WEEK 6 GROUP PROJECT

YOGA DATASET

DAQUEST GROUP

TABLE OF CONTENT

Introduction

Yoga is a type of exercise in which you move your body into various positions in order to become more fit or flexible.

The dataset is divided into train and test subdirectories, with 5 sub folders in each directory corresponding to the 5 classes of yoga poses.

5 CLASSES OF YOGA POSES

Tree

Goddess

Downdog

Warrior

Plank

IMAGE CLASSIFICATIONS

Image classification is a supervised learning problem define a set of target classes, and train a model to classify them using labeled example photos.

Image classifications

First model

- Read Train and Test path file

Read the train and test file

```
#Read the train and test file

train_dir = 'DATASET/TRAIN' #directory with training images
test_dir = 'DATASET/TEST' #directory with testing images
```

Image classifications

First model

- Rescale Train and Test images

Rescale the train and test images

In [229]:
```
# Rescale the train and test

train_datagen = ImageDataGenerator(width_shift_range= 0.1,
                                   horizontal_flip = True,
                                   rescale = 1./255,
                                   validation_split = 0.2)
test_datagen = ImageDataGenerator(rescale = 1./255,
                                  validation_split = 0.2)
```

In [179]:
```
train_generator =  train_datagen.flow_from_directory(directory = train_dir,
                                                     target_size = (224,224),
                                                     color_mode = 'rgb',
                                                     class_mode = 'categorical',
                                                     batch_size = 16,
                                                     subset = 'training')
validation_generator  = test_datagen.flow_from_directory(directory = test_dir,
                                                    target_size = (224,224),
                                                    color_mode = 'rgb',
                                                    class_mode = 'categorical',
                                                    subset = 'validation')
```

```
Found 866 images belonging to 5 classes.
Found 92 images belonging to 5 classes.
```

Image classifications

First model

- Create the model

Create the model

In [200]:

```
model = tf.keras.models.Sequential([
   # Using two filter in hidden layer with Acticition funcation 'relu'
    tf.keras.layers.Conv2D(64, (3,3), activation='relu',padding = 'Same', input_shape=(224, 224, 3)),
    #Use MaxPooling to reduce the dimensions of the feature map
    tf.keras.layers.MaxPooling2D(2, 2),
     #Use dropout layer to prevent the overfitting
    tf.keras.layers.Dropout(0.25),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu',padding = 'Same'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Dropout(0.25),


   #using the flatten is converting the data into a 1-dimensional array for inputting it to the next layer.

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(100, activation='relu'),
    tf.keras.layers.Dropout(0.5),

    # The output layer using the softmax funcation because our output contain of 5 classes
    tf.keras.layers.Dense(5, activation='softmax')
])
```

Image classifications

First model

- Compile the model

Compile the model

In [203]:
```
#Using 'adam' optimizer with 15 epoch

optimizer = Adam(lr=0.001)
model.compile(loss='categorical_crossentropy',
              optimizer = optimizer,
              metrics=['accuracy'],
             )
epochs = 15
batch_size = 16
```

- Fit the model

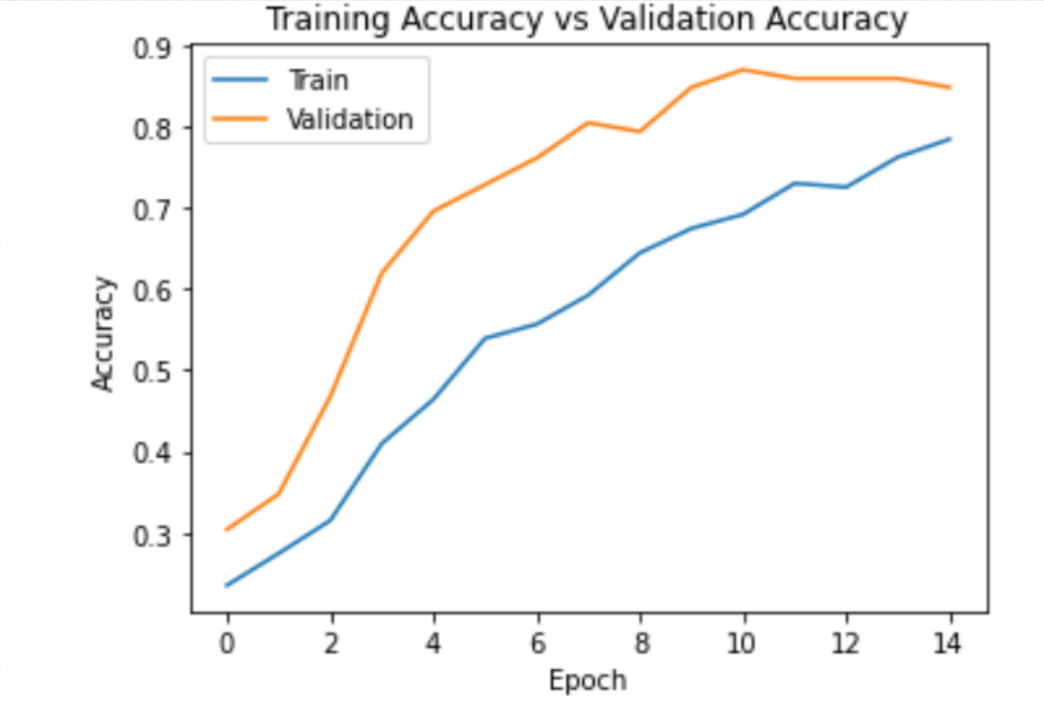
Fit the model

In [204]:
```
history = model.fit(train_generator, epochs = epochs,validation_data = validation_generator )
```

Image classifications

First model

- Accuracy chart



- Loss chart

Image classifications

First model

- Classification Report

```
In [195]: print(classification_report(y_val_org, predication_classes))
```

	precision	recall	f1-score	support
0	0.79	1.00	0.88	19
1	0.82	0.56	0.67	16
2	0.95	0.91	0.93	23
3	1.00	0.92	0.96	13
4	0.74	0.81	0.77	21
accuracy			0.85	92
macro avg	0.86	0.84	0.84	92
weighted avg	0.85	0.85	0.84	92

Image classifications

Second model

- Create the model

In [23]:
```
model = tf.keras.models.Sequential([
    # Three hidden layers using 'swish' activation function
    tf.keras.layers.Conv2D(32, (3,3), activation='swish',padding = 'Same', input_shape=(224, 224, 3)),
    # Max pooling to reduce the dimentions
    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Conv2D(64, (3,3), activation='swish',padding = 'Same'),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Conv2D(128, (3,3), activation='swish',padding = 'Same'),
    tf.keras.layers.MaxPooling2D(2,2),

    # Flatten layer to create one dimention array
    tf.keras.layers.Flatten(),
    # Fully connected layer
    tf.keras.layers.Dense(50, activation='swish'),
    tf.keras.layers.Dropout(0.5),
    # Output layer using 'softmax' activation function
    tf.keras.layers.Dense(5, activation='softmax')
])
```

Image classifications

Second model

- Compile the model

Compile the model

In [25]:
```
# Using 'adam' optimizer and 25 epochs
model.compile(loss='categorical_crossentropy',
              optimizer = 'adam',
              metrics=['accuracy'])

epochs = 25
batch_size = 16
```

- Fit the model

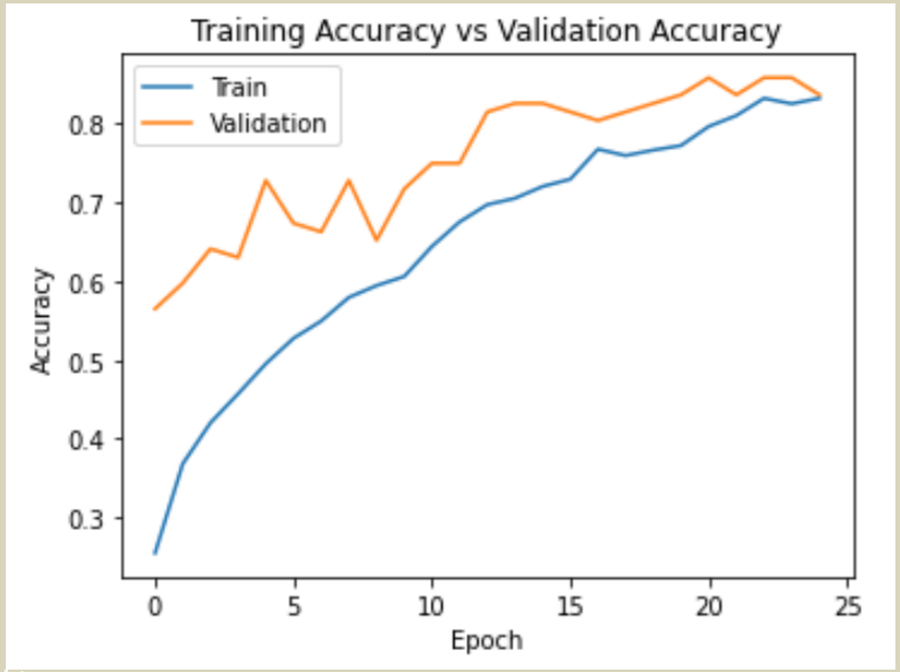
Fit the model

In [27]:
```
history = model.fit(train_generator, epochs = epochs,validation_data = validation_generator)
```

Image classifications

Second model

- Accuracy chart



- Loss chart

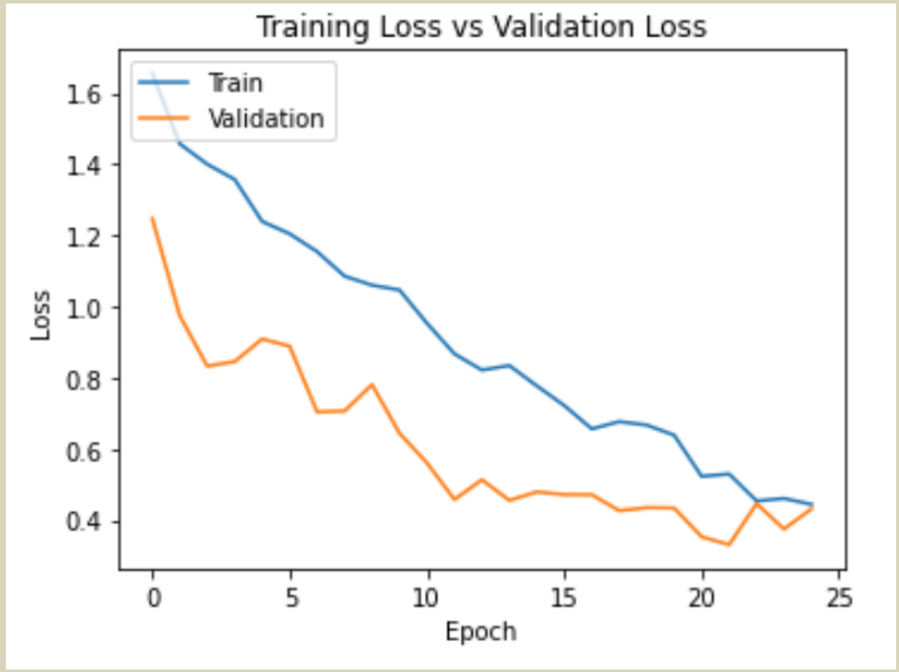


Image classifications

Second model

- Predication of validation

Prediction of validation

```
In [31]: predication = model.predict(validation_generator)

         predication
```

```
         3/3 [==============================] - 4s 1s/step

Out[31]: array([[2.06471197e-02, 5.99238157e-01, 1.98412806e-01, 2.27184352e-02,
                 1.58983439e-01],
                [1.35387927e-01, 4.16623568e-03, 8.24159026e-01, 3.82210070e-04,
                 3.59046161e-02],
                [9.75168943e-01, 4.09768109e-06, 2.48002205e-02, 2.36887740e-06,
                 2.42952938e-05],
                [2.00878200e-03, 1.67305768e-01, 1.00919949e-02, 1.15695084e-03,
                 8.19436550e-01],
                [1.60857769e-06, 2.66900301e-01, 5.70908742e-05, 1.87394799e-05,
                 7.33022273e-01],
                [9.99996781e-01, 9.96466132e-10, 3.22335563e-06, 3.87568276e-11,
                 4.51131132e-09],
                [8.17588170e-06, 4.76454757e-03, 2.10598003e-07, 2.65851785e-08,
                 9.95226979e-01],
                [2.39023883e-02, 4.84619915e-01, 5.33859283e-02, 1.41479671e-01,
                 2.96612054e-01],
```

Image classifications

Second model

- Classification Report

```
In [33]: print(classification_report(y_val_org, predication_classes))
```

	precision	recall	f1-score	support
0	0.90	1.00	0.95	19
1	0.67	0.50	0.57	16
2	0.88	0.96	0.92	23
3	1.00	0.85	0.92	13
4	0.74	0.81	0.77	21
accuracy			0.84	92
macro avg	0.84	0.82	0.83	92
weighted avg	0.83	0.84	0.83	92

Image classifications

Third model

- Create the model

Model

```
# Adding Dropout layer to prevent overfitting problem and  dense layer with relu Activation function
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(64, (3,3), activation='relu',padding = 'Same', input_shape=(224, 224, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Dropout(0.25),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu',padding = 'Same'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Dropout(0.25),
    tf.keras.layers.Conv2D(256, (3,3), activation='relu',padding = 'Same'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Dropout(0.25),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(1024, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    #output layer
    tf.keras.layers.Dense(5, activation='softmax')
])
```

Image classifications

Third model

- Compile the model

```
#Using 'adam' optimizer with 50 epoch
optimizer = Adam(lr=0.001)
model.compile(loss='categorical_crossentropy',
              optimizer = optimizer,
              metrics=['accuracy'])
epochs = 50
batch_size = 16
```

- Fit the model

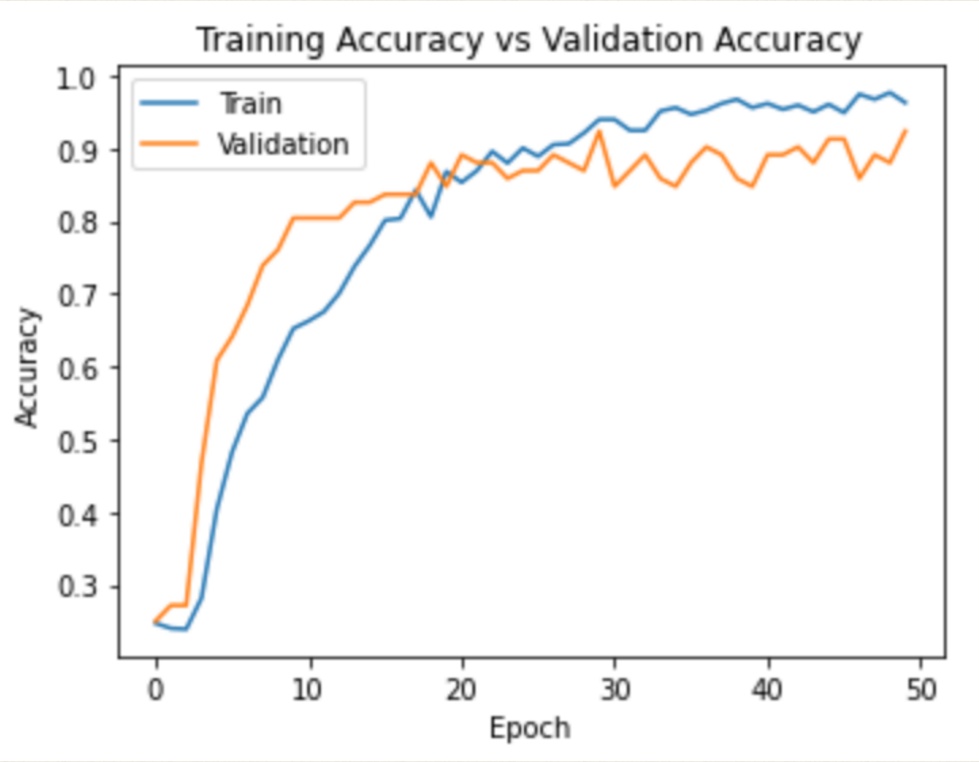
Fit the model

In [204]:
```
history = model.fit(train_generator, epochs = epochs,validation_data = validation_generator )
```

Image classifications

Third model

- Accuracy chart



- Loss chart

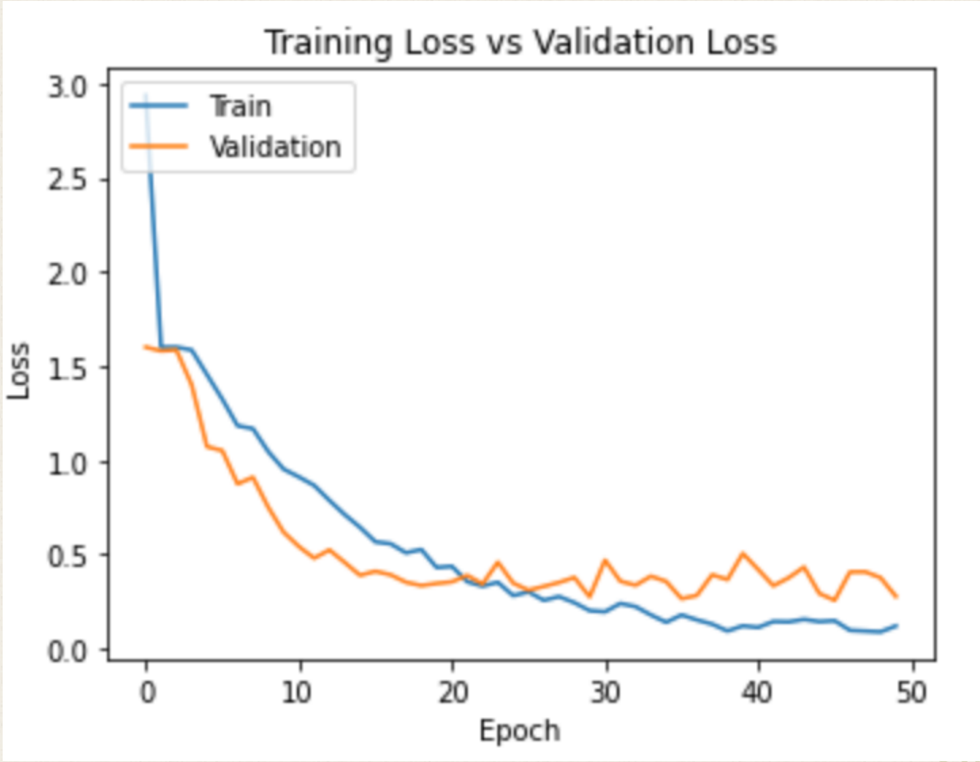


Image classifications

Third model

- Classification Report

```
In [67]: print(classification_report(y_val_org,prediction_classes))
```

	precision	recall	f1-score	support
0	0.90	0.95	0.92	19
1	0.92	0.75	0.83	16
2	0.96	1.00	0.98	23
3	1.00	1.00	1.00	13
4	0.86	0.90	0.88	21
accuracy			0.92	92
macro avg	0.93	0.92	0.92	92
weighted avg	0.92	0.92	0.92	92

Image classifications

Third model

- Testing

['downdog', 'tree', 'worrior2', ' goddess', 'plank']

In [256]:
```
from tensorflow.keras.preprocessing import image
from keras.models import load_model
from keras.applications.vgg16 import preprocess_input
img = image.load_img('/Users/fatimah/Desktop/DATASET-2/TEST/downdog/00000000.jpg', target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
img_data = preprocess_input(x)
classes = model.predict(img_data)
print(classes)
```

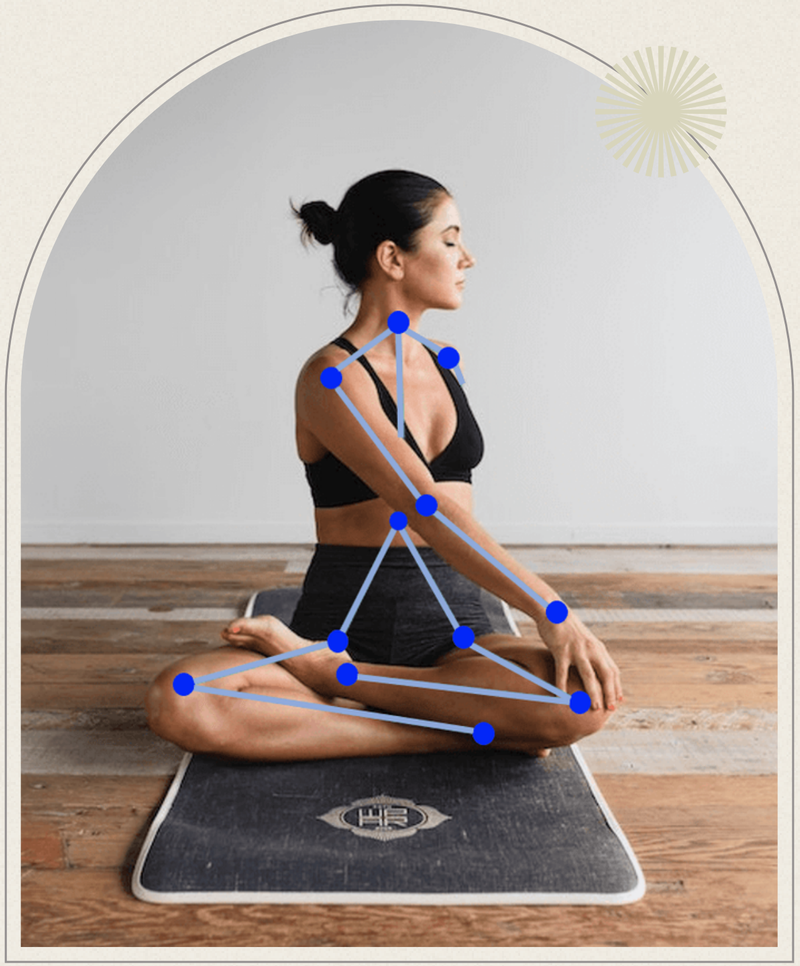
```
1/1 [==============================] - 1s 653ms/step
[[1. 0. 0. 0. 0.]]
```

In [257]:
```
img = image.load_img('/Users/fatimah/Desktop/DATASET-2/TEST/plank/00000000.jpg', target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
img_data = preprocess_input(x)
classes = model.predict(img_data)
print(classes)
```

```
1/1 [==============================] - 0s 146ms/step
[[0. 0. 1. 0. 0.]]
```

POSE DETECTION AND CLASSIFICATION

POSE DETECTION & POSE CLASSIFICATION WITH MEDIAPIPE



Pose Detection Goal

- Practicing Yoga in the wrong posture doesn't lead to a good result.
- Pose estimator is one of the real-world problems.

MediaPipe

Cross-platform library developed by Google that provides amazing ready-to-use ML solutions for computer vision tasks.

It employs machine learning (ML) to infer 3D landmarks from just a single frame

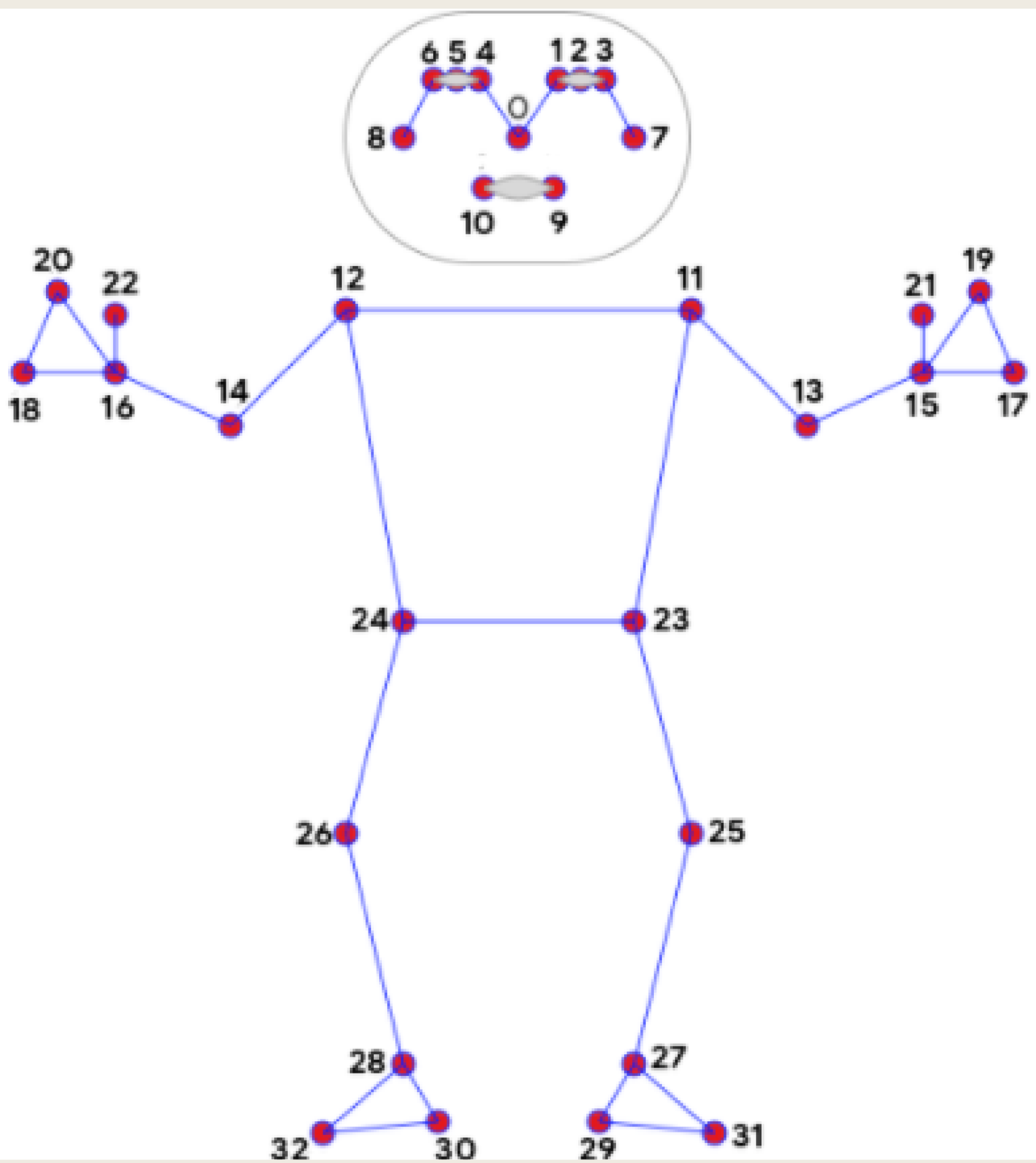
LANDMARKS

Pose Detection is a widely used computer vision task that enables you to predict humans poses in images or videos by localizing the key body joints (also referred as landmarks), these are elbows, shoulders, and knees, etc.

LANDMARKS

landmark detection is a computer vision task where we want to detect and track keypoints from a human

Keypoints are human skeleton joints on the body which on connecting resembles the human skeleton structure.



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

Pose Detection

Initialize the Pose Detection Model

```
# Initializing mediapipe pose class.
mp_pose = mp.solutions.pose

# Setting up the Pose function.
pose = mp_pose.Pose(static_image_mode=True, min_detection_confidence=0.3, model_complexity=2)

# Initializing mediapipe drawing class, useful for annotation.
mp_drawing = mp.solutions.drawing_utils
```

Read an Image

```
# Read an image from the specified path.
sample_img = cv2.imread('../DATASET/TRAIN/tree/File72.jpg')
```

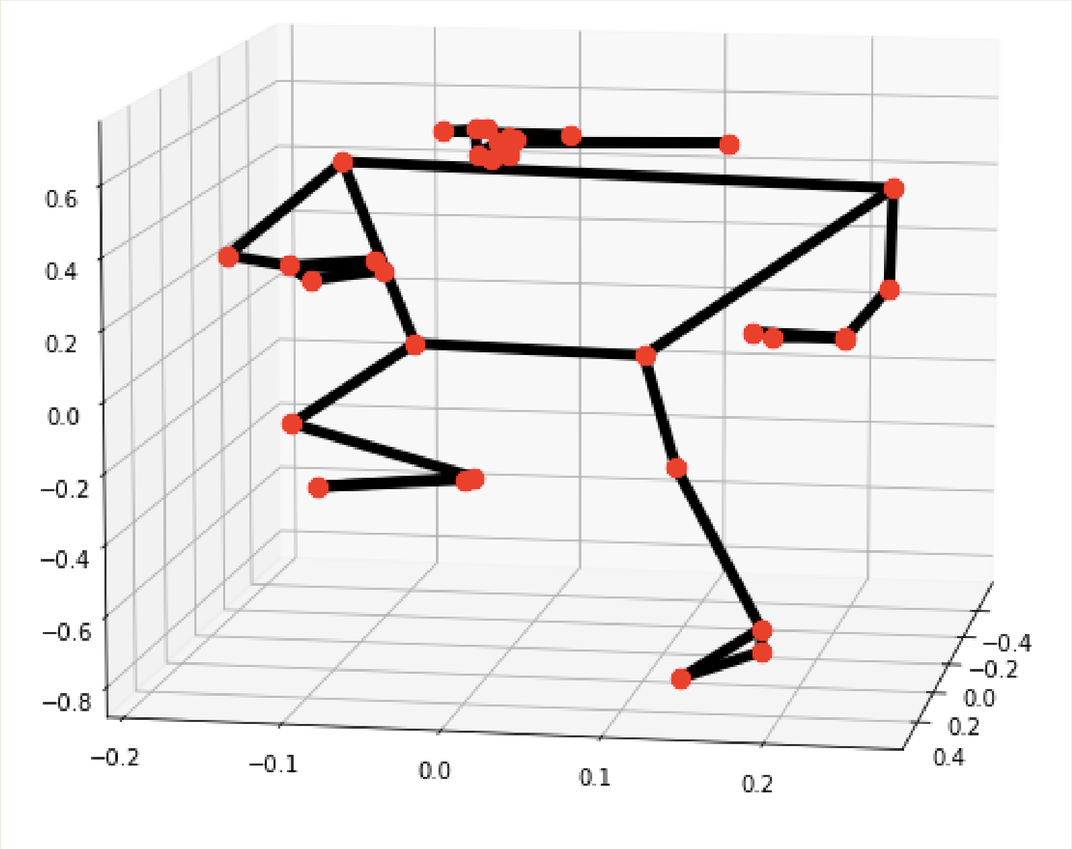
Create a Pose Detection Function

The function performs pose detection on an image.

Args:

image: The input image with a prominent person whose pose landmarks needs to be detected.

pose: The pose setup function required to perform the pose detection.



Create a Function to Calculate Angle between Landmarks

function calculates angle between three different landmarks.

Args:

landmark1: The first landmark containing the x,y and z coordinates.

landmark2: The second landmark containing the x,y and z coordinates.

landmark3: The third landmark containing the x,y and z coordinates.

Returns:

angle: The calculated angle between the three landmarks.

Create a Function to Perform Pose Classification

function classifies yoga poses depending upon the angles of various body joints.

Args:

landmarks: A list of detected landmarks of the person whose pose needs to be classified.

output_image: A image of the person with the detected pose landmarks drawn.

display: A boolean value that is if set to true the function displays the resultant image with the pose label written on it and returns nothing.

Returns:

output_image and label

```python
# Read a sample image and perform pose classification on it.
image = cv2.imread('media/warriorIIpose.jpg')
output_image, landmarks = detectPose(image, pose, display=False)
if landmarks:
    classifyPose(landmarks, output_image, display=True)
```

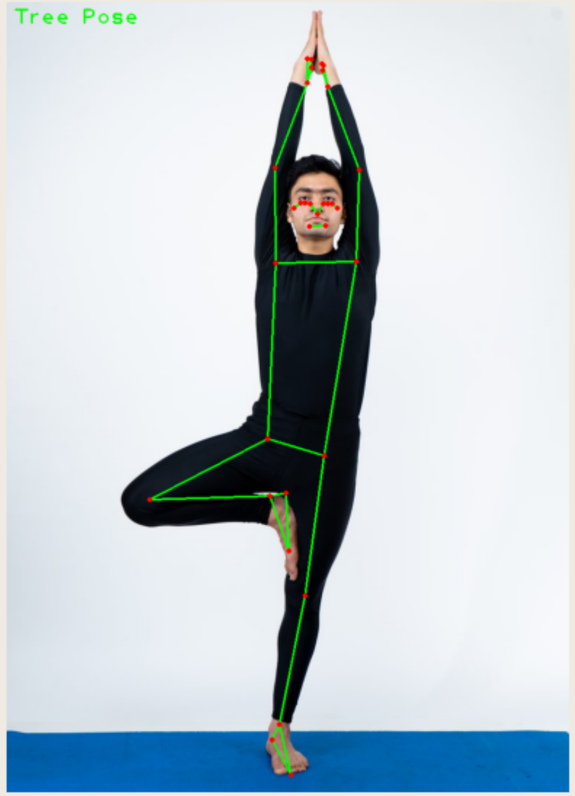
Output Image

Pose Classification On Video

test the function created above to perform the pose classification on a Video

Returns:

video and label.



Pose Classification On Video

```
def predict_video(model, video="0", show=False):
        cap = cv2.VideoCapture(video)
        while cap.isOpened():
                temp = []
                success, img = cap.read()
                if not success:
                        print("Ignoring empty camera frame.")
                        continue
                img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
                results = pose.process(img)
                if results.pose_landmarks:
                        landmarks = results.pose_landmarks.landmark
                        for j in landmarks:
                                temp = temp + [j.x, j.y, j.z, j.visibility]
                        y = model.predict([temp])
                        name = str(y[0])
                        if show:
                                mpDraw.draw_landmarks(img, results.pose_landmarks, mpPose.POSE_CONNECTIONS)
                                (w, h), _ = cv2.getTextSize(name, cv2.FONT_HERSHEY_SIMPLEX, 1, 1)
                                cv2.rectangle(img, (40, 40), (40+w, 60), (255, 255, 255), cv2.FILLED)
                                cv2.putText(img, name, (40, 60), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 0), 3)
                                cv2_imshow(img) #cv2.imshow("Video", img)
                                if cv2.waitKey(5) & 0xFF == 27:
                                        break
        cap.release()
```

Pose Classification On Video

Build the dataset using landmarks detection and save it as csv

Predict the name of the poses in the image

```
# Use this to evaluate any dataset you've built
def evaluate(data_test, model, show=False):
        target = data_test.loc[:, "target"]  # list of labels
        target = target.values.tolist()
        predictions = []
        for i in range(len(data_test)):
                tmp = data_test.iloc[i, 0:len(data_test.columns) - 1]
                tmp = tmp.values.tolist()
                predictions.append(model.predict([tmp])[0])
        if show:
                print(confusion_matrix(predictions, target), '\n')
                print(classification_report(predictions, target))
        return predictions
```

Predict Report

```
[[81  0  5  0  1]
 [ 0 53  3  6  5]
 [ 6  2 92  1  2]
 [ 0  8  0 53  4]
 [ 0 13  0  3 90]]
```

	precision	recall	f1-score	support
downdog	0.93	0.93	0.93	87
goddess	0.70	0.79	0.74	67
plank	0.92	0.89	0.91	103
tree	0.84	0.82	0.83	65
warrior2	0.88	0.85	0.87	106
accuracy			0.86	428
macro avg	0.85	0.86	0.85	428
weighted avg	0.87	0.86	0.86	428

Pose Prediction from Video (run)

THANK YOU

Any Questions?