Posture Recognition

- Aman Sharma B2021005
- Hazia Fernandes B2021021
- Luv Saxena B2021023
- Manali Hedaoo B2021025
- Sumit Grover B2021045

Introduction

POSE ELIMINATION

Pose estimation is a computer vision technique to track the movements of a person or an object.

This is usually performed by finding the location of key points for the given objects. Based on these key points we can compare various movements and postures and draw insights.

Pose estimation is actively used in the field of augmented reality, animation, gaming, and robotics.

3 Step Implementation Process

1

Human Body Landmark Detection

Using the tensorflow library **MoveNet** to detects 17 key points of a body

2

Training Model to Recognize Poses

Landmark Detected from MoveNet passed to the ANN in CSV format

3

Detection of Pose on Live Video

Live video is fed to the model frame by frame for pose detection

DEMO

MoveNet

MoveNet is an ultra fast and accurate model that detects **17 key points** of a body.

Model run faster than real time (30+ FPS) on most modern desktops, laptops, and phones, which proves crucial for live fitness, health, and wellness applications.

MoveNet - Working

1

Feature Extraction - Layers of Image

From Image that is input to the model the key features are extracted

2

Predict 17 Confidence Map

Map representing a particular part of the human pose skeleton.

3

Predict 38 Part Affinity Fields (PAFs)

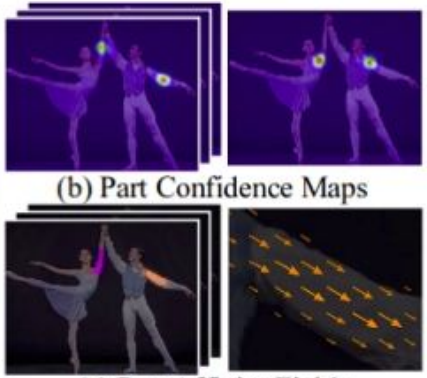
Fields represents the degree of association between parts.

4

Forming Bipartite Graphs of Parts

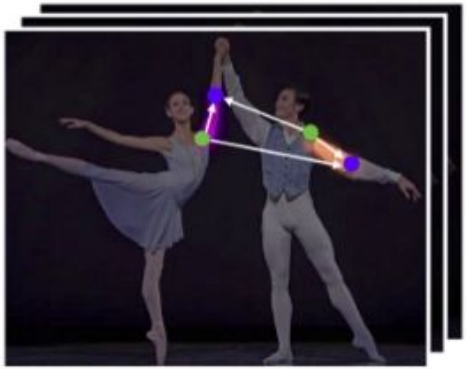
Using the part confidence maps the graph is formed

(a) Input Image



(b) Part Confidence Maps

(c) Part Affinity Fields



(d) Bipartite Matching

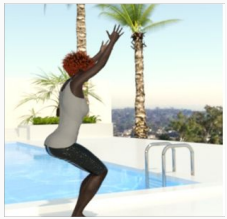
(e) Parsing Results

Project Implementation

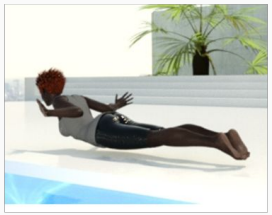
1. Training & Testing Dataset

- 8 different classes of yoga poses

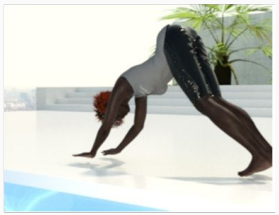
Warrior



chair

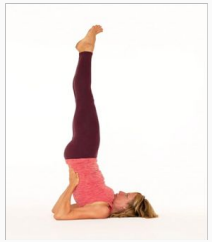


Cobra



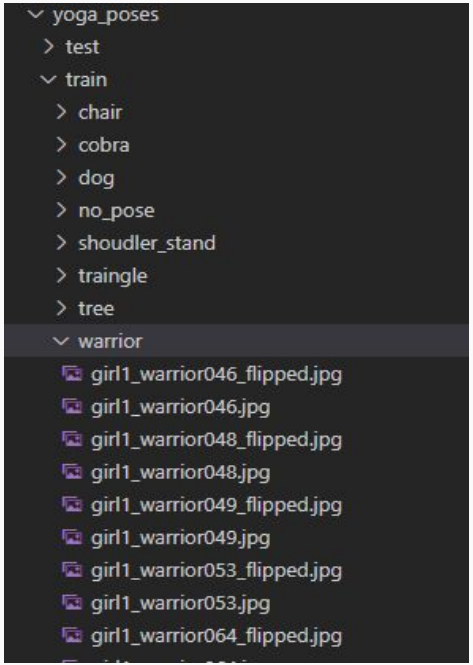
Dog

Tree

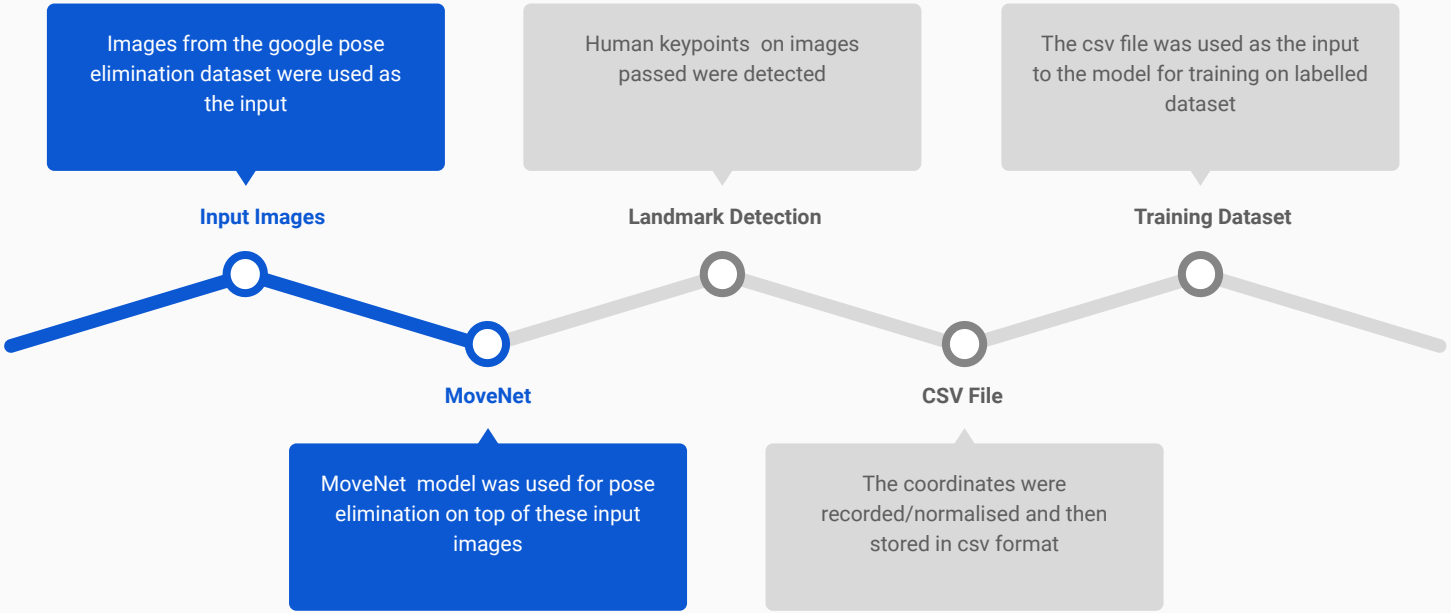


Shoulder stand

Triangle



2. Preprocessing Data



Preprocessing Data

```
1 processed_X_train[0]
```

```
<tf.Tensor: shape=(34,), dtype=float32, numpy=
array([-0.36180115, -0.00283766, -0.35612583, -0.01986359, -0.36180115,
       -0.02270125, -0.30221036, -0.03121422, -0.31639865, -0.03688953,
       -0.20856771,  0.00283766, -0.23126896, -0.06242843, -0.05817195,
        0.05107781, -0.131951  , -0.05107781, -0.14897694,  0.13620749,
       -0.20005475,  0.01986359,  0.01276945,  0.02270125, -0.01276945,
       -0.02270125,  0.19154178,  0.01986359,  0.12911335, -0.01702594,
        0.313561  , -0.02837656,  0.29653504, -0.04256484], dtype=float32)>
```

- chair
- cobra
- dog
- no_pose
- shoudler_stand
- traingle
- tree
- warrior

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	girl1_chair070.jpg	158	91	0.486214	159	86	0.567006	154	86	0.471732	155	87	0.576235	144	87	0.69681	158	106	0.655757	139	108	0.6429
2																						
3	girl1_chair070_flipped.jpg	138	91	0.602957	143	87	0.507938	139	87	0.73279	155	88	0.710499	147	87	0.680588	160	106	0.856527	141	105	0.8587
4																						
5	girl1_chair075.jpg	165	100	0.674502	165	95	0.771247	161	95	0.767396	160	97	0.489812	150	97	0.656828	162	115	0.698976	142	116	0.8431
6																						
7	girl1_chair075_flipped.jpg	134	101	0.498701	138	96	0.598085	134	95	0.545087	151	97	0.737462	143	97	0.542549	157	115	0.814292	137	115	0.6357
8																						
9	girl1_chair076.jpg	165	102	0.640302	166	99	0.595476	162	98	0.702521	160	100	0.5167	151	99	0.575117	164	116	0.743439	143	118	0.800
10																						
11	girl1_chair076_flipped.jpg	132	103	0.731163	136	98	0.692581	132	98	0.820119	148	99	0.695886	140	99	0.648976	155	118	0.748228	137	117	0.6813
12																						
13	girl1_chair080.jpg	175	108	0.582157	176	103	0.534049	171	103	0.642105	170	104	0.696472	159	103	0.75937	171	122	0.728444	149	124	0.8461
14																						

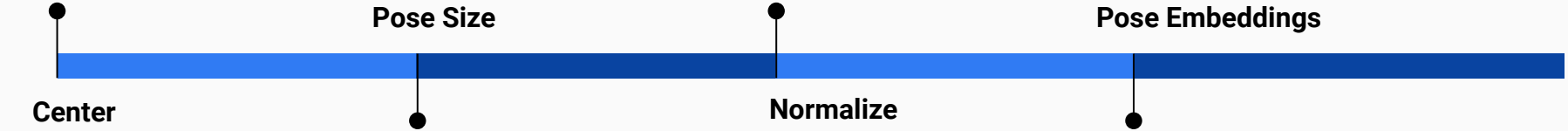
2. Model Training

Calculates the center point of the two given landmarks.

Helps in determining the pose and the coordinates in better manner with the links

Flattening the coordinates into a feature vector

Normalizes the landmarks translation by moving the pose center to (0,0) and scaling it to a constant pose size.



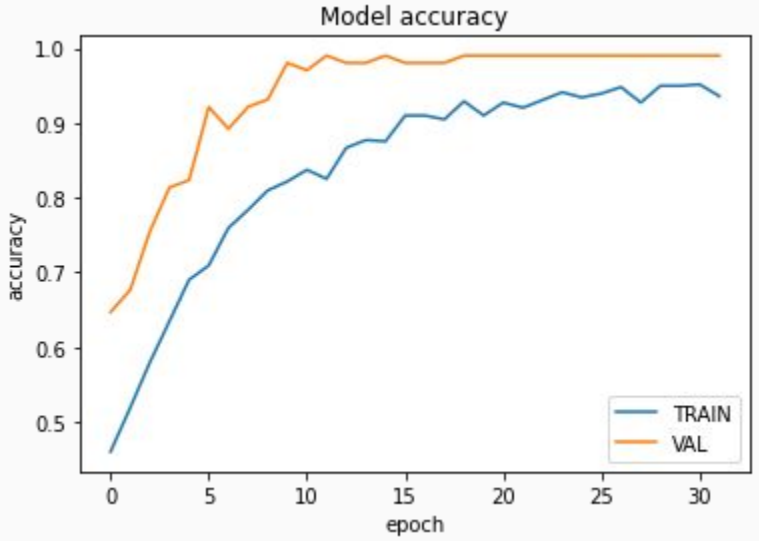
Scaling the pose so that the pose size becomes 1

Torso size multiplied by torso_size_multiplier` or the maximum distance from pose center to any pose landmark

Feature vector

Convert the pose landmarks to vectors for training the model in Pose Detection

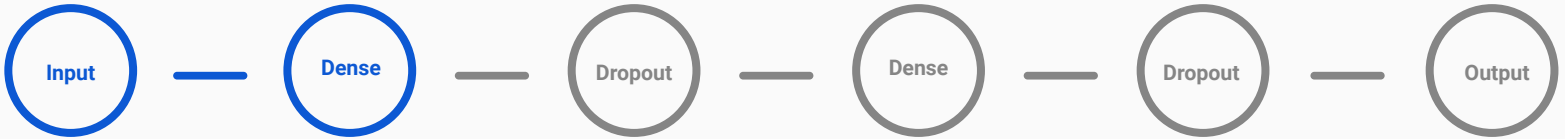
Model Training




```
]:  1 # Start training
    2 print('-------------TRAINING---------------')
    3 history = model.fit(processed_X_train, y_train,
    4                     epochs=2000,
    5                     batch_size=16,
    6                     validation_data=(processed_X_val, y_val),
    7                     callbacks=[checkpoint, earlystopping])
```

```
-------------TRAINING---------------
Epoch 1/2000
41/81 [==============>..............] - ETA: 0s - loss: 1.8939 - accuracy: 0.3338
Epoch 1: val_accuracy improved from -inf to 0.51542, saving model to weights.best.hdf5
81/81 [==============================] - 1s 3ms/step - loss: 1.6579 - accuracy: 0.4058 - val_loss: 1.2030 - val_accuracy: 0.5
154
Epoch 2/2000
80/81 [============================>.] - ETA: 0s - loss: 1.1163 - accuracy: 0.5781
Epoch 2: val_accuracy improved from 0.51542 to 0.71366, saving model to weights.best.hdf5
81/81 [==============================] - 0s 2ms/step - loss: 1.1158 - accuracy: 0.5779 - val_loss: 0.8631 - val_accuracy: 0.7
137
Epoch 3/2000
45/81 [===============>..............] - ETA: 0s - loss: 0.9422 - accuracy: 0.6250
Epoch 3: val_accuracy improved from 0.71366 to 0.77974, saving model to weights.best.hdf5
81/81 [==============================] - 0s 2ms/step - loss: 0.9106 - accuracy: 0.6449 - val_loss: 0.7218 - val_accuracy: 0.7
797
```

3. Artificial Neural Network



Input	Dense	Dropout	Dense	Dropout	Output
Input Shape = 34	**Nodes = 128**	**Dropout = 0.5**	**Nodes = 64**	**Dropout = 0.5**	**Nodes = 8**
17 Landmarks * 2 Coordinates	Activation Function = Rectified Linear Unit	Reducing overfitting in artificial neural networks	Activation Function = Rectified Linear Unit	Reducing overfitting in artificial neural networks	Predict multinomial probability distribution

Model Creation

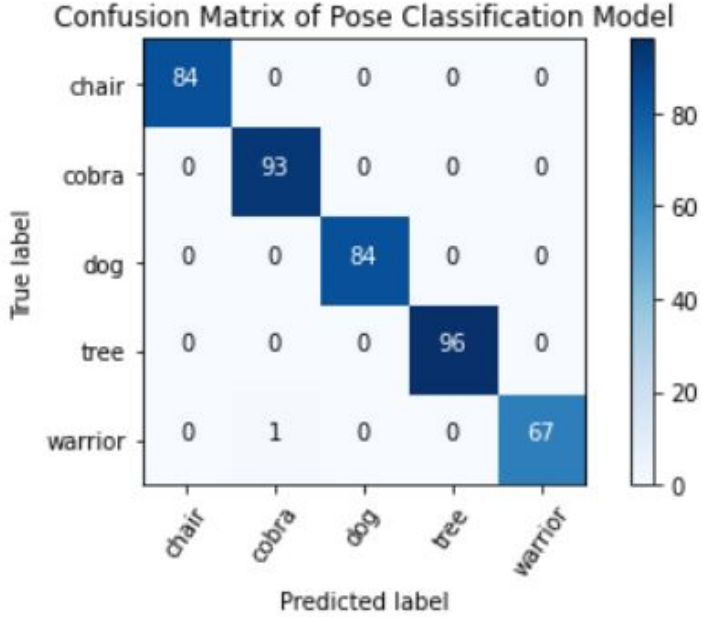
In [24]:
```
X, y, class_names = load_csv('train_data.csv')
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.15)
X_test, y_test, _ = load_csv('test_data.csv')


processed_X_train = preprocess_data(X_train)
processed_X_val =  preprocess_data(X_val)
processed_X_test = preprocess_data(X_test)

inputs = tf.keras.Input(shape=(34))
layer = keras.layers.Dense(128, activation=tf.nn.relu6)(inputs)
layer = keras.layers.Dropout(0.5)(layer)
layer = keras.layers.Dense(64, activation=tf.nn.relu6)(layer)
layer = keras.layers.Dropout(0.5)(layer)
outputs = keras.layers.Dense(len(class_names), activation="softmax")(layer)

model = keras.Model(inputs, outputs)
```

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

Model Performance Evaluation

Classification Report:

	precision	recall	f1-score	support
chair	1.00	1.00	1.00	84
cobra	0.99	1.00	0.99	93
dog	1.00	1.00	1.00	84
tree	1.00	1.00	1.00	96
warrior	1.00	0.99	0.99	68
accuracy			1.00	425
macro avg	1.00	1.00	1.00	425
weighted avg	1.00	1.00	1.00	425




```
-----------------EVAUATION----------------
28/28 [==============================] - 0s 995us/step - loss: 0.0059 - accuracy: 0.9977
LOSS:  0.005906759295612574
ACCURACY:  0.9977195262908936
```

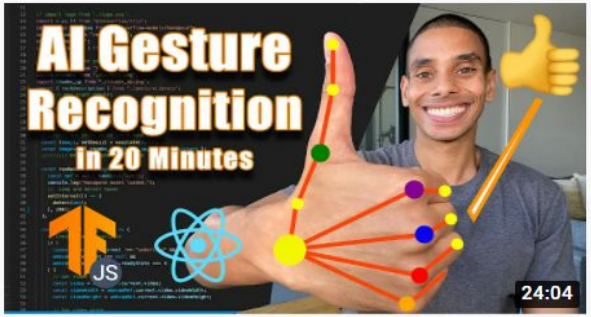
Inspirations & References

Sign Language Detection using ACTION RECOGNITION with Python | LSTM Deep Learning Model

69K views • 8 months ago

Nicholas Renotte

Want to take your sign language model a little further? In this video, you'll learn how to leverage action detection to do so! You'll ...



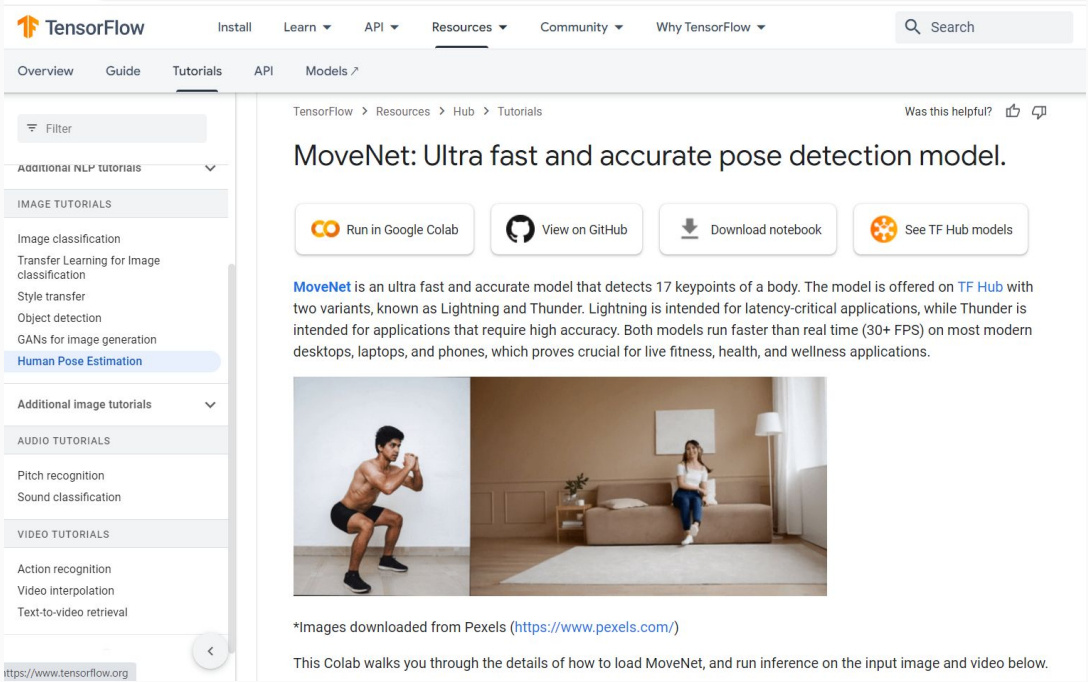
Real Time AI GESTURE RECOGNITION with Tensorflow.JS + React.JS + Fingerpose

33K views • 1 year ago

Nicholas Renotte

I'm talking deep learning powered computer vision based **gesture recognition**. Using Tensorflow.JS and fingerpose, you're able to ...

Inspirations & References



Special Thanks

Thanks!