Project Report

On

YOGA POSTURE RECOGNITION FOR SELF TRAINING

Submitted in partial fulfillment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

Submitted by

Abhinav Dubey
IT/56/15

Aakash Kumar Singh
IT/70/15

Under the supervision of

Mr. Janib ul Bashir

Department of Information Technology

Department of Information Technology

National Institute of Technology Srinagar, J&K

June 2019

CERTIFICATE

This is to certify that the project titled **Yoga Posture Recognition For Self Training** has been completed by **Abhinav Dubey (IT/56/15)** and **Aakash Kumar Singh (IT/70/15)** under my supervision in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Information Technology**, it is also certified that the project has not been submitted or produced for the award of any other degree.

Janib ul Bashir

Project Supervisor
Department of Information Technology
NIT Srinagar, J&K

Department of Information Technology

National Institute of Technology Srinagar, J&K

STUDENT'S DECLARATION

We, hereby declare that the work, which is being presented in the project entitled **Yoga Posture Recognition For Self Training** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Information Technology** in the session 2017, is an authentic record of our own work carried out under the supervision of **Mr. Janib ul Bashir** , Department of Information Technology, National Institute of Technology, Srinagar. The matter embodied in this project has not been submitted by us for the award of any other degree.

Dated: _____

(i) Name Abhinav Dubey

Signature _____

(ii) Name Aakash Kumar Singh

Signature _____

Department of Information Technology

National Institute of Technology Srinagar, J&K

ACKNOWLEDGEMENTS

We take this opportunity to express the deepest gratitude towards Mr. Janib ul Bashir , our project guide , who has been the driving force behind this project and whose guidance and cooperation has been a source of inspiration for us . We also thank all the other teachers of our department for their support and encouragement .

We are very thankful to our professors , colleagues and authors of various publications to which we have been referring to . We express our sincere appreciation and thanks to all those who have guided us directly or indirectly in our project . Also much needed moral support and encouragement was provided on numerous occasions by our whole division .

Finally , we thank our parents for their immense support .

Abhinav Dubey (IT/56/15)

Aakash Kumar Singh (IT/70/15)

ABSTRACT

Fitness exercises are very beneficial to personal health and fitness; however, they can also be ineffective and potentially dangerous if performed incorrectly by the user. Exercise mistakes are made when the user does not use the proper form, or pose. Detecting objects and estimating their pose remains as one of the major challenges of the computer vision research community . There exists a compromise between localizing the objects and estimating their viewpoints. The detector ideally needs to be view-invariant, while the pose estimation process should be able to generalize towards the category-level.

In our work, we introduce an application that detects the user's exercise pose and provides personalized, detailed on how the user can improve their performance . This project system uses the state of the art in pose estimation to detect a user's pose, then evaluates the vector geometry of the pose through an exercise to provide useful feedback. We record a dataset of about 8 exercise of correct and incorrect form, based on personal training guidelines, and build geometric-heuristic and machine learning algorithms for evaluation (detection and correction , if any).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

SVM	SUPPORT VECTOR MACHINE
CNN	CONVOLUTIONAL NEURAL NETWORK
DNN	DEEP NEURAL NETWORK

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Overview and Motivation :

Human by nature is vulnerable and subject to wide ranging health diseases of which musculoskeletal disorders is an important arena and needs urgent attention. Every year a wide range of people are affected from various types of musculoskeletal disorders due to accidents or aging problem. Yoga can promote positive physical change. Studies have shown that yoga is effective in managing symptoms associated with musculoskeletal disorders including osteoarthritis carpal tunnel syndrome, hyper kyphosis and low back pain. Additionally, Yoga shows significant improvements in motor skills and physiological methods that includes blood pressure, heart speed rate and human body weight have been noted .

Traditionally, yoga is done in a yoga center in the presence of a yoga trainer who can guide the patient through therapeutic assistance. Our solution suggests a distant yoga training approach without the trainer where the patient can stand in front of a system and perform yoga poses correctly without the need of trainer or being present at the yoga training center. Real time human pose detection is a growing and important area which specializes on realizing and understanding the human pose from depth images. We have set a reference model of true data for certain poses with respect to which we have detected the poses. The main challenge was the detection of body coordinate joints. Using the coordinates of PoseNet , the poses are detected. The purpose is to remotely monitor the yoga poses for patients. The system will work as a personal guidance system. The purpose of remotely monitoring yoga poses is to increase fitness with minimal monetary investment. As, at a certain time doctors or physicians can't look after all the patients, the will benefit both the patients and doctors in terms of efficiency.

1.2 Objectives :

Being an active field of research, there are no compilations of successful solution strategies and frequently used techniques for Human Pose Estimation yet. The aim of this thesis is, to fill this gap by analyzing few successful approaches. We have read the research papers on various object detection and human body detection to get thorough knowledge about the same so that we have a good visual presentation to describe the topic thoroughly . In a nutshell , the objective of this project is the detection and correction of the yoga postures performed by the practitioner and thus contribute to the feild of Real Time Human Pose Estimation .

The problem of human pose estimation, defined as the problem of localization of human joints, has enjoyed substantial attention in the computer vision community. In
Fig. 1, one can see some of the challenges of this problem – strong articulations, small and barely visible joints, occlusions and the need to capture the context. The main stream of work in this field has been motivated mainly by the first challenge, the need to search in the large
space of all possible articulated poses. Part-based models lend themselves naturally to model articulations and in the recent years a variety of models with efficient inference have been proposed . The above efficiency, however, is achieved at the cost of limited expressiveness – the use of local detectors, which reason in many cases about a single part, and most importantly by modeling only a small subset of all interactions between body parts. These limitations, as exemplified in Fig. 1, have been recognized and methods reasoning about pose in a holistic manner have been proposed but with limited success in real-world problems.

Chapter 2
LITERATURE REVIEW

2.1 Human Pose Estimation via Deep Neural Networks

A few previous works capitalize on recent developments of deep learning and propose a novel algorithm based on a Deep Neural Network (DNN). DNNs have shown good performance on visual classification tasks [1] and more recently on object localization . However, the question of applying DNNs for precise localization [2,3] of articulated objects has largely remained unanswered . It formulates the pose estimation as a joint regression problem and shows how to successfully cast it in DNN settings. The location of each body joint is regressed to using as an input the full image and a 7-layered generic convolutional DNN.

There are two disadvantages of this formulation. First, the DNN is incapable of capturing the full context of each body joint – each joint regressor uses the full image as a signal. Second, the approach is substantially complex to formulate than methods based on graphical models – no need to explicitly design feature representations and detectors for parts i.e. we need to explicitly design a model topology and interactions between joints. Further, the above mentoned work proposes a cascade of DNN-based pose predictors. Such a cascade allows for increased precision of joint localization. Starting with an initial pose estimation, based on the full image, we learn DNN-based regressors which refines the joint predictions by using higher resolution sub-images .

Fig 2.1 : Human Pose Estimation via Deep Neural Networks

2.2 2D and 3D Pose Estimation from a single image using Convolution Neural Network

Most state-of-the-art approaches for 2D human pose estimation employ CNN architectures [4], [5] . They can be divided into two groups: (a) methods which first search the image for local body parts and model their dependencies using graphical models [4], [5] and (b) holistic approaches that directly estimate the full body [6] . Methods based on local body parts require a tight bounding

box around each human to estimate his pose , others can detect multiple people in natural images at once . Most methods extract joint heatmaps, i.e., probabilistic maps that estimate the probability of each pixel to contain a particular joint. An iterative procedure is often used a refined estimate of the heatmaps is obtained from the previous estimate and the convolutional features. Joint positions can be estimated by taking the local maxima of the heatmaps. In Convolutional Pose Machines, Wei et al. [7] refine the predictions over successive stages with intermediate supervision at each stage. In the Stacked Hourglass network [8] , repeated bottom-up, top-down processing used in conjunction with intermediate supervision improve the performance of the network . Given the joint positions extracted from the heatmaps, additional post-processing is required to build human poses, such as graph partitioning . Cao et al. [4] proposed an alternative approach by also regressing affinities between joints, i.e., the direction of the bones, together with the heatmaps. In contrast to these methods that build human poses from

local body parts, our method extract full-body 2D and 3D poses, even in case of occlusions. Holistic approaches often assume that the individuals have been localized, and that a bounding box around each person is available. Toshev and Szegedy [6] directly regress the positions for each joint using an iterative procedure. Fan et al.[9] combines the local appearance with an holistic view of the body to estimate the position of the joints. Instead of relying on a multi-stage approach, our network is trained in an end-to-end fashion and outputs both 2D and 3D poses jointly.

Methods for 3D human pose estimation from a single image can be decomposed into two groups: (a) the ones that first compute 2D poses and use them to estimate 3D poses and (b) approaches that directly learn mappings from image features to 3D poses. Motivated by the recent advances in 2D pose detection, a large body of work tackles 3D pose estimation from 2D poses assuming that the 2D joints are available [10], provided by an off-the-shelf 2D pose detector [11] or obtained through a 2D pose estimation module within the proposed architecture . Most of these methods reason about geometry. Chen and Ramanan[12] estimate 3D pose from 2D through a simple nearest neighbor search on a given 3D pose library with a large number of 2D projections. Moreno- Noguer formulates the problem as a 2D-to-3D distance matrix regression. Nie et al. [13] predict the depth of human joints based on their 2D locations using LSTM, whereas Martinez et al.[14] lift 2D joints to 3D space using a simple, fast and lightweight deep neural network. These methods remain limited by the performance of the 2D pose estimator. Some other approaches directly estimate the 3D pose from image features . Recently, this has been naturally extended to end-to-end mappings using CNN architectures, either in monocular images or in videos . Pavlakos et al.[15] propose a volumetric representation for 3D human pose and employ a ConvNet to predict per-voxel likelihoods for each joint. In a structure-aware regression approach is followed with a reparameterized pose representation using bones instead of joints .

2.3 Single person pose estimation

Over the past few years, many CNN based methods in single person pose estimation [16], [17],[18] used very deep networks. Also, a recursive methodology has been adopted in many competitive methods. Newell et al. [17] proposed a model with multiple hourglass modules that repeats bottom-up and top-down processing and Wei et al. [19] proposed a convolution version of the pose machine which has been proposed by Ramakrishna et al. Features in these networks possess a large receptive field which extracts an efficient representation of human context. Advances in the single person pose estimation have made it possible to proceed research on multi-person pose estimation

2.4 Multi person pose estimation

Multi-person pose estimation [3],[18], [20] methods can be categorized as top-down and bottom-up approaches. The top-down approaches [21] firstly detect a person's bounding box and estimate single pose on the extracted bounding box. On the other hand, the bottom-up approaches [16] firstly detect parts of people and then determine poses in the input image by connecting the candidate parts. Cao et al. [4] proposed part affinity fields (PAFs) to associate body parts and determined the pose using the PAFs. The performances of stateof-the-art multi-person pose estimation methods are pretty good for a single frame. However, to apply the methods on real applications, we need to combine them with tracking algorithms for video data.

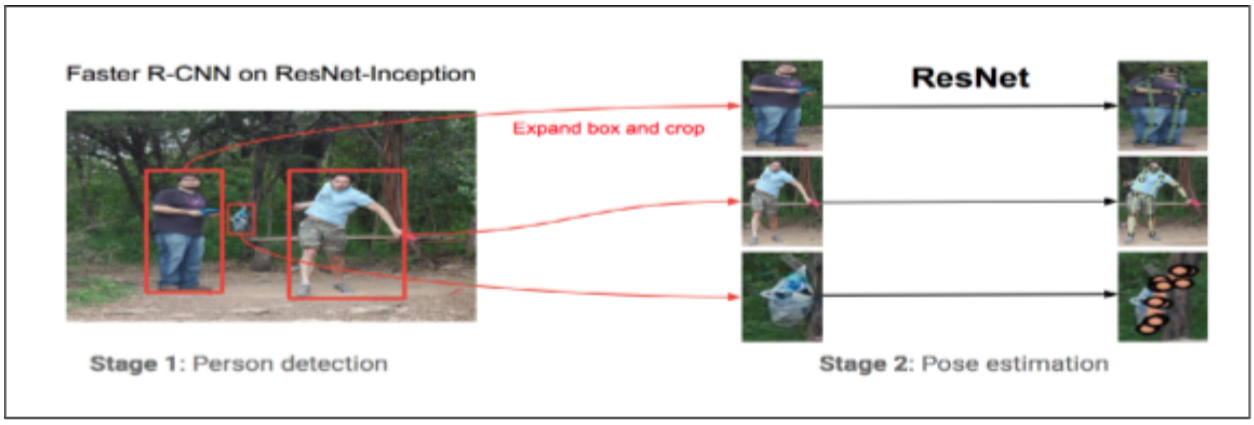

Fig 2.2 : Multi-person Pose Estimation

2.5 Human pose estimation with tracking

Several methods have been proposed to estimate and track human poses on videos [4], [16], [21] . These methods can be divided into two groups depending on whether the learned temporal information is used or not. For the methods that do not use the learned temporal information, they track the pose by applying optical flow, box tracking algorithm, and so on. Xiu et al. [24] proposed a pose tracker based on a pose flow that is a flow structure indicating the same person in different frames by pose distance. Xiao et al. [22] tracked the pose to use a flow-based pose tracking algorithm based on box propagation using optical flow and a flow-based pose similarity. Instead of naively connecting the relationships between detected poses, several papers trained sequential information. Radwan et al. [23] used a bi-directional long-short term memory (LSTM) framework to learn the consistencies of the human body shapes. Doering proposed temporal flow fields that are vector fields to indicate the direction of joints. However, tracking only a single joint may not contain enough temporal information due to lack of representation power

and also it may be vulnerable to occlusion of joints. Therefore, in this paper, instead of a single joint point, a limb connecting two adjacent joints is tracked, which is expected to resolve the

above mentioned problems. Also, during both training and testing, we consider a pair of frames with more than one time interval rather than only using two consecutive frames for the robustness of the proposed architecture.

Chapter 3
BACKGROUND

This chapter contains the information related to everything that is required to understand the project . It includes description about PoseNet and other libraries we have used in our project and the reason of usage

3.1 PoseNet

PoseNet [25] is a pre-trained machine learning library that can estimate human body coordinates . It was released by Google Creative Lab, and built on Tensorflow.js. It's powerful and fast enough to estimate human poses in real time, and works entirely in the browser. Even better, it has a relatively simple API. Posenet will run your image through a machine learning model and output an object (or an array of objects, if multiple) . Our object contains a score (how confident the model is about the pose estimation), and an array of 17 key points that correspond to different points on the body, which all have x and y coordinates. Each keypoint comes with its own confidence score. We only render the points that Posenet is relatively sure of . To be clear, this technology is not recognizing who is in an image. The algorithm is simply estimating where key body joints are.

The key points detected are indexed by "Part ID", with a confidence score between 0.0 and 1.0 where 1.0 being the highest.

TABLE 3.1: ID(S) OF THE CORDINATES RETURNED BY POSENET

ID	Body Part
0	nose
1	Left Eye
2	Right Eye
3	Left Ear
4	Right Ear
5	Left Shoulder
6	Right Shoulder
7	Left Elbow
8	Right Elbow
9	Left Wrist
10	Right Wrist
11	Left Hip
12	Right Hip
13	Left Knee
14	Right Knee
15	Left Ankle

16	Right Ankle

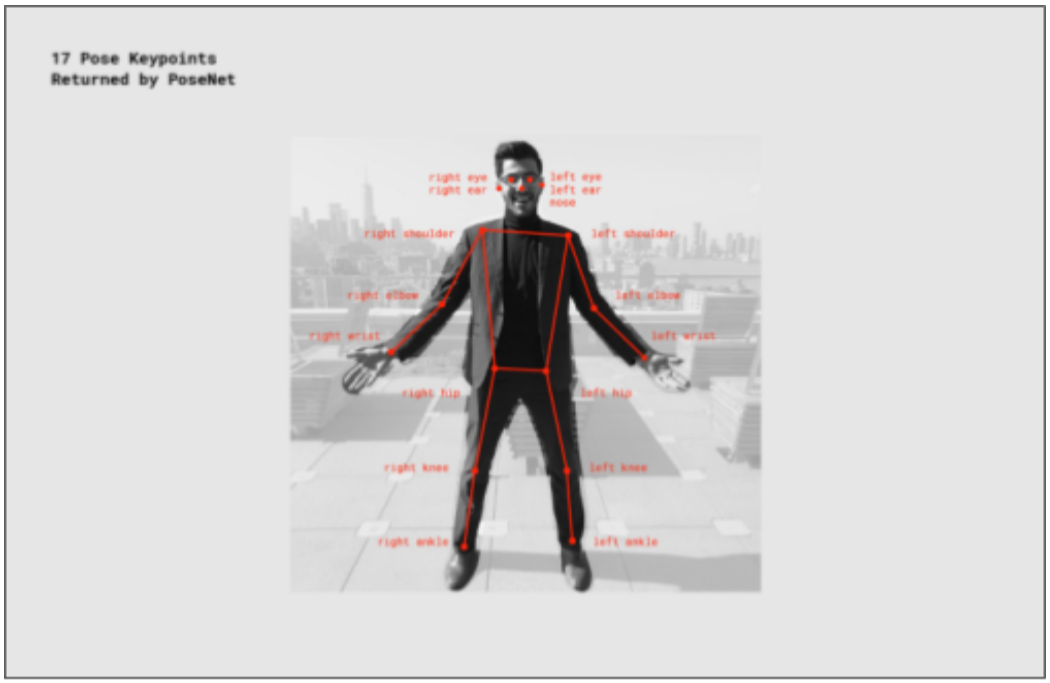

Fig 3.1 : Body coodinates derived from PoseNet

3.2 Performance of PoseNet

Performance varies based on your device and output stride (heatmaps and offset vectors). The PoseNet [25] model is image size invariant, which means it can predict pose positions in the same scale as the original image regardless of whether the image is downscaled. This means PoseNet can be configured to have a higher accuracy at the expense of performance.

Fig 3.2 : PoseNet returns confidence values for each person detected as well as each pose keypoint detected

The output stride determines how much we're scaling down the output relative to the input image size. It affects the size of the layers and the model outputs. The higher the output stride, the smaller the resolution of layers in the network and the outputs, and correspondingly their accuracy. In this implementation, the output stride can have values of 8, 16, or 32. In other words, an output stride of 32 will result in the fastest performance but lowest accuracy, while 8 will result in the highest accuracy but slowest performance. We recommend starting with 16.

At a high level pose estimation happens in two phases:

1. An input RGB image is fed through a convolutional neural network.

2. Either a single-pose or multi-pose decoding algorithm is used to decode coordinates , *pose confidence scores*, *keypoint positions,* and *keypoint confidence scores* from the model outputs.

Let's review the most important ones:

- **Pose** — at the highest level, PoseNet will return a pose object that contains a list of keypoints and an instance-level confidence score for each detected person.
- **Pose confidence score** — this determines the overall confidence in the estimation of a pose. It ranges between 0.0 and 1.0. It can be used to hide poses that are not deemed strong enough.
- **Keypoint** — a part of a person's pose that is estimated, such as the nose, right ear, left knee, right foot, etc. It contains both a position and a keypoint confidence score. PoseNet currently detects 17 keypoints illustrated in the following diagram:
- **Keypoint Confidence Score** — this determines the confidence that an estimated keypoint position is accurate. It ranges between 0.0 and 1.0. It can be used to hide keypoints that are not deemed strong enough.
- **Keypoint Position** — 2D x and y coordinates in the original input image where a keypoint has been detected.

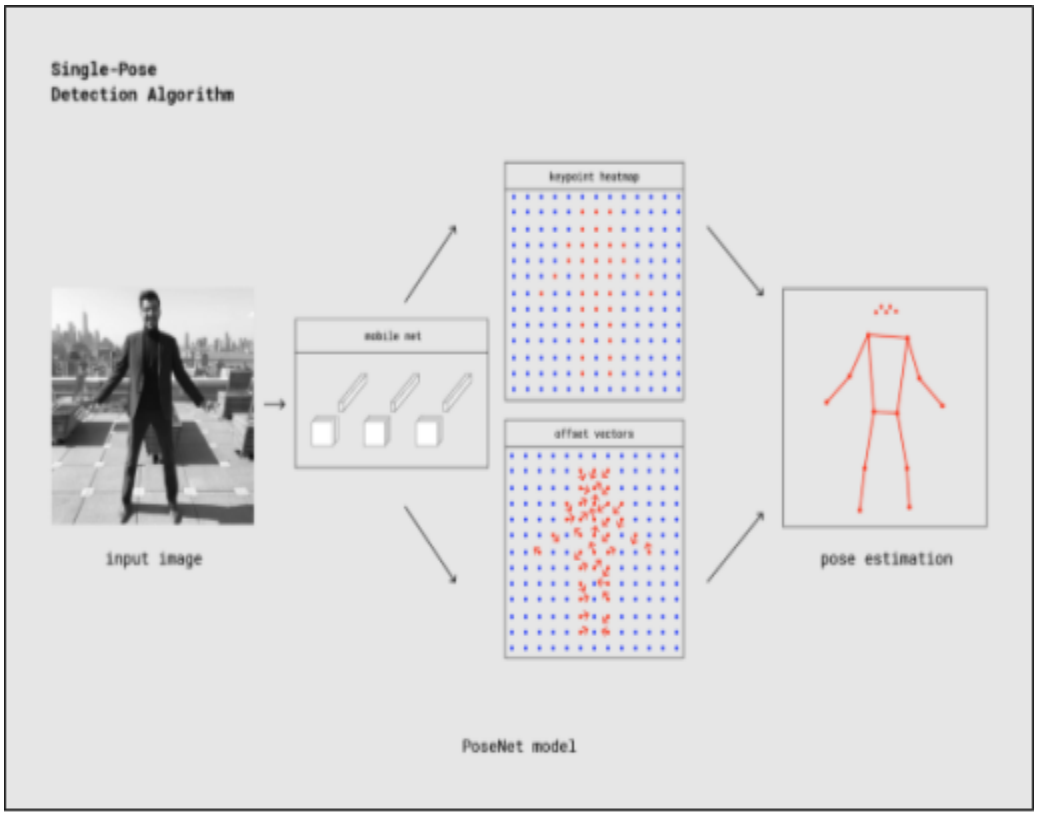

Fig 3.3 : Single person pose detector pipeline using PoseNet

3.3 Importing the TensorFlow.js and PoseNet Libraries

A lot of work went into abstracting away the complexities of the model and encapsulating functionality into easy-to-use methods. Let's go over the basics of how to setup a PoseNet project . Imported using javascript as :

```
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs"></script>
```

```
<script src="https://cdn.jsdelivr.net/npm/@tensorflow-models/posenet"></script>
```

This short code block shows how to use the single-pose estimation algorithm:

```
function mainpose(){
        console.log("start");
        var imageScaleFactor = 0.5;
        var outputStride = 16;
        var flipHorizontal = false;

        var imageElement = document.getElementById('cat');

        posenet.load().then(function(net){
          var poses = net.estimatePoses(imageElement, {
        flipHorizontal: flipHorizontal,
        decodingMethod: 'single-person'
      });
      var pose = poses[0];
      return poses;
        }).then(function(pose){
        pose=pose[0];
        console.log(pose[0]);
}
```

An example output pose looks like the following:

```
{
  "score": 0.32371445304906,
  "keypoints": [
    {
```

```
      "position": {
        "y": 76.291801452637,
        "x": 253.36747741699
      },
      "part": "nose",
      "score": 0.99539834260941
    },
    {
      "position": {
        "y": 71.10383605957,
        "x": 253.54365539551
      },
      "part": "leftEye",
      "score": 0.98781454563141
    },
     .
     .
     .
     .
    {
      "position": {
        "y": 347.41177368164,
        "x": 203.88229370117
      },
      "part": "rightAnkle",
      "score": 0.8255187869072
    }
  ]}
```

3.4 Flask

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Extensions are updated far more regularly than the core Flask program.

Applications that use the Flask framework include Pinterest , LinkedIn and the community web page for Flask itself.

Flask was designed to be easy to use and extend. The idea behind Flask is to build a solid foundation for web applications of different complexity. From then on you are free to plug in any extensions you think you need. Also you are free to build your own modules. Flask is great for all kinds of projects. It's especially good for prototyping. Flask depends on two external libraries: the Jinja2 template engine and the Werkzeug WSGI toolkit.

3.5 Pandas

Pandas [26] is a high-level data manipulation tool developed by Wes McKinney. It is built on the Numpy package and its key data structure is called the DataFrame. DataFrames allow you to store and manipulate tabular data in rows of observations and columns of variables.There are several ways to create a DataFrame. One way to create a DataFrame is by importing a csv file using Pandas. Now, the csv cars.csv is stored and can be imported using pd.read_csv:

```
# Import pandas as pd

import pandas as pd
```

```
# Import the cars.csv data: cars

cars = pd.read_csv('cars.csv')

# Print out cars

print(cars)
```

3.6 Numpy

NumPy [27] is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

At the core of the NumPy package, is the *ndarray* object. This encapsulates *n*-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:

- NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an *ndarray* will create a new array and delete the original.
- The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements.

3.7 Sklearn

sklearn (formerly **scikits.learn**) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms

including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

- **sklear.model_selection[28] :** Split arrays or matrices into random train and test subsets Quick utility that wraps input validation and next(ShuffleSplit().split(X, y)) and application to input data into a single call for splitting (and optionally subsampling) data in a oneliner.

- **Sklearn.preprocessing [29] :** This package provides several common utility functions and transformer classes to change raw feature vectors into a representation that is more suitable for the downstream estimators. In general, learning algorithms benefit from standardization of the data set. If some outliers are present in the set, robust scalers or transformers are more appropriate. The behaviors of the different scalers, transformers, and normalizers on a dataset containing marginal outliers is highlighted in Compare the effect of different scalars on data with outliers .

- **Sklearn.neighbours[30] :** provides functionality for unsupervised and supervised neighbors-based learning methods. Unsupervised nearest neighbors is the foundation of many other learning methods, notably manifold learning and spectral clustering. Supervised neighbors-based learning comes in two flavors: classification for data with

discrete labels, and regression for data with continuous labels. The principle behind nearest neighbor methods is to find a predefined number of training samples closest in distance to the new point, and predict the label from these. The number of samples can be a user-defined constant (k-nearest neighbor learning), or vary based on the local density of points (radius-based neighbor learning).

3.8 google_image_download [31] : This program lets you download tons of images from Google. Google Images is a search engine that merely indexes images and allows you to find them. It does NOT produce its own images and, as such, it doesn't own copyright on any of them. The original creators of the images own the copyrights.

Images published in the United States are automatically copyrighted by their owners, even if they do not explicitly carry a copyright warning. You may not reproduce copyright images without their owner's permission, except in "fair use" cases, or you could risk running into lawyer's warnings, cease-and-desist letters, and copyright suits. Please be very careful before its usage! Use this script/code only for educational purposes.

3.9 glob : Many a times a program needs to iterate through a list of files in the file system, often with names matching a pattern. The glob module is useful in creating lit of files in specific directory, having a certain extension, or with a certain string as a part of file name.

The pattern matching mechanism used by glob module functions follows UNIX path expansion rules. This module though doesn't expand tilde (~) and shell variables.

In order to use glob() and related functions we need to import the glob module. Keep in mind that glob module contains glob() and other related functions.

3.10 Support vector machine (SVM) :

"Support Vector Machine" (SVM) [32] is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well .

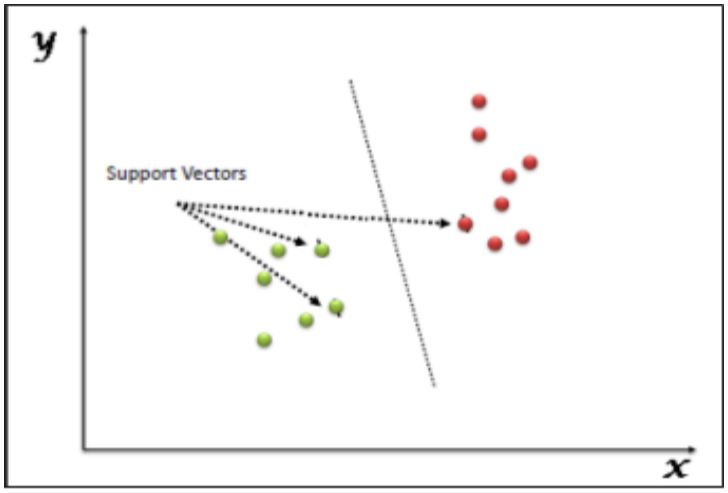

Fig 3.4 : Basics of support vectors

Support Vectors are simply the co-ordinates of individual observation. Support Vector Machine is a frontier which best segregates the two classes (hyper-plane/ line).

3.11 Working of SVM :

Above, we got accustomed to the process of segregating the two classes with a hyper-plane. Now the burning question is “How can we identify the right hyper-plane?” Let’s understand :

- **Identify the right hyper-plane (Scenario-1):** Here, we have three hyper-planes (A, B and C). Now, identify the right hyper-plane to classify star and circle.

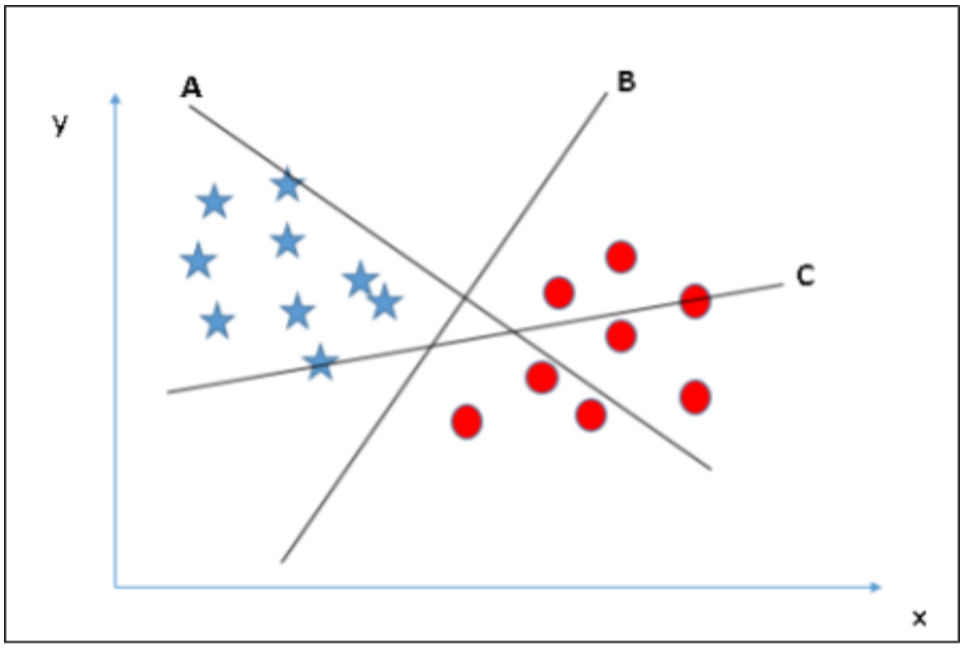

Fig 3.5 : SVM Plane Detection -1

You need to remember a thumb rule to identify the right hyper-plane: “Select the hyper-plane which segregates the two classes better”. In this scenario, hyper-plane “B” has excellently performed this job.

- **Identify the right hyper-plane (Scenario-2):** Here, we have three hyper-planes (A, B and C) and all are segregating the classes well. Now, How can we identify the right hyper-plane?

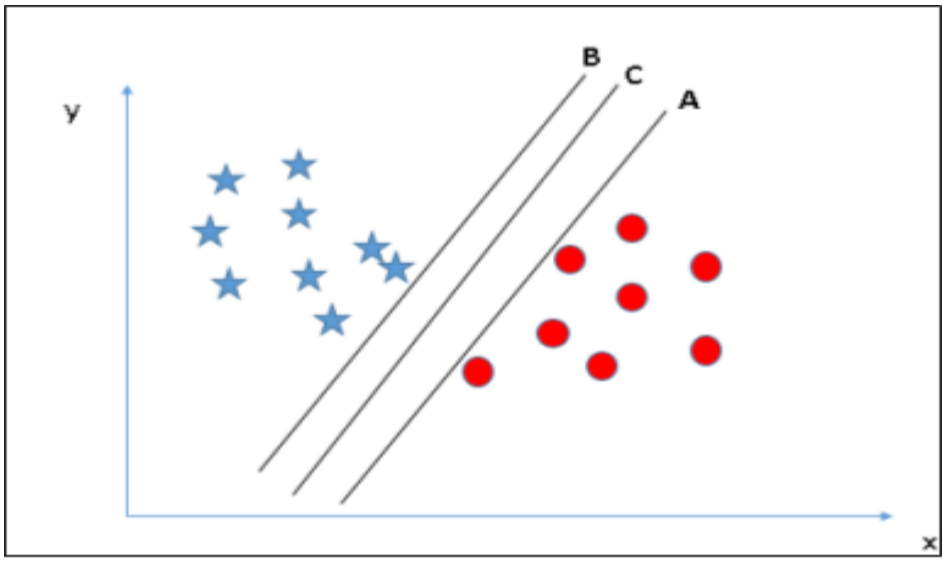

Fig 3.6 : SVM plane detection -2

Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as Margin. Let’s look at the below image:

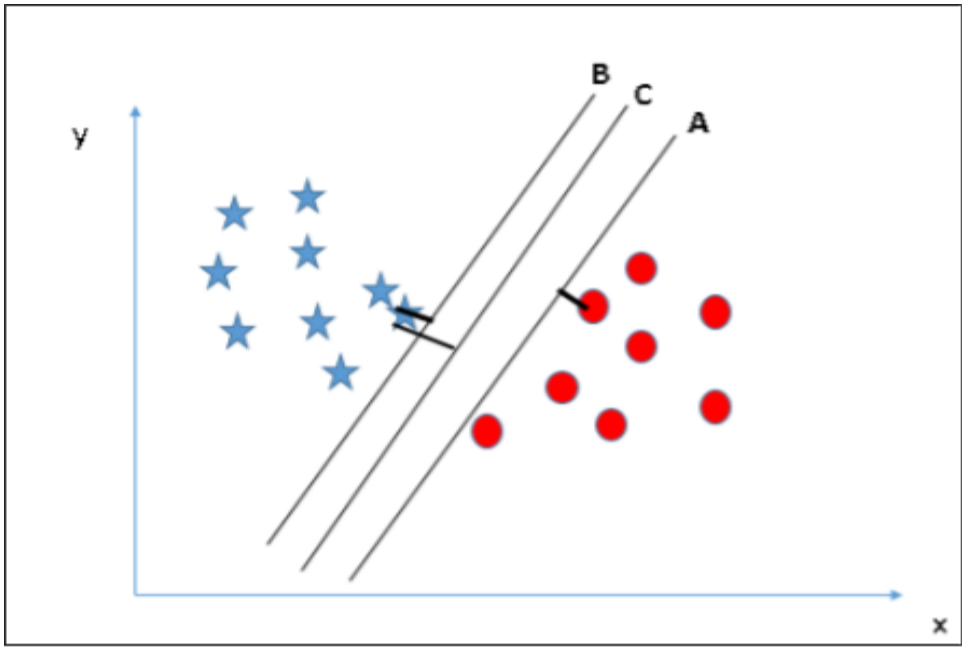

Fig 3.7 : Explanation of SVM plane detection - 2

Above, you can see that the margin for hyper-plane C is high as compared to both A and B. Hence, we name the right hyper-plane as C. Another lightning reason for selecting the hyper-plane with higher margin is robustness. If we select a hyper-plane having low margin then there is high chance of miss-classification.

- **Identify the right hyper-plane (Scenario-3):Hint:** Use the rules as discussed in previous section to identify the right hyper-plane

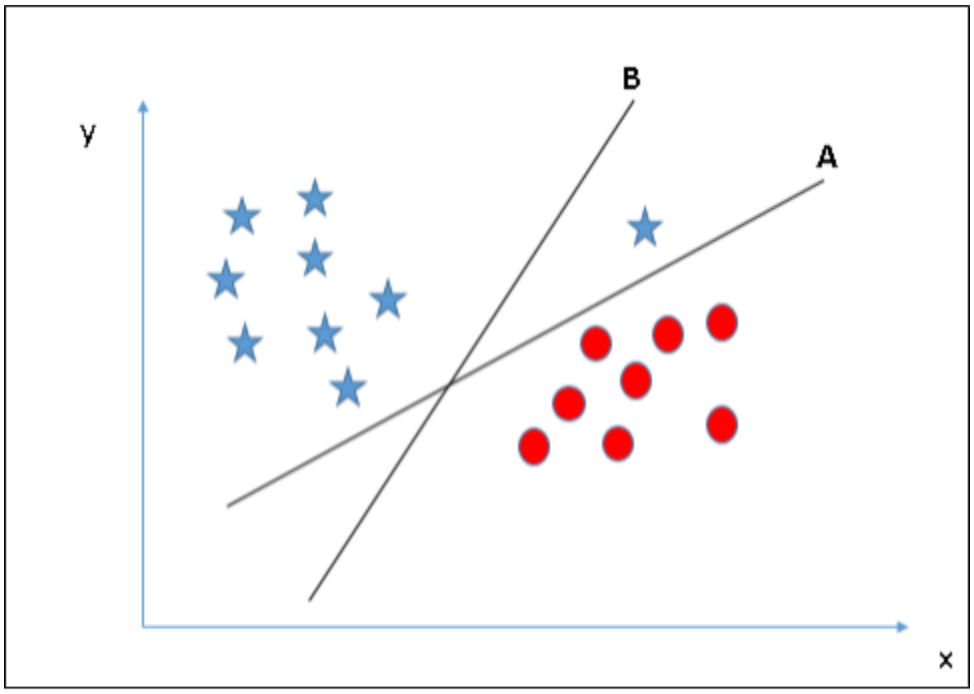

Fig 3.8 : SVM plane detection - 3

Some of you may have selected the hyper-plane B as it has higher margin compared to A. But, here is the catch, SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin. Here, hyper-plane B has a classification error and A has classified all correctly. Therefore, the right hyper-plane is A.

- **Find the hyper-plane to segregate to classes (Scenario-4):** In the scenario below, we can't have linear hyper-plane between the two classes, so how does SVM classify these

two classes? Till now, we have only looked at the linear hyper-plane.

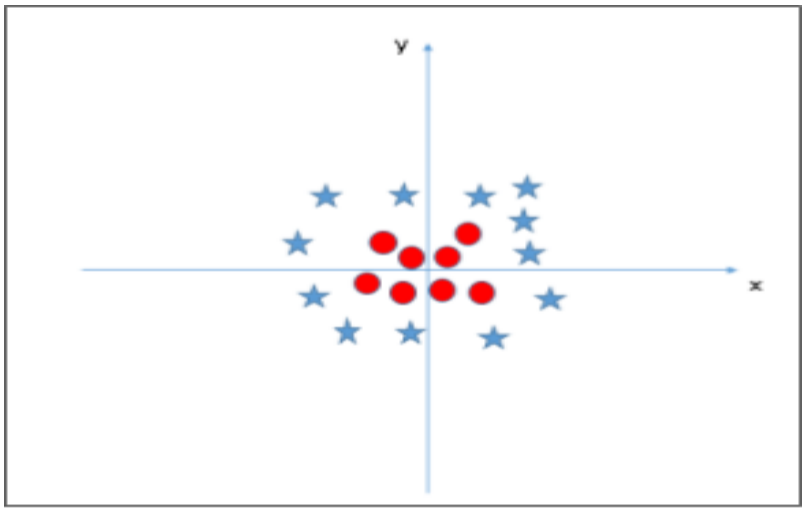

Fig 3.9 : Example of Non- linear SVM

SVM can solve this problem. Easily! It solves this problem by introducing additional feature. Here, we will add a new feature z=x^2+y^2. Now, let's plot the data points on

axis x and z:

When we look at the hyper-plane in original input space it looks like a circle:

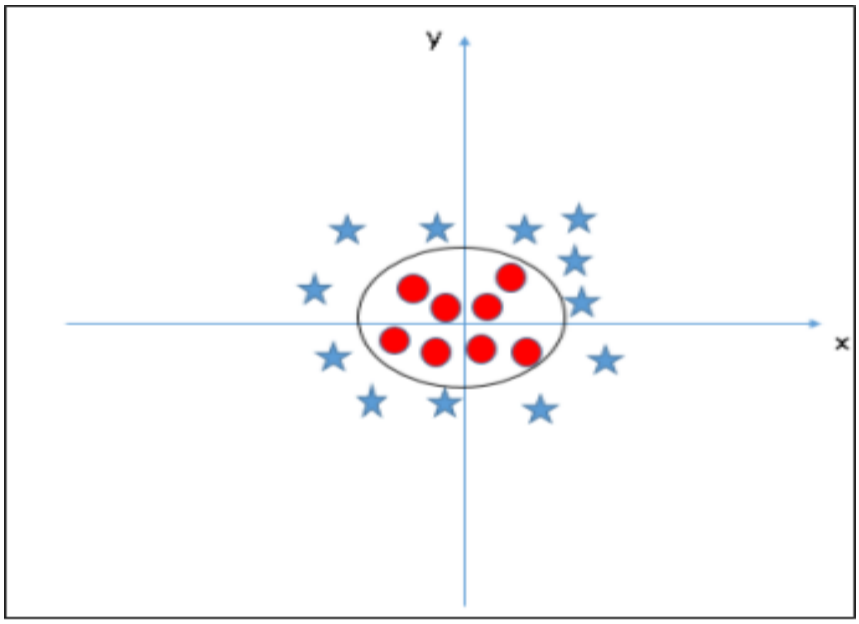

Fig 3.10 : Solution of Non-Linear SVM

Chapter 4
PROPOSED MODEL

4.1 Database formation via Web Scraping :

- Web Scraping : also termed Screen Scraping, Web Data Extraction , Web Harvesting etc. is a technique employed to extract large amounts of data from websites whereby the data is extracted and saved to a local file in your computer or to a database in table (spreadsheet), text, images , video format. Data displayed by most websites can only be viewed using a web browser. They do not offer the functionality to save a copy of this data for personal use. The only option then is to manually copy and paste the data - a very tedious job which can take many hours or sometimes days to complete. Web Scraping is the technique of automating this process, so that instead of manually copying the data from websites, the Web Scraping software will perform the same task within a fraction of the time.

```
from google_images_download import google_images_download  #
importing the library

response = google_images_download.googleimagesdownload()  #
class instantiation
arguments = {"keywords": "Tree yoga pose", "limit": 100,
"print_urls": True, "format": "png", "output_directory":
"/home/aakash/Downloads/dataset/", "size": ">640*480", "type":
"photo"}
```

- Normalization : since the images collected from the above step vary in dimensions i.e. length and width , so we need to establish uniformity among the images so that the application of any algorithm is smooth and precise . We used each person's bounding box coordinates to crop and scale each image (and corresponding keypoint coordinates) to a consistent size. We further normalized the resulting key points coordinates by treating them as a normalized vector array.

```
img = Image.open(image)
img = img.convert('RGB')
img1 = img.resize((400, 300), Image.ANTIALIAS)
```

- Using PoseNet : as stated in the previous chapter , we use PoseNet to obtain the coordinates of the human body . The TensorFlow and PoseNet libraries are imported in the web browser . We then provide jpg/png images as input to PoseNet library and receive json data as output .

  If the overall confidence of json data is less than 0.4 , then that particular entry is not processed further .

  Using Python we converted json data into data frames . Data frames were converted into csv files using python . We also added a column named "Pose" to complete the dataset , which has 35 columns . Separate csv file was created for each pose .

```
posenet.load().then(function(net){
        var poses = net.estimatePoses(imageElement, {
        flipHorizontal: flipHorizontal,
        decodingMethod: 'single-person'
      });
```

```
        var pose = poses[0];
        return poses;
        }).then(function(poses){
        pose = poses[0]
        #Json Data
        console.log(pose);
}
```

4.2 Pose prediction :

- The image whose pose is to be determined is given as input to the system , which uses PoseNet , Flask and other python libraries as mentioned above .
- Using PoseNet we obtained json data and then with the help of python converted the same into data frame .
- Using training data set , ie the csv files made using database formation , we made data frames and concatenated all the training data frames to a single data frame .
- Since we have the x-coordinate and y-coordinate of the test image , we use SVM algorithm to find the Pose column's value for the desired image .

```
classifier2 = svm.SVC()
classifier2.fit(X_train, y_train)
y_pred2 = classifier2.predict(X_test)
return y_pred2
```

4.3 Estimation of Region of Interest (RoI) :

- The estimation of RoI is made after we have predicted what the pose the user wants to perform or is currently performing .
- RoI is proposed in this project to calculate how accurately user is performing the pose.
- The similarity (or difference) between the performed pose and ideal pose is found out using the angles formed by 3 consecutive body joints .
- The ideal angles are predefined and calculated by using the median of the angles in the training data set images .
- Using the same method we find the angles of the current image and compare them with the ideal angles .

```
def dist(self, x1, y1, x2, y2):
    return math.sqrt(pow((x2 - x1), 2) + pow((y2 - y1), 2))

def angle(self, A_x, A_y, B_x, B_y, C_x, C_y):
    a = self.dist(B_x, B_y, C_x, C_y)
    b = self.dist(A_x, A_y, C_x, C_y)
    c = self.dist(B_x, B_y, A_x, A_y)
    cosa = (b * b + c * c - a * a) / (2 * b * c)
    radian = math.acos(cosa)
    return radian * 180 / 3.14
```

```
 def one(self, data):
        temp = data.apply(
            lambda x: self.angle(x.leftShoulder_x, x.leftShoulder_y,
x.leftElbow_x, x.leftElbow_y, x.leftHip_x,x.leftHip_y),axis=1)
        return np.median(temp)
```

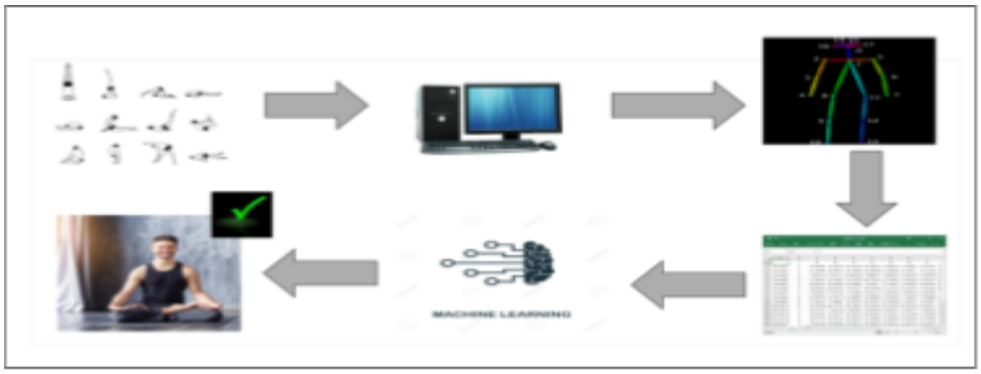

Fig 4.1 : Project work at a glance

CHAPTER 5

RESULTS

The model explained in chapter 4 was applied on the set 9 yoga postures for predicting the pose performed by the user . The efficiency for the same is displayed in the form of confusion matrix and classification report below .

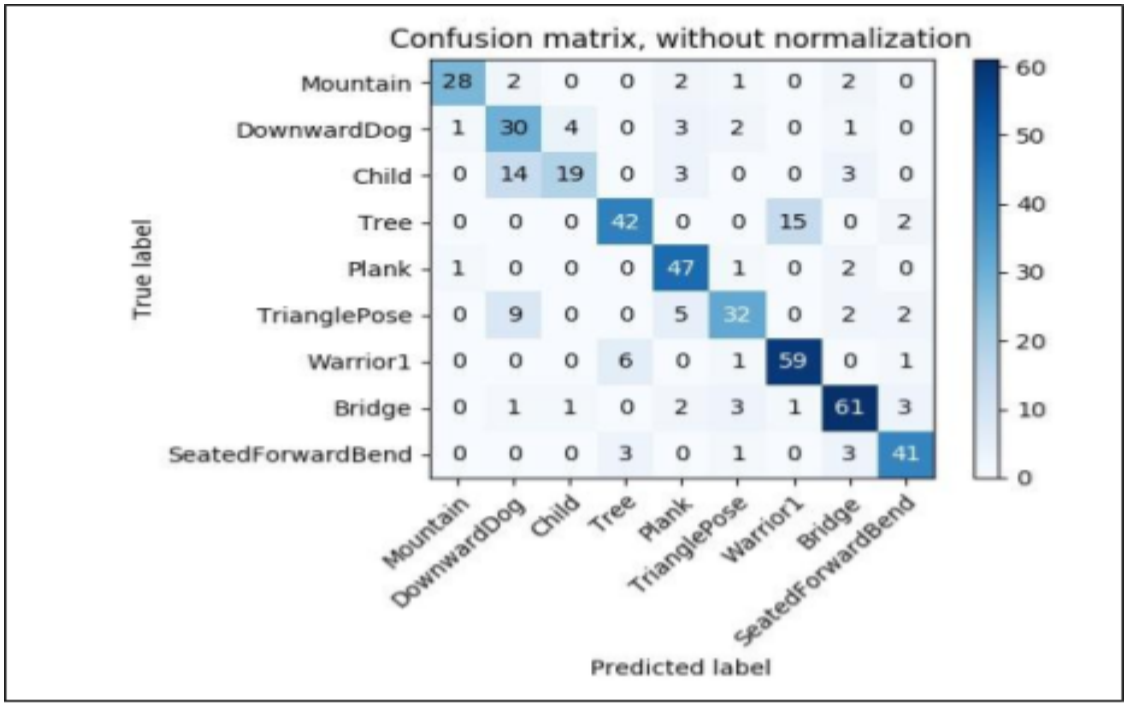

Fig 5.1 : Confusion matrix depicting the prediction

Pose	f1-score	precision	recall	support
Bridge	0.6909090909	0.7307692308	0.6551724138	29
Child	0.6788990826	0.6379310345	0.7254901961	51
DownwardDog	0.7058823529	0.8571428571	0.6	40
Mountain	0.8099173554	0.8596491228	0.765625	64
Plank	0.752293578	0.7454545455	0.7592592593	54
SeatedForwardB	0.7547169811	0.7547169811	0.7547169811	53
Tree	0.775862069	0.6923076923	0.8823529412	51
TrianglePose	0.8085106383	0.7808219178	0.8382352941	68
Warrior1	0.8080808081	0.8510638298	0.7692307692	52
micro avg	0.7619047619	0.7619047619	0.7619047619	462
macro avg	0.753896884	0.7677619124	0.7500092061	462
weighted avg	0.7617358568	0.7704404674	0.7619047619	462

Table 5.1 : Classification Report

CHAPTER 6

CONCLUSION

This work presents a novel end-to-end trainable model for addressing the problem of musculoskeletal disorders including osteoarthritis carpal tunnel syndrome, hyper kyphosis and lower back pain . Our formulation of the problem as SVM-based classification of joint coordinates and the presented RoI calculation mechanism has the advantage of capturing context and reasoning about pose in a holistic manner . As a result, we are able to achieve state-of-art or better results on several challenging academic datasets.

CHAPTER 7

FUTURE SCOPE

In future, we plan to investigate novel architectures which could be potentially better tailored towards localization problems in general, and in pose estimation in particular. The upper bound performance shows that there is room for improvement and that a considerable boost could be obtained by adequately scoring the pose proposals. Our first attempt at rescoring them has shown encouraging results in that direction. Another line of improvement concerns the training data. In this work, we proposed a solution to automatically annotate 2D images with “pseudo” ground-truth 3D poses. Our ongoing research indicates that better 3D and 2D performances could be obtained with LCRNet if more accurate real-world training data was available, e.g. through manual curation. The combined network can be trained in an end-to-end manner influencing each other

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

[2] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR, 2014.

[3] C. Szegedy, A. Toshev, and D. Erhan. Object detection via deep neural networks. In NIPS 26, 2013.

[4] Z. Cao, T. Simon, S. Wei, and Y. Sheikh, "Realtime multi-person 2D pose estimation using part affinity fields," in CVPR, 2017.

[5] J. J. Tompson, A. Jain, Y. LeCun, and C. Bregler, "Joint training of a convolutional network and a graphical model for human pose estimation," in NIPS, 2014.

[6] A. Toshev and C. Szegedy, "DeepPose: Human pose estimation via deep neural networks," in CVPR, 2014.

[7] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in CVPR, 2016.

[8] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in ECCV, 2016.

[9] X. Fan, K. Zheng, Y. Lin, and S. Wang, "Combining local appearance and holistic view: Dual-source deep neural networks for human pose estimation," in CVPR, 2015.

[10] I. Akhter and M. Black, "Pose-conditioned joint angle limits for 3D human pose reconstruction," in CVPR, 2015.

[11] C. Wang, Y. Wang, Z. Lin, A. L. Yuille, and W. Gao, "Robust estimation of 3D human poses from a single image," in CVPR, 2014.

[12] C. Chen and D. Ramanan, "3D human pose estimation = 2D pose estimation + matching," in CVPR, 2017.

[13] B. Xiaohan Nie, P. Wei, and S.-C. Zhu, "Monocular 3D human pose estimation by predicting depth on joints," in ICCV, 2017.

[14] J. Martinez, R. Hossain, J. Romero, and J. J. Little, "A simple yet effective baseline for 3D human pose estimation," in ICCV, 2017.

[15] G. Pavlakos, X. Zhou, K. G. Derpanis, and K. Daniilidis., " Coarse-tofine volumetric prediction for single-image 3D human pose," in CVPR,
2017.
[16] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele, "Deepercut: A deeper, stronger, and faster multi-person pose estimation model," in ECCV, 2016 .
[17] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation."
[18] W. Yang, S. Li, W. Ouyang, H. Li, and X. Wang, "Learning feature pyramids for human pose estimation," in ICCV, 2017
[19] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in CVPR, 2016
[20] H.-S. Fang, S. Xie, Y.-W. Tai, and C. Lu, "RMPE: Regional multi-person pose estimation," in ICCV, 2017.
[21] S. Jin, X. Ma, Z. Han, Y. Wu, W. Yang, W. Liu, C. Qian, and W. Ouyang, "Towards multi-person pose tracking: Bottom-up and topdown methods," in ICCV PoseTrack Workshop, 2017.
[22] B. Xiao, H. Wu, and Y. Wei, "Simple baselines for human pose estimation and tracking," ECCV, 2018.
[23] I. Radwan, A. Asthana, and R. Geocke, "Global pose refinement using bidirectional long-short term memory," https://posetrack.net/workshops/ iccv2017/pdfs/MPR.pdf.
[24] Y. Xiu, J. Li, H. Wang, Y. Fang, and C. Lu, "Pose flow: Efficient online pose tracking," BMVC, 2018.

[25]https://medium.com/tensorflow/real-time-human-pose-estimation-in-the-browser-with-tensorflow-js-7dd0bc881cd5
[26]https://www.learnpython.org/en/Pandas_Basics
[27]https://docs.scipy.org/doc/numpy/user/whatisnumpy.html
[28]https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
[29]https://scikit-learn.org/stable/modules/preprocessing.html
[30]https://scikit-learn.org/stable/modules/neighbors.html
[31]https://pypi.org/project/google_images_download/
[32]https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example