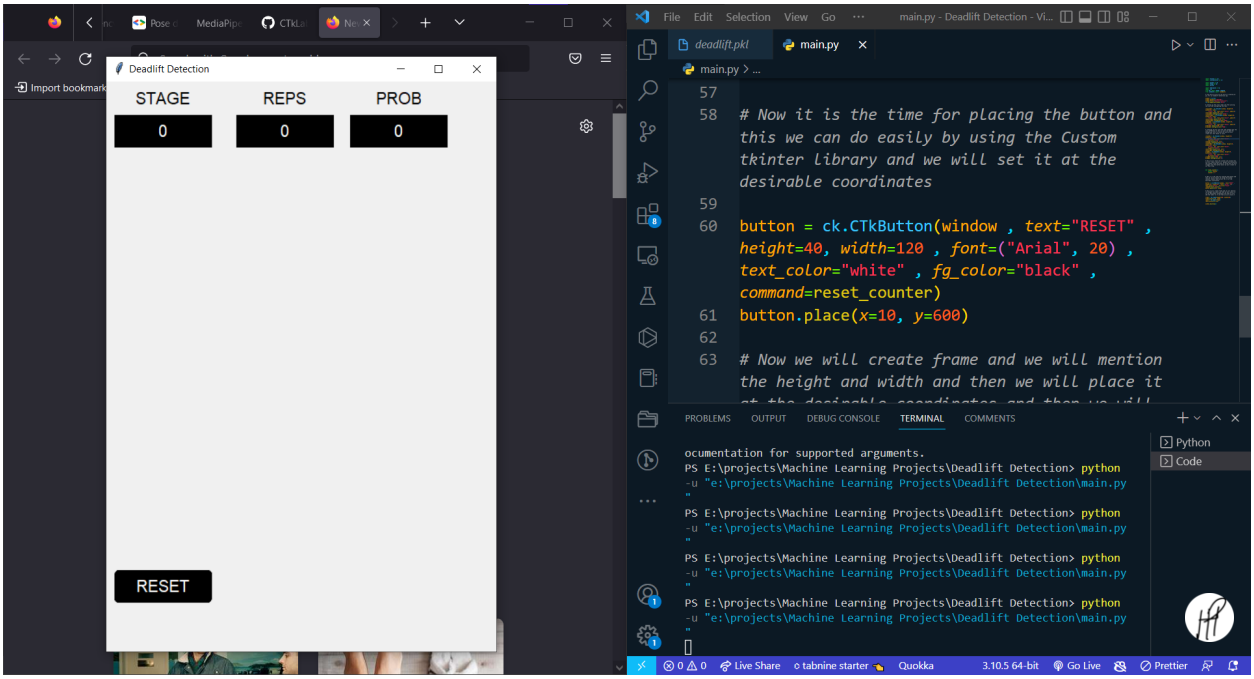
- Now it is the time for placing the button and this we can do easily by using the Custom Tkinter library and we will set it at the desirable coordinates.

```
button = ck.CTkButton(window , text="RESET" , height=40, width=120 , font=("Arial", 20) , text_color="white" , fg_color="black" , command=r
button.place(x=10, y=600)
```

- Now we will create frame and we will mention the height and width and then we will place it at the desirable coordinates and then we will add the label to it and then we will place it.

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

- Above you can actually see the GUI which we have created and this you can also see by adding the command `window.mainloop()` in your code and then you will notice the tkinter appearing in front of you with the GUI which we have developed but now it is time to create and initialize our machine learning model.



- Now from here we will use the mediapipe by google to draw_utils on our pose and then we will give the min_tracking_confidence and min_detection_confidence values as 0.5 in out test phase 1
- The code of execution for the following is given below :

```
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)
```

- As soon as again you will run this file you will notice one line in your output terminal where it is mentioned that " *Created TensorFlow Lite XNNPACK delegate for CPU* " Now the actual meaning of this file is that :
  - `XNNPACK` is a highly optimized library of floating point neural network inference operators for ARM , Web Assembly , and x86 Platforms and it is the default TensorFlow Lite CPU inferance engine for floating point models.
  - It enables best in-class performance on x86 and ARM CPU's.
  - It is over 10x faster then the default Tensorflow Lite backend in some cases.
- Now we will open the Pickle file and load it in model and this we will do by the following code :

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

- Now it is the time for cv2 to capture the video and then we will use the videoCapture(web_cam_no) to start capturing the video then we will make the current_stage , bodylang_class as empty string counter as 0 and in the bodylang_prob we will add the np.array([0,0]).

```
cap = cv2.VideoCapture(3)
current_stage = ''
counter =  0
bodylang_prob = np.array([0,0])
bodylang_class = ''
```

- Now we will create the detect function in which we are dividing the function in 3 steps in which we will step by step we will create and link our model with the tkinter and also we will add the try and exception to avoid the run time errors in our software so lets start it.
- In the first step we will declare the `current_stage` , `counter` , `bodylang_class` and `bodylang_prob` as global and then we will read from `cap.read()` function and store it in variable ret , frame.
- Then we will get the image by converting it from BGR to RGB and then we will process the image and store it in the results.
- Then from the mp_drawings which we have previously initiated in the code we will draw the landmarks on the image and then we will give it the specific color , thickness and circle radius and this we will do by the function in the mp_darwing which is `DrawingSpec` that you can easily notice and see in the given below code and this will be the end of the phase 1.

```
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_CONNNECTIONS,
    mp.drawing.DrawingSpec(color=(106,13,173) , thickness=4, circle_radius=5),
    mp.drawing.DrawingSpec(color=(255,102,0) , thickness = 5 , circle_radius= 10))
```

- Now it is the Phase 2 and here we will use the try block in which we will first add the condition in which we will add that if the bodylang_class is equal to down and bodylang_prob[bodylang_prob.argmax()] > 0.7 then we will update the current stage as “down” and in the elif block we will add that if the current stage is “down” and bodylang_class is up and bodylang_prob[bodylang_prob.argmax()] > 0.7 then we will update the current_stage as “up” and then we will update the counter value with 1.
- This is the end of the phase2 and its code is given below :

```
try:
        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)
```

- Now in the phase 3 we will add the image and we will do this from array and then we will add this image into the tkinter and then we will continue this step after 10 seconds we will detect the image and then we will put it into the tkinter GUI frame and code to do that is given below:

```
img  = image[:,:460,:]
    imgarr = Image.fromarray(img)
    imgtk = ImageTk.PhotoImage(imgarr)
    lmain.imgtk = imgtk
    lmain.configure(image=imgtk)
    lmain.after(109,detect)
```

- Now we are on the last phase of our complete project here we will dynamically update the values in the labels and all and for this first we will code the stuff in the try block above the if elif which we have coded previously and in that we will add the things like :
- First we will get the value in the row form x , y and visibility and then from results.pose_landmarks.landmark we will convert this into list and then we will store this in X in the form of the Dataframe.
- Then in the bodylang_class we will predict_probability and this predicted value will come from the X dataframe which we created in the previous step .

- Then in the bodylang_class we will again do the prediction and this time also the value will come from the X dataframe which we have created in out previous to previous step.
- Then in the end we will configure the text as counter.
- Then we will configure the (text = bodylang_prob[bodylang_prob.argmax()]) and then we will configure the classBox text as the current stage.
- Then we will call the detect function.

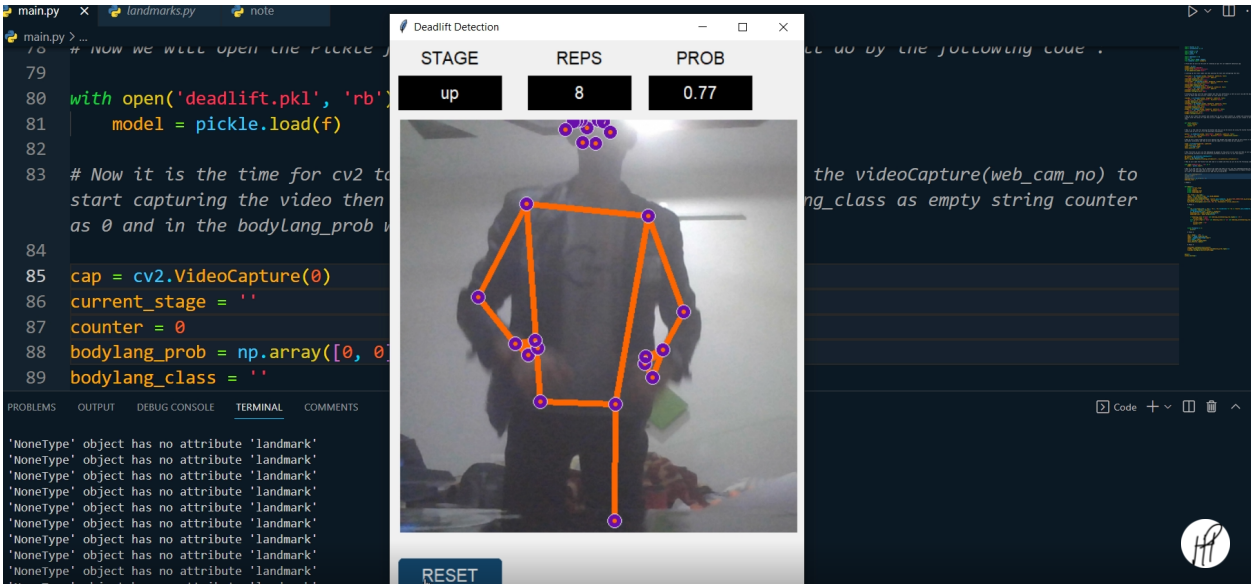
```
row = np.array([[res.x, res.y , res.visibility] for res in results.pose_landmarks.landmark]).flatten().tolist()
        X = pd.DataFrame([row], columns=landmarks)
        bodylang_prob = model.predict_proba(X)[0]
        bodylang_class = model.predict(X)[0]
```

- Enter the above code above the if else statement which we have coded previously in the try catch.
- Then in the end of the phase 4 enter this code to configure the values of the counter and etc.

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


detect()
```

- Now lets run our Machine Learning app for fun 🚀🚀.



CONCLUSION

- So as an conclusion machine learning as an wide variety of application and industry giants like google , amazon are creating there products to help the developers to build and give world class experience to the people.
- This app is created from an inspiration of the gym automation and all the dependencies are mentioned pervious also.
- You can find this project on the GitHub and its explanation on our YouTube channel.