HUMAN POSE ESTIMATION IN 2D IMAGES BY DETECTING UPPER BODY JOINTS

A thesis

submitted in partial fulfilment of the requirement for the Degree of

Master of Computer Science and Engineering

of

Jadavpur University

by

AKASHNIL SARKAR

Registration number: 140748 of 2017-18

Roll Number: M4CSE19017

Under the guidance of

Dr. Debotosh Bhattacharjee

Department of Computer Science and Engineering

Jadavpur University, Kolkata – 700032, India,2019

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

Certificate of Recommendation

This is to certify that the dissertation entitled “Human Pose Estimation in 2D Images By Detecting Upper Body Joints” has been carried out by Akashnil Sarkar (University Registration Number: 140748 of 2017-18, Roll: M4CSE19017) under my guidance and supervision and be accepted in partial fulfilment of the requirement for the Degree of Master of Computer Science and Engineering. The research results presented in the thesis have not been included in any other paper submitted for the award of any degree in any other University or Institute.

……

Dr. Debotosh Bhattacharjee(Thesis Supervisor)

Department of Computer Science and Engineering

Jadavpur University, Kolkata – 32.

Countersigned

……..

Prof. Mahantapas Kundu

Head, Department of Computer Science and Engineering

Jadavpur University, Kolkata – 32.

……..

Prof. Chiranjib Bhattacharjee

Dean, Faculty of Engineering and Technology

Jadavpur University, Kolkata – 32.

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

Certificate of Approval*

This is to certify that the thesis titled "Human Pose Estimation in 2D Images by Detecting Upper Body Joints" is a bona-fide record of work carried out by Akashnil Sarkar in partial fulfilment of the requirements for the award of the degree of Master of Computer Science and Engineering in the Department of Computer Science and Engineering, Jadavpur University during the period of September 2017 to June 2019. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein but approve the thesis only for the purpose for which it has been submitted.

……

Signature of Examiner 1

Date:

……

Signature of Examiner 2

Date:

*Only in case the thesis is approved

FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY

Declaration of Originality and Compliance of Academic Ethics

I hereby declare that the thesis entitled "Human Pose Estimation in 2D Images By Detecting Upper Body Joints" contains literature survey and original research work by the undersigned candidate, as part of his Degree of Master of Computer Science & Engineering.

All information has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials that are not original to this work.

Name: Akashnil Sarkar

Registration Number: 140748 of 2017-18

Examination roll number: M4CSE19017

Thesis title: Human Pose Estimation in 2D Images By Detecting Upper Body Joints

……

Signature with date

Acknowledgement

I would like to start off by thanking the holy trinity for guiding me into deploying the right resources and shape me into a better human being. Besides, I would like to express my heartfelt gratitude to **Dr.Debotosh Bhattacharjee**, my research guide, who took time out of his busy schedule and guided my in every possible way to make my thesis comprehensive within a short campus. Our discussions and his constructive comments and criticisms has improved my work.

It goes without saying that the amazing academic ambience that prevails in Jadavpur University which provides ample technical and academic resources enabling students to pursue their projects / thesis smoothly.

Moreover, I am immensely indebted to my friends **Mr**. **Ranit Dey**, **Mr. Gourab Ghosh**, **Miss Asmita Nandy**, **Miss Charu Arora** who have constantly motivated me and kept me going.

Most importantly, none of all this would have been possible to accomplish for me without the love and support of my family, especially to mother whose tolerance and whole-heated support helped this endeavour succeed.

This thesis would not have been completed without the inspiration and support of a number of individuals --- my thanks and appreciation to all of them for being a part of this journey and making the thesis take shape.

……..

Akashnil Sarkar

Registration Number: 140748 of 2017-18

Examination Roll Number: M4CSE19017

Department of Computer Science and Engineering

Jadavpur University

Contents

Chapter 1: Introduction

In this work, we have been given a dataset containing various 2D images obtained by randomly framing some gesture videos available in youtube.com. Also, we have been provided with the x and y positions of seven upper joints (person centric) in the images. Our task is to design an algorithm to train those images, so that the algorithm can successfully detect those upper body joints in unknown images.

Joint detection in 2D images, 3D images or gesture videos have wide range of applications nowadays in several spheres such as analysis of human motion, abnormal movement of muscles, action recognition, theft detection by surveillance cameras etc.

1.1 Joint: Definition and Relevance to pose

Dr.William C. Sheil defined a joint as a *area in the body where many (generally two) bones meet for the purpose of permitting body parts to move.* The most commonly used joints used for the purpose of motion and other activities are shoulders, elbow, wrist, ankle, foot etc. However, for other kinds of

motion, there are some special joints in human body such as the ball and socket joint, pivot joint, gliding joint, hinge joint, candyloid joint which helps in all kinds of angular and pivotal movements.

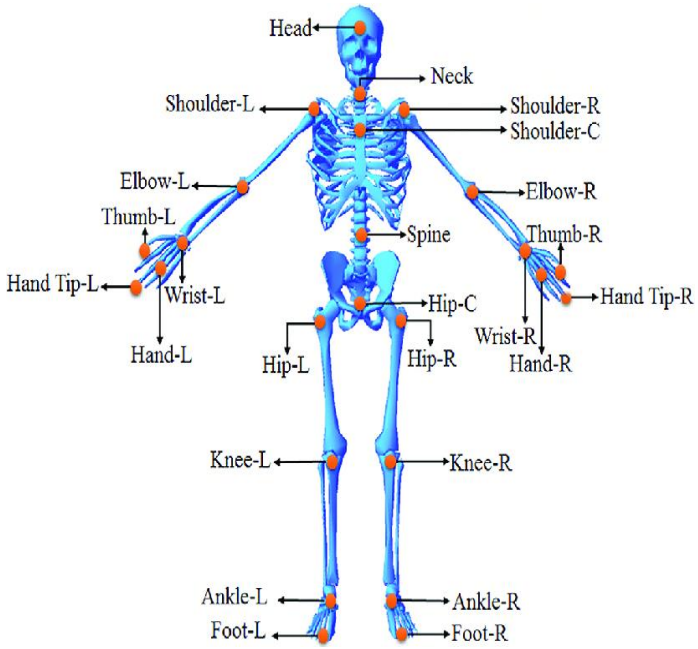


Figure 1-1 Some commonly used and visible joints

In the domain of computer science, specifically computer vision and machine learning, a typical topic of ongoing research is to identify and detect these body joints and determine the **position** and **orientation** of each of those with respect to a chosen co-ordinate system(2-dimentional or 3-dimentional) either from 2D images or videos. The combination of this position and orientation is sometimes referred to as the **pose** of the individual.

Various algorithms and methodologies have been proposed and studied which recovers the pose of an articulated body, which consists of joints and rigid body parts using image-based observations. It is one of the longest running problems in the field of computer vision and applied mathematics because of the complexity of the models that relate observation with pose, and because of the variety of situations it would be useful.

Scientists Yap Wooi Hen and Raveendran Paramesran [1] has mentioned a brief description of various applications of understanding human pose. Some of the applications are given below :

1) In various computer games, the position of various joints together helps in creating realistic animation.

2) In sports, accurate reconstruction of human pose assists athletes to visually analyse their movements and improve their performances.

3) In the fields of physiotherapy, gait analysis uses human pose to the underlying causes of a patient's movement anomalies which may be triggered by stroke, cerebral palsy, or other neuro-muscular malfunctions.

4) Another emerging application is in the sphere of video surveillance. With lower costs of cameras and

advancements in computing power, accurate analysis of human pose (individual person-centric) from videos can help surveillance operators to identify actions such as running, walking, shop-lifting, wall-climbing and various other suspicious activities.

In the last decade, deep learning has been successfully applied in this domain of research especially convolutional neural nets (CNN). It has been observed that CNNs have high success rate in tasks like image classification, object detection and classification, face recognition and many more.

1.2 Motivation

Although considerable research work has been conducted in this domain, but the work of analysing human pose is inherently faced with quite a few challenges, most important being unavailability of sufficient data. Even if images or videos are available online, it takes a lot of manual pre-processing to organise the training data (ground truth) and compare them with the test results. Also, in most of the works done, each image is considered as a training sample and several such images (in case of videos, the entire video is framed into fixed time intervals) make the training dataset, which may lead to overfitting as deep learning is able to capture inter-spatial distance between joints leading to overfitting.

Hence, in our work we have tried to develop a methodology where instead of entire image, we have treated small fragments of each image separately to determine whether it represents a joint(head, wrist, elbow, shoulder) or background, irrespective of its neighbouring fragments.

1.3 Scope of Current Work

In this work, I have developed a convolutional neural net to train small equal sized patches of 2D images with 2 possible labels for each. Either it is a joint(since, a single 2D co-ordinate is difficult to train, I have prepared the ground truth images by making each point as the centre and taking a square patch surrounding it and labelling the entire patch as the joint) or not. After repeated training, the model is prepared. Now given an unknown image, the model can identify the joints in the entire image (if any).

However, this work has a limitation. Since the method mainly detects upper body joints , estimating the exact pose (forward or backward) cannot be accomplished. For that purpose, my model may be coupled with a face detector to identify if the person is looking at the camera or away from it.

1.4 Organisation of the Thesis

In *Chapter 1*, we have discussed briefly about the definition of body joints and its applications in computer science. Also, this chapter contains our motivation and scope of the thesis.

In *Chapter 2*, we have given a brief survey of various works done in this domain of joint detection and body language understanding.

In *Chapter 3*, We describe the hardware and software resources that has been used throughout the course of this work.

Chapter 4 mainly deals with the pre-processing of the images to make them ready for training.

In *Chapter 5,* we have given the training results and how accurately the model is performing on the unknown data samples.

In *Chapter 6*, we give an overall discussion of our work related to its advantages, limitations and possible scope for improvement and extension.

Chapter 2: Literature Survey

This survey mainly focuses on some of the notable experimental works that has been performed by data scientists to detect joints and estimate human pose.

Jammie Shotton et al[2] proposed a naïve approach of quick and accurate prediction of joint positions from single depth image without capturing temporal information. The approach utilises the concept of object recognition designing an intermediate body part representation problem that maps the pose estimation problem into a per-pixel classification problem. It consists of the following parts:

1.Depth imaging: This technology mainly used to represent the calibrated depth in the image rather than pixel intensity or colour. It helps in capturing human motion, and handling different body shapes and occlusions. For the training data, realistic synthetic depth images of humans of variable shapes and sizes in highly varied poses are sampled from a large motion capture database.

2.Motion capture data: In order to account for large variations in human poses, instead of normal RGB images, the dataset used is a large database of motion capture (MOCAP) consisting of approximately 500k frames in a few hundred sequences of

driving, dancing kicking etc. A semi-local body part classifier is implemented to generate unseen poses. In particular, location of all the limbs need not be recorded, a wide range of poses prove sufficient for training and evaluation.

3.Synthetic data generation: A randomized pipeline is built from which to sample fully labelled training images. The synthesis pipeline first randomly samples a set of parameters from the depth images obtained from motion capture data, which is targeted to various 3D meshes spanning the range of body shapes and sizes. Further, slight random variation in in height, camera angles, clothing, hairstyles give extra coverage of boy shapes.

4.Body part labelling: This is the key task in this algorithm, which is done for intermediate body part representation. Several localized body parts that densely cover the body. These parts are specified in a texture map that is retargeted to skin various characters during rendering. The pairs of depth and body part images are used as fully labelled data for learning the classifier. The commonly used body parts are head, arms, wrists, elbows, ankles and feet. The co-ordinates of each are estimated from the depth images. During the evaluation phase, for each of the corresponding body parts in the test image, their locations are estimated and matched with the training images. The maximal matching is used for estimation of poses.

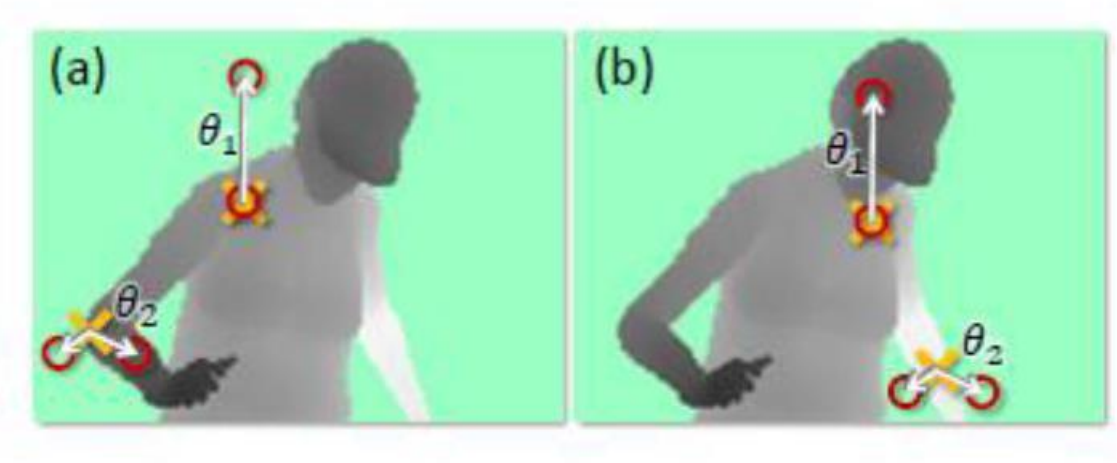


Figure 2-1 Depth Image features. The yellow crosses indicate that the pixel x is classified. In (a), the two example features give a large depth difference response. In (b), the same features at new images give much smaller response

Greg Mori and Jeetendra Malik[3] proposed a graphical approach in which the positions of joints (head, elbows, wrists etc.) are detected and those results are used to estimate the pose in 3D space. The idea is to capture the image of the same human body in different positions and configurations with respect to the camera. In each of these images, the locations of the joints are manually marked and stored for future reference. The unknown (test image) is matched with each of these stored images using shape context matching.

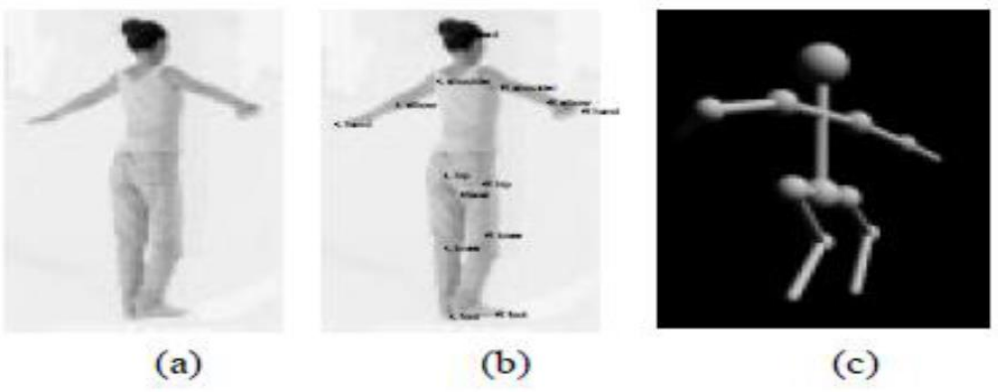


Figure 2-2 a) Input image, b) Manually extracted key points, C)3D projection of 2D configuration

Taking each of the stored images and the test image and a set of pre-decided joints, an optimal match between the corresponding joints in both the images is estimated. For this purpose, they have used a bipartite graph, in which the nodes in each side represents the corresponding joints in each image. Now each edge has a weight that represents the matching cost of the joints represented by its end points. Similar points have a low matching cost and dissimilar points have a higher matching cost. Now after comparing with all the stored images, the pose of the test image is estimated as the one which has the least matching cost.

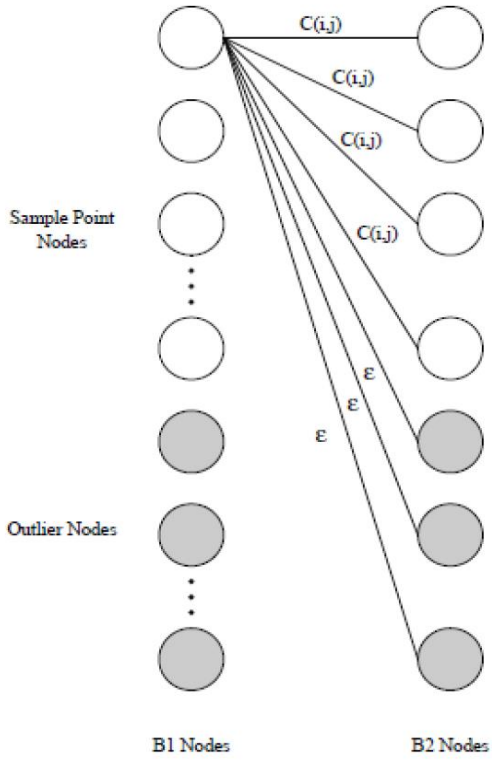


Figure 2-3 The bipartite graph used to match sample points of two bodies. Only the edges from the first node are shown for clarity. Each node from B1 is connected to every node from B2. In addition, _-cost outlier nodes are added to either side. These outlier nodes allow us to deal with missing sample points between figures (arising from occlusion and noise).

Over the last few years, with the rapid advancement of machine learning (especially deep learning), the task of joint localisation has become comparatively less cumbersome since, manual feature selection and hand annotation of data are no longer required. In this sub-section, we discuss some of the notable works done in this regard.

Thomas Pfister et al.[4] proposed a method of tracking 2D human upper body joints in long gesture videos, containing high variations in pose and background. In this approach, the task of pose estimation is treated as a regression problem where they have used a convolutional neural network consisting of several layers of convolutions and non-linearities. The input to the network is a set of RGB video frames and the outputs are the co-ordinates of upper body joints. The architecture is based on the work of Sermanet et al.[5] which has achieved excellent results on ImageNet Challenge 2013 of object classification and localisation tasks. The only difference is that instead of a single image, multiple frames are given as input at a time to capture video specific information. The training data is also augmented with random crops and flips.

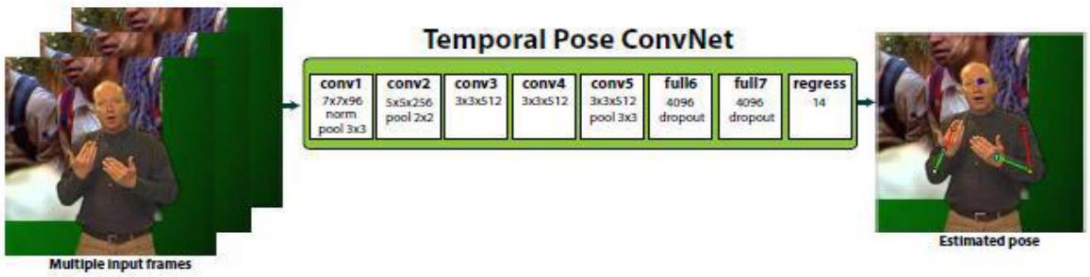


Figure 2-4 Given a set of input frames, the network regresses the position of head, shoulders, elbows and wrists

Alexander Toshev and Christian Szegedy[5] formulated a cascaded deep learning architecture. In their work, every image is labelled with a pose vector containing the x and y co-ordinates of upper body joints. Thus, every image is denoted by (x,y) where x is the image data and y is the ground truth pose vector. The joint co-ordinates are then normalised with respect to a bounding box that bounds the human body. Position of the joints are thus scaled by the centre of the box and the box size. The task is similar to the previous model, where the output is expected to be the normalised pose vector for the unknown image. The architecture is based on that used by Krizhevsky et al.[6] for image classification. The uniqueness of their approach lies in cascading of regressor. The results obtained in each stage is fed into the next stage to estimate the displacement of each joint towards the actual (ground truth) location. In each subsequent stage, the images are cropped around the predicted joints and the same regressor is applied on the sub-image. Thus, the images become more and more high resolution in each stage ultimately leading to higher precision.

Chapter 3: Technical Resources

3.1 Hardware Used

1. Laptop
2. 4 GB RAM and 1 TB HDD
3. Intel i5 CPU
4. Windows 10 64 bit

3.2 Software Used

1. Matlab 2018a

2. Anaconda3 (64 bit) operating on Python 3.7

    - Modules:

        a) Matplotlib 3.0.3
        b) OpenCV 3.4.3
        c) Scikit-learn 0.20.3
        d) Pandas 0.24.2
        e) Keras (with Tensorflow backend) 2.2.4
        f) Math
        g) OS

h) Skimage 0.14.1

3.3 Online Platform

1. Google Colaboratory with K80 gpu RAM (for training purpose)

3.4 Dataset

I have used YouTube pose dataset[7] for my thesis. The dataset is a collection of 50 YouTube videos covering a broad range of activities like dance, stand-up comedy, sports, disk jockeys and many more. Each such video is divided into 100 frames, thus giving a total of 5000 samples. However, the division of training data and test data is not explicitly given.

Apart from the images organised in separate folders, a matlab file named “Youtube_Pose_dataset.mat” is given which contains a structure array named “data”. There are 50 elements in this array each corresponding to one video. Each element is structured as follows :

a) data(i).url - string containing the youtube weblink for video i.

b) data(i).videoname – string containing the code name of the youtube video.

c) data(i).locs – 2X7X100 array containing 2D locations for the ground truth upper body joints. The first row denotes the x-values and the second row denotes the y-values. The columns are formatted from left to right as: Head, Right wrist, left wrist, right elbow, left elbow, right shoulder and left shoulder.

d) data(i).frameids – 1X100 array containing the frame indices which were annotated.

e) data(i).label_names – cell array of strings corresponding body joint labels. They are {‘Head’},{‘Right wrist’},{‘Left wrist’},{‘Right elbow’},{‘Left elbow’},{‘Right shoulder’} and {‘Left shoulder’}.

f) data(i).crop – 1X4 array giving the crop bounding box [topx, topy, botx, boty] from the original video.

g) data(i).scale – value the video should be scaled by.

h) data(i).imgPath – cell array containing paths to the pre scaled and cropped annotated frames.

   data(j).imgPath{k} refers to video j and frame k with frameid data(i).frameids(k) and joint locations data(i).locs(:,:,k).

i) data(i).origRes – 1X2 array [height and width] resolution of the original video.

j) data(i).YouTubeSubset – Boolean, true if the video belongs to the YouTube Subset dataset.

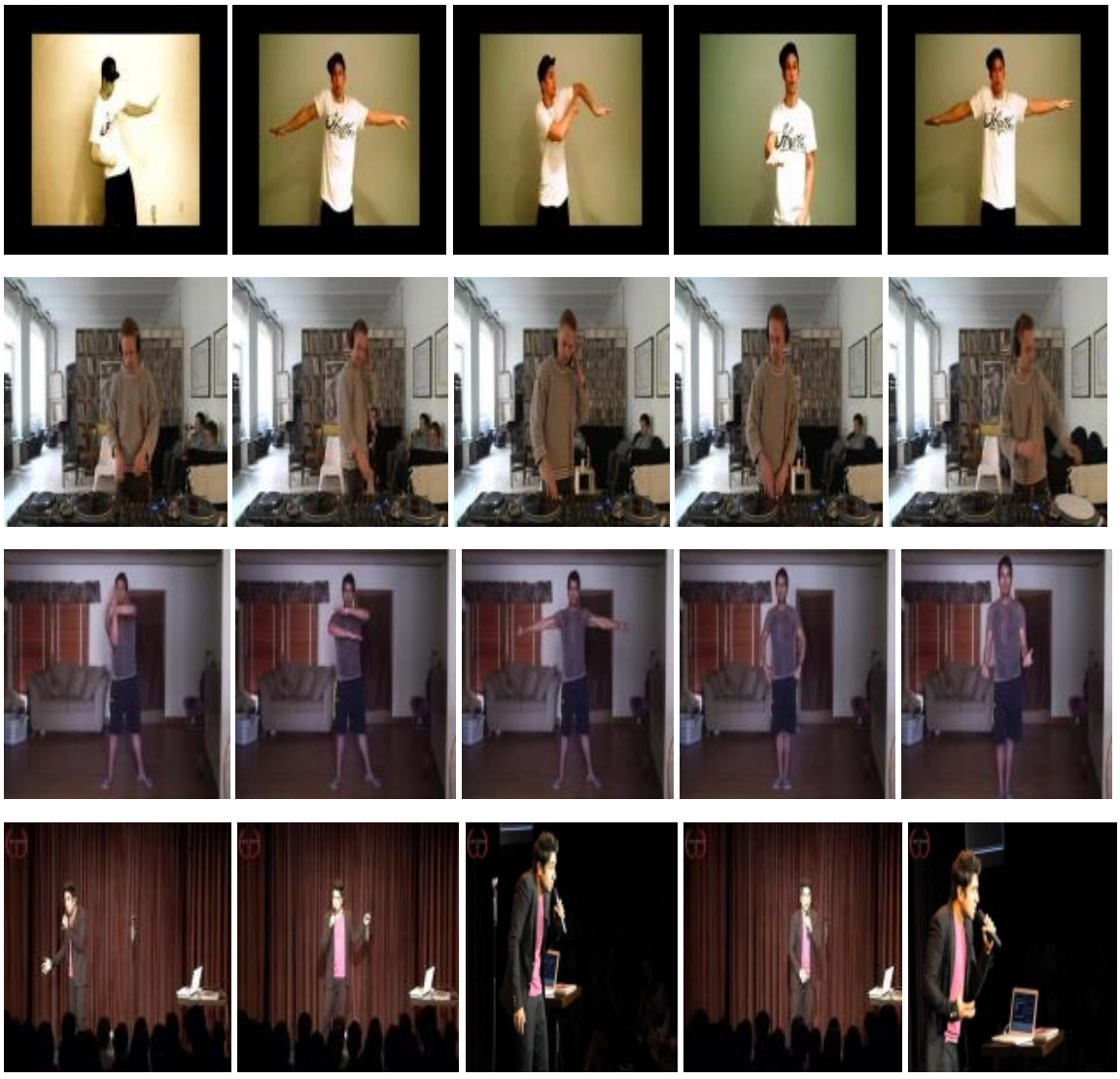


Figure 3-1 Some sample images from the dataset

Chapter 4: Image Pre-processing

Different images given in the dataset has different size. Hence, it is necessary to convert them to a specific size as convolutional neural network can take a fixed sized input. Also, the ground truth co-ordinates, are given in terms of that size. So, we had to re-compute them with respect to the resized image. Let us illustrate this with the help of an example.

Figure 3-2 Training image of original size

The size of the image shown above is 236X419X3(for R, G, B channels). The positions of the joints corresponding to this image is given below in tabular form:

Joint	x-coordinate	y-coordinate
Head	203.1011	76.4029
Right wrist	156.9210	55.8471
Left wrist	276.5951	231.5568
Right elbow	142.8417	110.1932
Left elbow	274.3424	188.1925
Right Shoulder	185.3612	120.6119
Left Shoulder	253.7866	128.7779

For our convenience, we have chosen a size of 256X256X3 as the chosen image size for our experiment. Hence, in order to locate the joint positions in the changed image, we have used a scaling process. Let us see how it works.

If the position of a point in image of size aXb, is given as (p,q), then the position of the same point in the resized image (a”Xb”) is given as follows:

1. p” = (p X b”)/b.
2. q” = (q X a”)/a.

Putting the values, and rounding-off to the nearest digit, we get the new set of joint positions as follows:

Joint	x-coordinate	y-coordinate
Head	124	83
Right wrist	92	61
Left wrist	169	251
Right elbow	88	120
Left elbow	168	204
Right Shoulder	114	131
Left Shoulder	155	136

After rescaling all the images, the next job is to mark the joints. For every image, we take each joint position as mid-point and then we draw a white square patch on it, and make the rest of the image as black. Therefore, the black portion represents the background and the white portion represents the joints. Now, for the image given above, after resizing and scaling, we get its corresponding ground truth image as follows:-

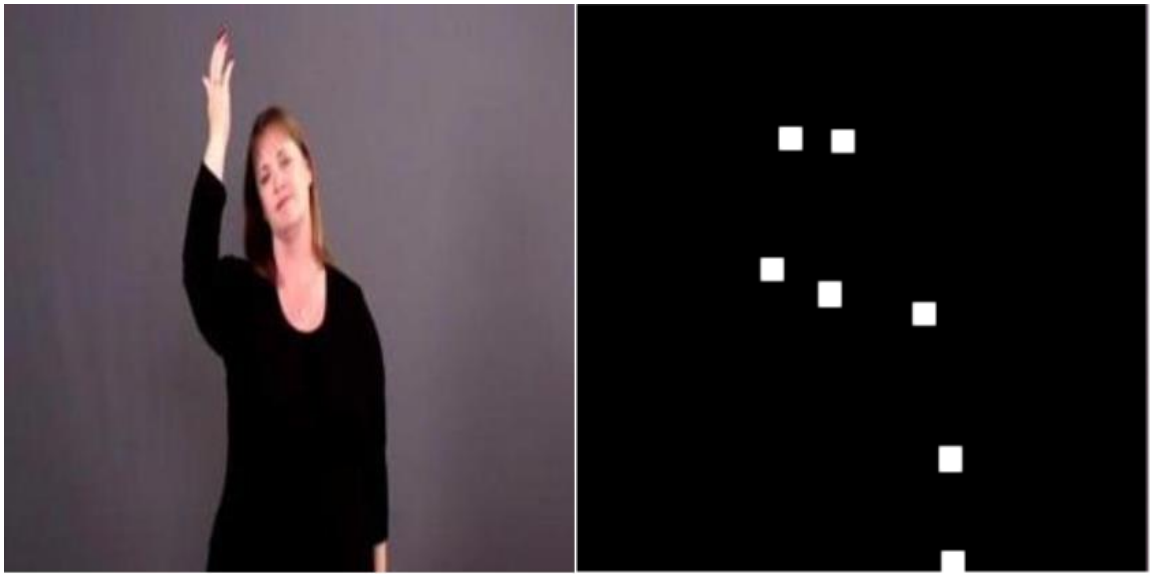


Figure 3-3 Training image resized and rescaled (left) and the ground truth (right)

Now, let us check if the patches are correctly marked. For this, we superimpose both the images.

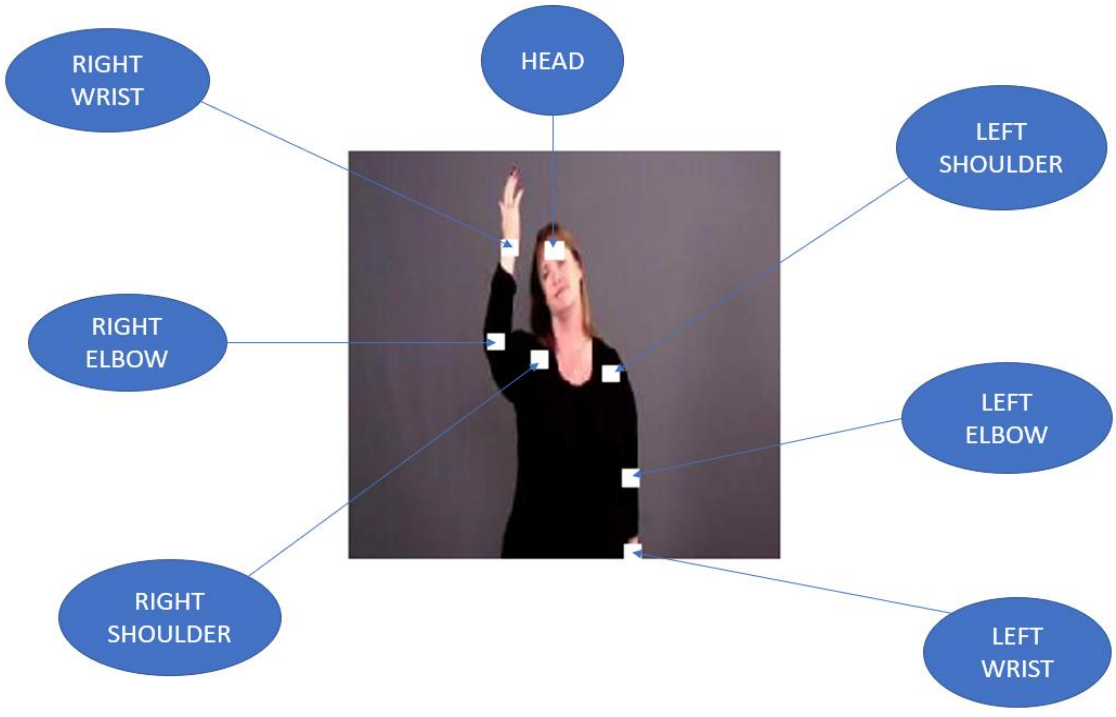


Figure 3-4 Superimposed training and ground truth images

In this process, we have manually prepared the ground truths of all the images in the dataset for training.

Chapter 5: Working Methodology: Training and Results

We have used a patch-based classification for our work, instead of directly regressing the joint locations from the whole image. The main idea behind this approach is the work done by patch-based convolutional neural network for whole slide tissue image classification by Le Hou et al[7].The purpose is to prevent the network from capturing too much of background information which might lead to overfitting. We treat each **joint** as an object, and the remaining portion as **background**.

Based on this, we segment each image into equal sized patches. We have used 16X16 patches for each image and its ground truth. Now, we compare both the images patch by patch and check if the patch is black (background) or white (joint) which can be done by comparing the pixel values. Now, for each patch, if it represents a background, then we assign it a label of 0, otherwise 1.

Therefore, the total number of patches extracted from each image is (256 X 256)/(16 X 16) = 256. We store these 256 labels in an array.

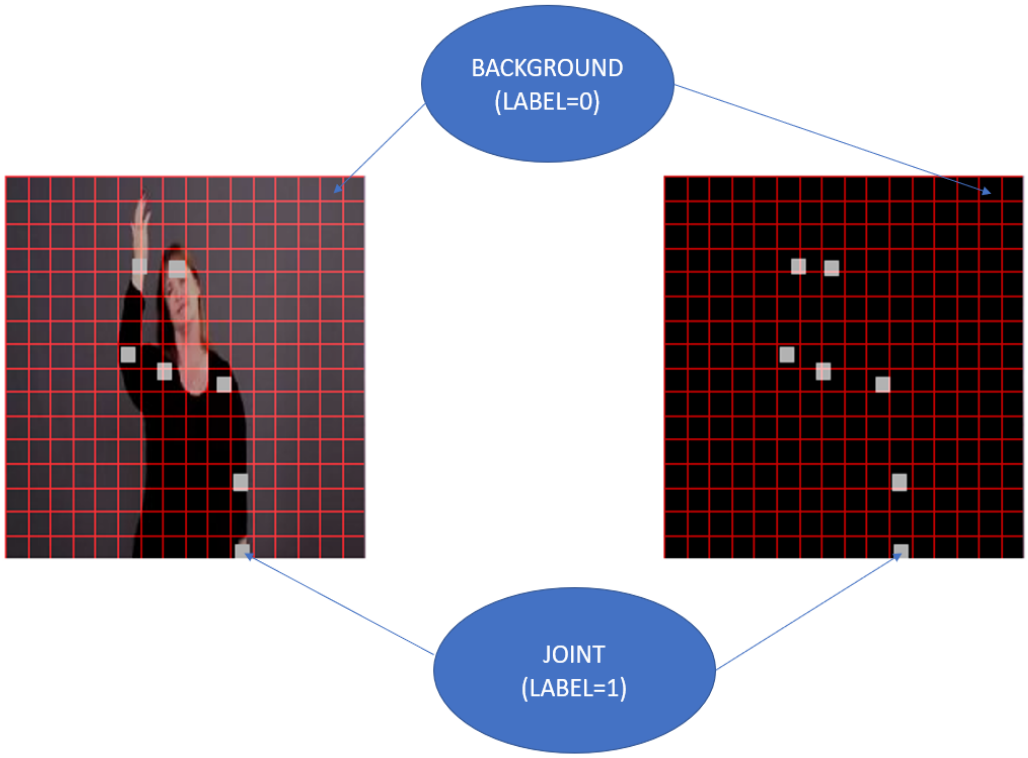


Fig 5-1 Marking of the joints and background patches from the image and its corresponding ground truth image

However, one important point is to be noted here. Apart from, background and joint patches, there are some mixed patches containing part of background and joint both. Labelling such patches becomes difficult. Hence, before labelling, we compute for each patch in the ground truth image, we check if more than 50% of the pixel values are 255 (white) including the mid-point, then we make it a joint patch, otherwise it is treated as background.

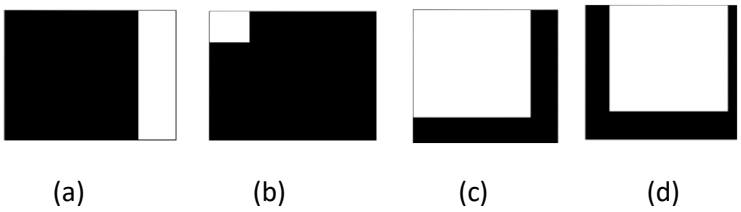


Fig 5-2 a) background, b) background, c) joint, d) joint

For the image given in Fig 4-1, we can represent the class labels of all the 256 patches in array as follows:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

The dataset consists of 5000 images among which we have randomly selected 4500 images for training and 500 for testing. Hence, after manually annotating all the training images, we get a total of 115,2000 patches and labels. For better convergence, we had to normalise the images by subtracting the mean and dividing by standard deviation. After

that, the data is ready to be fed into our deep learning network.

Due to rapid success of deep learning (convolutional neural network) in computer vision and pattern recognition tasks, we have used CNN classification network. The input size is 16X16 and the output is a class-label (0 - background and 1 - joint). The following table shows the network we have used:

Layer		Feature map	Size	Kernel size	Stride	Activation
Input	Image	1	16X16	-	-	-
1	Convolution2D	6	12X12	5X5	1	tanh
2	Average pooling2D	6	6X6	2X2	2	tanh
3	Convolution2D	16	6X6	5X5	1	tanh
4	Average pooling2D	16	3X3	2X2	2	tanh
5	Convolution2D	120	3X3	5X5	1	tanh
6	FC	-	84	-	-	tanh
7	FC	-	100	-	-	tanh
8	FC	-	50	-	-	tanh
9	FC	-	10	-	-	tanh
Output	FC	-	2	-	-	softmax

We have trained our network on the training patches in Google colab provided gpu, for 300 epochs, with a validation split of 0.25, which took about 12 hours to complete. The choice of the metric, i.e. the loss function is very important

cornerstone for deep learning to perform adequately. Since, ours is a binary classification problem, we have used **binary cross entropy** as our loss function.

$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$$

M = Number of classes (2 in our case).

$Y_{o,c}$ = 1 or 0 (correct class label for the observed image – whether image "o" belongs to class "c").

$P_{o,c}$ = Probability score for image "o" corresponding to class "c".

After training, now comes the testing on unknown images. It is clear from above, that the problem of patch classification is a skewed class problem where the number of negative samples (background) is very high compared to positive ones (joints). Hence, for these kind of problems, the most suitable accuracy metric is the analysis of **confusion matrix.**

A confusion matrix is a tabular representation of 4 kinds of samples in the test data as follows:

There are 4 basic terms which are needed to be computed to construct confusion matrix.

a) True Positives(TP): The number of samples which are actually positive(label – 1) and also predicted as positive.

b) True Negatives(TN): The number of samples which are actually negative(label – 0) and also predicted as negative.

c) False Positives(FP): The number of samples which are actually negative but classified falsely as positive.

d) False Negatives(FN): The number of samples which are actually positive but classified as negative.

N.B. The actual class labels of the test data must be known beforehand to construct confusion matrix.

	Predicted Negative	Predicted Positive
Actual Negative	TN	FP
Actual Positive	FN	TP

Fig 5-3 Confusion Matrix

Now, the total number of samples is given by (Predicted Negative + Predicted Positive) or (True Positive + True Negative).

Now, from the above matrix, we can compute various rates which gives us an idea of how well, our classifier works. The most commonly used rates are **precision**, **recall** and **f1-score**.

Precision gives the *fraction of the total number of positive samples which are correct*.

Therefore, **precision = TP / (TP +FP)**.

ON the other hand, *recall gives the fraction of the actual positive examples are classified correctly*.

Therefore, **recall = TP / (TP + FN)**.

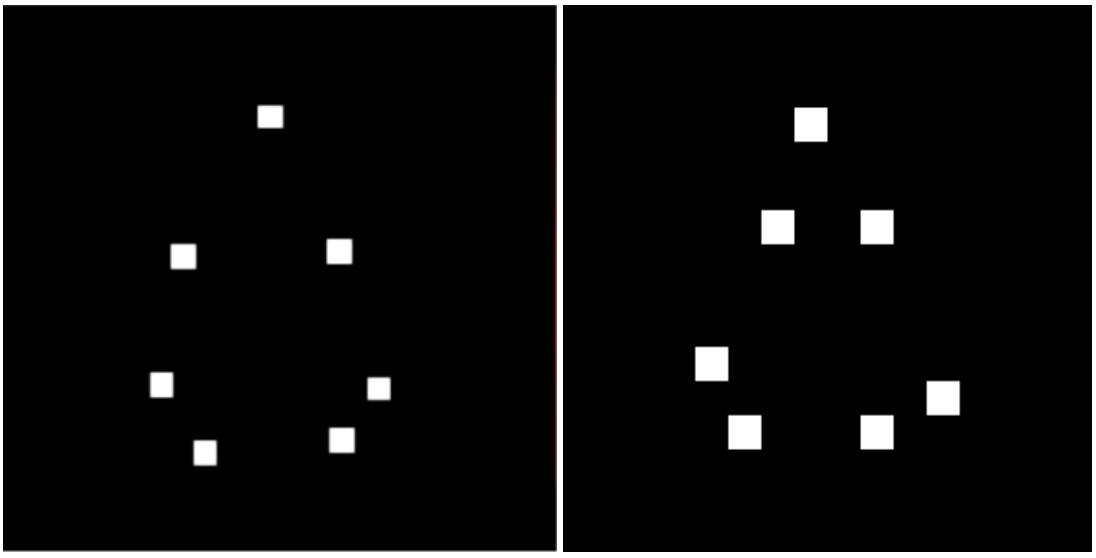
Now, while evaluating the classifier, we have to consider both the precision and the recall. However, it can be shown that both precision are recall are often antagonistic in action. That is to say, when we are attempting to increase one, the other decreases. But, a good classifier is expected to give a sufficiently high value of both precision and recall. Hence, we combine both of them which results in another matric called

the **F1-score**. F1-score is the harmonic mean of precision and recall.

Therefore, **F1-score** = **(2 * Precision * Recall)/(Precision + Recall)**.

Now, while testing the test images, we once again divide each image into 16X16 patches and give all the patches of the image into our pre trained model. The output is an array of total 256 entries containing 0 or 1 representing a background or joint respectively. Moreover, while pre-processing, we had already computed the binary ground truth images of all the images. Hence, from the ground truth, we again extract the labels for each patch and store them in another array. Thus, we get a predicted array and the ground truth array. Let the predicted array be named Y_pred and the ground truth array be named Y. Now, we compare both the arrays and compute the confusion matrix. Let us demonstrate with the help of an example.

Fig 5-4 Sample image of the test set



(a) (b)

Fig 5.5 - a) Ground truth label and b) predicted result of the figure given above

Now, evaluating the arrays Y and Y_pred, we get the results as follows :

Y = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,1,0,0,0,0,0, 0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,1,0,0,0,1,0,0,0,0,0,0,0
, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,1,0,0,0,0, 0,1,0,0,0,0,0,0,0
0,0,0,0,0,0,0,1,0,0,0,0,0, 0,0,0,0,0,0,1,0,0,0,0, 0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0, 0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0].

Y_pred = [0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 1,0,0,0
, 0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0, 0,0,0,0,0,0,0,1,0,0,1,0,0, 0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0, 0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0
, 0,1,0,0,0,0,0,0,0,0,0,1, 0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0].

From the 2 arrays, we obtain the confusion matrix as follows:

N = 256	Predicted Negative	Predicted Positive	
Actual Negative	247(TN)	1(FP)	248
Actual Positive	2(FN)	6(TP)	8
	249	7	

Therefore, we can compute precision, recall and f1-scores from the matrix. Thus, we get,

Precision = TP / (TP + FP) = 6 / (6 + 1) = **0.8571**

Recall = TP / (TP + FN) = 6 / (6+2) = **0.75**

F1-score = (2 * Precision * Recall) / (Precision + Recall)

= (2 * 0.8571 * 0.75) / (0.8571 + 0.75) = **0.7999**

In order to check whether the predicted joint patches actually overlap with the actual location of the joints, we superimpose the test image and the predicted output.

Fig 5.6 – Test image and predicted image superimposed to check if the joints are covered.

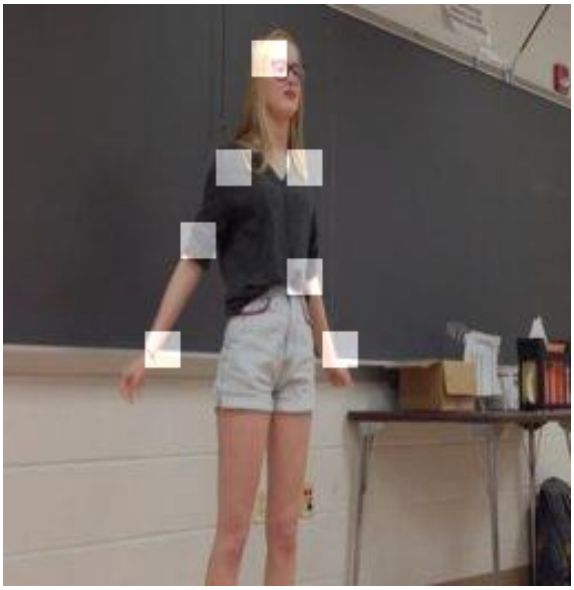
Similarly, for some of the other images, we compute the precision, recall and f1-scores to check how accurately, our model performs.

N = 256	Predicted Negative	Predicted Positive	
Actual Negative	246(TN)	2(FP)	248
Actual Positive	3(FN)	5(TP)	8
	249	7	

Precision = 0.7142

Recall = 0.625

F1-Score = 0.666

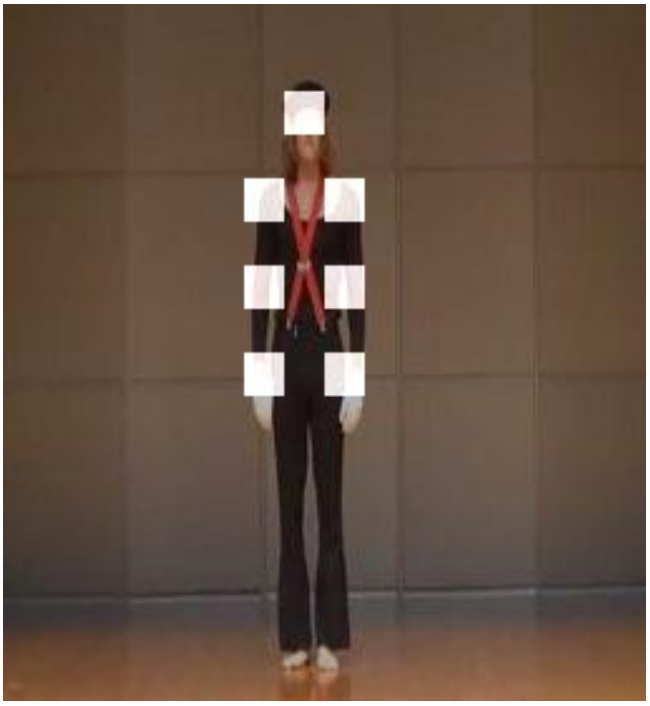


N = 256	Predicted Negative	Predicted Positive	
Actual Negative	248(TN)	1(FP)	249
Actual Positive	1(FN)	6(TP)	7
	249	7	

Precision = 0.8571

Recall = 0.8571

F1-Score = 0.8571



N = 256	Predicted Negative	Predicted Positive	
Actual Negative	246(TN)	1(FP)	249
Actual Positive	3(FN)	6(TP)	7
	249	7	

Precision = 0.8571

Recall = 0.666

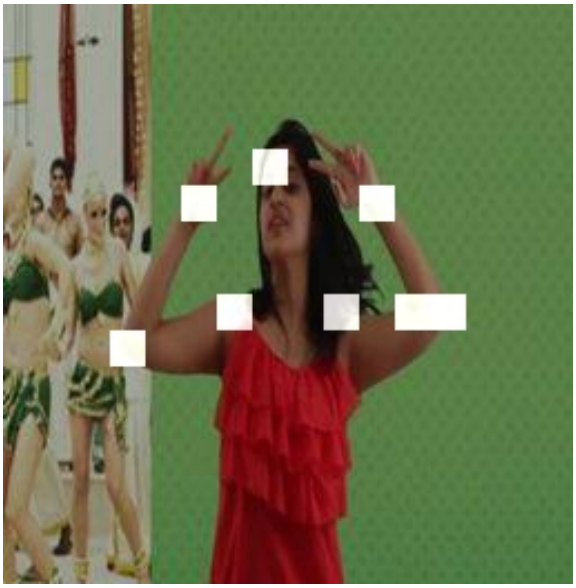
F1-Score = 0.7492

N = 256	Predicted Negative	Predicted Positive	
Actual Negative	247(TN)	2(FP)	249
Actual Positive	2(FN)	5(TP)	7
	249	7	

Precision = 0.7142

Recall = 0.7142

F1-Score = 0.7142

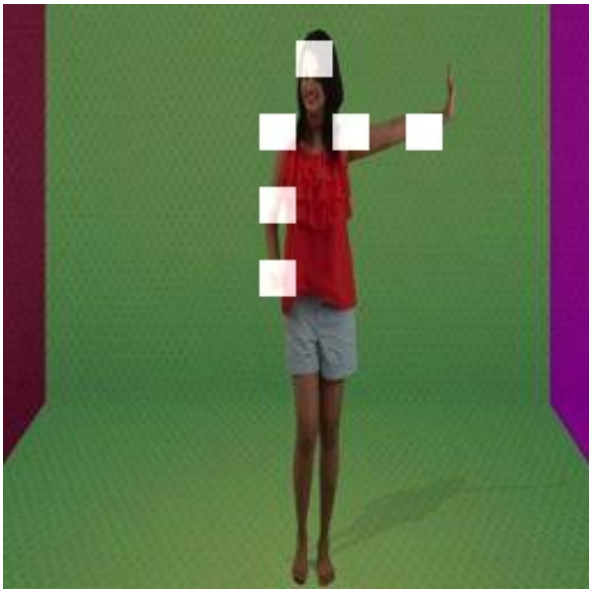


N = 256	Predicted Negative	Predicted Positive	
Actual Negative	246(TN)	2(FP)	248
Actual Positive	2(FN)	6(TP)	8
	248	8	

Precision = 0.75

Recall = 0.75

F1-Score = 0.75

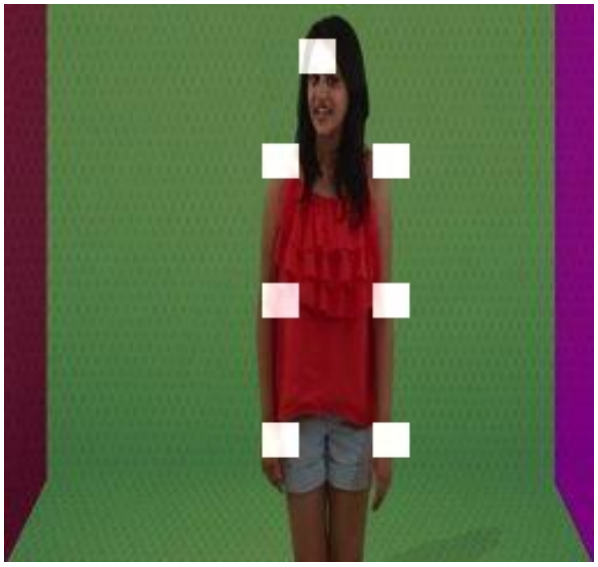


N = 256	Predicted Negative	Predicted Positive	
Actual Negative	247(TN)	2(FP)	249
Actual Positive	3(FN)	4(TP)	7
	250	6	

Precision = 0.666

Recall = 0.571

F1-Score = 0.614



N = 256	Predicted Negative	Predicted Positive	
Actual Negative	247(TN)	1(FP)	248
Actual Positive	2(FN)	6(TP)	8
	249	7	

Precision = 0.8571

Recall = 0.75

F1-Score = 0.8

N = 256	Predicted Negative	Predicted Positive	
Actual Negative	243(TN)	2(FP)	245
Actual Positive	4(FN)	7(TP)	11
	247	9	

Precision = 0.777

Recall = 0.6363

F1-Score = 0.7

N = 256	Predicted Negative	Predicted Positive	
Actual Negative	245(TN)	1(FP)	246
Actual Positive	2(FN)	8(TP)	10
	247	9	

Precision = 0.888

Recall = 0.8

F1-Score = 0.8413

N = 256	Predicted Negative	Predicted Positive	
Actual Negative	248(TN)	1(FP)	249
Actual Positive	1(FN)	6(TP)	7
	249	7	

Precision = 0.8571

Recall = 0.8571

F1-Score = 0.8571

N = 256	Predicted Negative	Predicted Positive	
Actual Negative	248(TN)	2(FP)	250
Actual Positive	1(FN)	5(TP)	6
	249	7	

Precision = 0.7142

Recall = 0.833

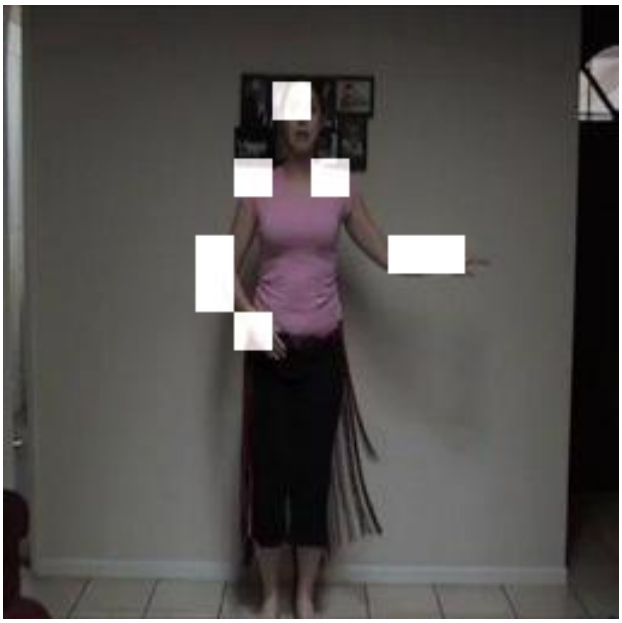
F1-Score = 0.7688

N = 256	Predicted Negative	Predicted Positive	
Actual Negative	247(TN)	3(FP)	250
Actual Positive	2(FN)	4(TP)	6
	249	7	

Precision = 0.5714

Recall = 0.66

F1-Score = 0.6125



N = 256	Predicted Negative	Predicted Positive	
Actual Negative	246(TN)	3(FP)	249
Actual Positive	2(FN)	5(TP)	7
	248	8	

Precision = 0.625

Recall = 0.7142

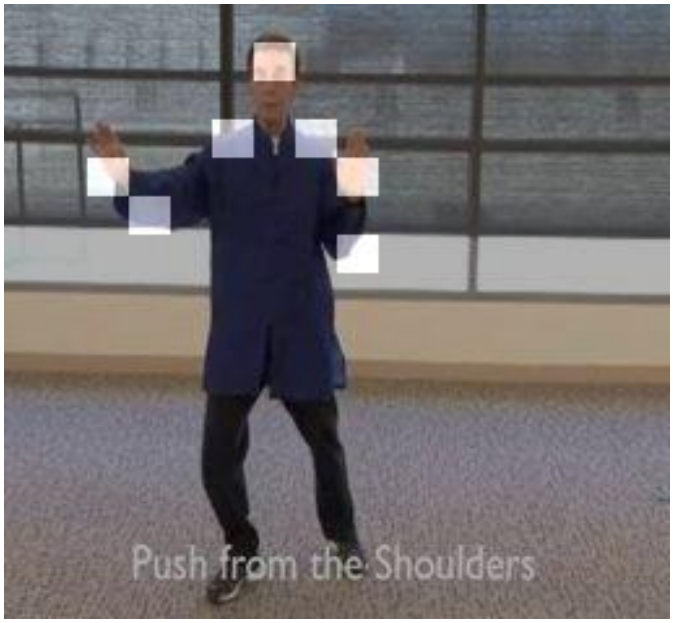
F1-Score = 0.666

N = 256	Predicted Negative	Predicted Positive	
Actual Negative	247(TN)	1(FP)	248
Actual Positive	1(FN)	6(TP)	7
	248	7	

Precision = 0.8571

Recall = 0.8571

F1-Score = 0.8571



N = 256	Predicted Negative	Predicted Positive	
Actual Negative	247(TN)	1(FP)	248
Actual Positive	2(FN)	5(TP)	7
	249	6	

Precision = 0.83

Recall = 0.7142

F1-Score = 0.7677

N = 256	Predicted Negative	Predicted Positive	
Actual Negative	248(TN)	0(FP)	248
Actual Positive	1(FN)	7(TP)	8
	249	7	

Precision = 1

Recall = 0.875

F1-Score = 0.933

N = 256	Predicted Negative	Predicted Positive	
Actual Negative	247(TN)	1(FP)	248
Actual Positive	2(FN)	6(TP)	8
	249	7	

Precision = 0.8571

Recall = 0.75

F1-Score = 0.8

Now, from calculating the precision and recall from the confusion matrices, we get an estimate of where the joints are located from the detected white (label = 1) patches as we have shown above in some of the test images. Now, to check mathematically, by how much are the predicted and ground truth images similar we can use the Euclidian distance metric.

It means that we have already mentioned previously that each image has seven joints and their locations in 2D plane. Also, in the predicted image, we have got the seven predicted patches So, we compute the mean x and y positions for all the ground truth and the predicted images, and compute the distance between the mean points. The more the distance, less is the similarity between the ground truth and the predicted joints.

After giving the test image as input, we can make a check that if the predicted output for the patch is 1 (joint) then we return the mid-point of that patch as well. Therefore, we again get 14 co-ordinates of the seven joints from which we compute the error metric. In the table given below, we measure the mean Euclidean distances of the test images of the test set with respect to the ground truths.

The Euclidean distance between two points (x,y) and (x',y') is given by

edist = square root ((x – x')^2 + (y – y')^2)

Ground truth mean (x,y)	Predicted mean (x’,y’)	Distance
123.88,154.09	127,144	10.56
112.13,124.34	118,130	8.15
145.67,156.72	150,162	6.82
135.19,201.07	132,196	5.99
245.13,135.34	230,147	19.1
112.17,120.89	109,118	4.28
114.06,236.17	95,245	21.0
101.08,103.67	100,103	1.27
240,123.76	128,108	113.1
112.13,117.69	112,115	2.69
117.65,128,17	101,98	4.85
208.16,215.77	187,203	11.96
111.13,176.37	124,156	14.76
113.15,107.59	108,115	5.87
110.87,102.45	117,87	14.76
210.01,198.12	215,193	6.81
113.89,127.15	101,125	7.21
113.78,122.12	112,116	1.27
87.54,111.07	93,119	4.89
112.16,95.76	102,89	3.67
231.67,126.15	236,120	6.88
156,230.15	145,208	6.65
124.12,136.17	113,145	8.95
87.59,113.67	89,109	10.56
124.12,137.95	108,116	9.67
86.13,127.15	88,125	7.58
98.16,145.16	99,135	8.56

88.67,135.57	102,113	6.87
123.78,114.89	105,114	11.45
145.78,134.78	114,132	7.65
156.18,208.19	152,206	4.56
234.78,134.67	232,1301	6.76
111.56,107.13	110,105	7.43
111.08,109.12	110,109	2.45
89.15,123.47	110,128	11.25
111.56,109.12	105,115	5.66
78.14,123.67	72,129	7.92
145.13,136.17	140,132	5.34
111.89,107.52	109,115	2.43
189.14,134.68	189,132	1.89
150.45,167,13	145,173	4.37
148.34,167.55	146,1641	2.64
134.16,167.59	113,162	4.87
140.18,156.89	142,159	3.66
123.14,127.12	120,127	5.15
134.16,146.18	138,145	6.23
145.18,156.89	139,158	6.88
111.67,120.67	109,116	4.37
109.12,167.13	107,162	2.66
109.67,111.19	108,110	3.75
167.14,139.57	157,135	8.68
115.15,167.45	14,160	5.43
145.78,157.12	143,152	3.45
136.67,157.47	133,149	4.87
145.35,146.78	140,143	3.62
135.17,124.78	132,125	3.66
111.73,124.56	109,119	3.57
156.23,165.14	154,162	3.58

123.67,187.15	120,184	4.87
145.13,167.23	138,156	4.96
143.67,124.18	145,123	5.2
155.67,176.12	160,174	3.78
198.67,201.45	187,202	4.8
145.46,176.45	139,179	6.66
126.09,110.18	129,108	5.85
134.78,156.85	132,155	4.63
165.56,187.67	167,182	3.81
143.15,187.56	139,185	3.86
143.75,176.45	140,178	2.37
136.13,157.34	135,162	1.87
182.17,195.17	189,193	6.63
122.17,145.14	124,152	5.26
156.89,144.56	159,147	4.77
134.89,154.13	132,156	2.14
156.23,166.18	159,167	3.65
176.34,156.89	156,149	5.88
120.45,106.57	89,112	13.76
156.78,123.67	152,134	5.8
163.16,164,87	157,163	5.89
201.17,103.45	197,102	4.76
154.88,145.17	145,167	5.25
123.89,156.87	126,159	5.43
134.76,145.16	128,142	2.37
123.56,155.15	118,164	4.63
132.67,154.78	127,152	7.65
234.14,134.16	219,135	5.44
135.17,187.56	138,188	3.87
137.16,134.27	132,156	17.56
128.12,164.76	134,170	16.42

132.78,164.15	132,159	5.85
167.46,164.67	162,157	6.76
124.67,154.38	121,137	13.67
122.67,154.78	118,145	10.86
123.45,164.14	119,165	9.56
111.24,109,87	110,97	8.56
98.34,115.17	119,108	6.32
134.56,176.89	113,164	11.88
124.87,192.72	132,188	12.45
122.56,137.18	117,135	6.15
154.17,149.19	159,142	5.47
145.67,124.15	138,122	7.23
122.56,145.78	121,142	3.15
145.78,109.16	145,108	4.78
111.67,89.92	110,89	1.24
108.65,117.56	107,115	3.67
102.67,108.15	198,112	5.86
111.07,109.85	108,111	6.74
108.56,117.81	110,119	5.67
127.67,189.12	125,192	7.86
115,98,176.89	110,175	10.67
136.23,134.34	132,126	18.34
134.12,145.16	132,145	8.13
111.19,108.15	110,108	6.68
98.18,107.65	105,112	4.46
111.43,108.45	112,108	2.23
143.67,154.16	139,154	3.65
89.16,115.76	93,118	4.56
135.89,147.56	138,145	3.89
112.34,109.87	108,110	2.87
115.16,109.86	105,109	4.67

107.56,112.56	101,110	6.76
86.47,92.14	108,101	8.71
112.45,118.76	108,112	6.78
128.17,145.76	125,138	5.64
112.96,117.56	112,117	1.23
109.38,110.17	107,104	2.47
134.96,155.47	138,156	2.87
108.19,112.67	110,98	4.98
98.13,113.78	123,114	1.09
102.87,109.15	99,107	2.45
115.67,108.86	106,114	3.56
123.87,149.56	126,146	5.67
187.67,145.38	185,148	7.85
111.67,192.18	95,178	4.56
136.24,165.78	134,167	5.76
145.07,187.16	143,192	9.56
156.67,198.15	158,195	4.76
104.76,112.17	110,109	6.78
119.46,103.26	117,106	2.45
117.45,108.18	118,110	4.56
145.78,159.66	143,156	4.83
115.87,120.56	114,121	1.76
240.23,187.56	245,188	1.49
109.45,112.34	110,113	2.01
104.16,114.65	109,115	5.67
107.16,114.65	107,114	0.86
119.76,112.76	120,113	1.37
127,135.78	127,138	1.08
151.12,237.56	149,238	2,13
120.56,111.45	118,107	3.12
78.77,168.16	79,165	3.15

112.45,108.14	111,108	1.45
116.17,109.87	109,106	1.87
145.78,101.18	132,126	11.76
119.87,107.86	114,117	8.76
113.67,109.14	109,115	7.65
119.56,110.56	112,102	7.42
127.68,113.17	122,117	4.86
129.65,110.56	134,109	11.21
230.56,108.45	245,108	8.23
119.12,115.78	122,118	4.86
104.26,112,56	108,111	2.57
187.45,119.85	175,132	5.56
113.56,109.56	110,107	2.87
54.67,167.85	52,163	5.97
117.38,109.64	115,107	2.07
116.45,118.15	112,108	11.87
113.67,157.96	111,147	11.12
86.47,92.14	89,94	2.13
112.45,118.76	110,115	3.16
128.17,145.76	125,142	6.87
112.96,117.56	109,115	5.67
109.38,110.17	107,111	2.46
134.96,155.47	122,148	11.45
108.19,112.67	110,109	3.67
98.13,113.78	102,111	4.23
102.87,109.15	95,109	9.56
115.67,108.86	115,109	1.02
123.87,149.56	122,151	2.85
87.134,119.45	111,117	12.16
128.27,193.29	127,184	13.87
110.78,116.57	115,108	12.56

109.12,113.46	111,108	5,13
120.45,145.18	123,136	5.16
112.56,108.56	101,98	8.74
89.56,197.56	92,195	3.67
134.56,176.15	128,166	4.76
123.67,110.56	118,109	4.67
134.78,119.36	124,88	11.19
87.78,192.781	101,189	5.67
56.14,137.65	57,142	5.13
110.76,109.15	110,108	0.99
89.99,108.45	93,115	12.89
119.78,116.85	118,121	5.56
93.76,119.67	98,115	6.77
201.34,187.68	197,189	6.87
117.67,110.87	110,123	13.56
116.45,107.34	112,118	11.78
118.76,109.24	109,115	9.24
112.75,110.76	113,108	2.36
167.89,156.97	154,163	14.36
135.76,145.89	132,147	3.85
127.36,189.56	125,187	2.52
116.45,110.65	120,111	2.87
89.15,118.87	90,119	1.43
109.45,154.87	123,149	5.67
119.45,110.75	117,115	6.83
188.55,192.87	185,186	7.65
201.56,184.27	201,187	6.58
110.84,187.56	118,183	5.52
117.34,98.67	115,96	2.16
198.35,203.67	185,199	3.18
112.89,213.56	113,209	3.14

176.35,109.76	156,119	13.76
113.28,109.84	110,115	6.87
94.78,109.87	92,108	2.24
115.98,107.34	113,108	2.54
110.45,94.28	115,96	5.23
145.78,169.45	144,165	5.87
104.57,145.58	103,148	3.15
125.87,189.54	123,179	4.56
118.47,109.45	123,109	2.57
128.37,110.89	127,110	3.66
182.39,118.94	183,123	4.82
98.27,115.98	98,114	5.04
134.86,229.78	137,238	11.15
105.67,115.89	103,114	2.43
109.38,88.56	108,91	3.12
138.49,119.56	136,119	2.89
129.49,156	127,153	3.78
112.46,99.85	111,99	2.12
129.45,167.89	127,164	2.34
154.76,189.94	153,183	5.67
134.85,198.56	136,196	3.02
230.56,109.86	243,107	8.73
185.85,123.56	188,121	3.86
110.56,106.56	107,115	4.56
156.58,109.67	149,108	2.14
185.67,154.86	183,149	3.62
112.45,108.56	110,107	2.45
115.67,108.45	114,110	2.35
134.98,118.39	128,117	3.12
106.56,112.34	104,117	4.23
123.67,109.45	122,107	2.56

123.56,109.56	122,110	2.45
118.45,137.56	115,132	4.65
113.28,109.87	110,116	2.341
201.98,118.75	198,121	11.34
119.56,102.34	118,101	4.67
98.67,99.180	113,99	5.32
118.56,101.87	115,102	3.46
184.56,198.45	182,198	19.15
108.45,116.57	108,115	14.37
146.37,127.48	142,126	8.54
118.29,107.56	119,111	6.63
134.68,119.89	137,121	4.65
96.27,119.78	98,120	5.52
134.87,106.75	139,109	6.36
122.67,109.86	123,112	5.23
119.28,118.98	123,117	4.07
135.36,116.35	132,108	8.89
197.17,108.65	194,111	8.65
123.67,109.45	116,105	6.87
154.67,115.98	149,127	5.76
194.57,220.18	186,218	4.58
190.67,118.39	193,117	4.89
102.56,129.56	105,131	3.87
196.49,185.84	199,187	11.38
194.58,118.45	186,123	5.15
119.49,107.89	120,107	1.45
109.67,98.17	109,98	2.03
129.48,113.20	127,118	5.78
128.38,110.48	127,109	1.24
108.19,117.38	111,115	6.89
103.45,110.38	109,115	5.86

135.36,116.35	132,119	2.34
197.17,108.65	187,111	3.46
123.67,109.45	118,118	4.58
154.67,115.98	152,103	13.67
194.57,220.18	195,221	1.23
190.67,118.39	188,178	16.14
102.56,129.56	104,118	24.64
196.49,185.84	196,184	1.89
194.58,118.45	188,121	3.56
78.14,123.67	81,129	6.76
145.13,136.17	139,142	6.56
111.89,107.52	113,108	7.27
189.14,134.68	188,134	1.25
150.45,167,13	150,167	1.08
148.34,167.55	148,164	3.56
134.16,167.59	128,167	6.57
140.18,156.89	138,157	4.37
123.14,127.12	127,128	3.28
134.16,146.18	134,145	2.34
145.18,156.89	145,158	3.67
111.67,120.67	109,114	6.17
109.12,167.13	114,156	12.56
109.67,111.19	110,113	11.34
167.14,139.57	156,141	2.56
115.15,167.45	109,115	4.76
145.78,157.12	135,165	13.27
136.67,157.47	132,156	12.16
145.35,146.78	148,152	5.6
135.17,124.78	138,138	6.4
111.73,124.56	112,128	4.12
156.23,165.14	154,163	2.65

123.67,187.15	123,185	2.45
187.45,119.85	182,120	5.12
113.56,109.56	111,108	2.16
54.67,167.85	56,165	2.13
117.38,109.64	115,110	2.23
116.45,118.15	114,118	3.46
113.67,157.96	112,157	1.57
86.47,92.14	89,91	3.18
112.45,118.76	112,117	1.09
128.17,145.76	127,142	3.56
112.96,117.56	112,116	5.67
109.38,110.17	109,111	0.93
134.96,155.47	132,151	4.92
108.19,112.67	111,109	7.34
98.13,113.78	99,112	1.57
128.37,110.89	126,110	2.56
182.39,118.94	187,121	5.65
98.27,115.98	99,118	5.08
134.86,229.78	138,225	4.13
105.67,115.89	105,114	1.27
109.38,88.56	108,921	2.47
138.49,119.56	138,119	3.68
129.49,156	129,156	1.85
112.46,99.85	112,99	1.82
129.45,167.89	129,167	0.88
154.76,189.94	156,184	4.59
134.85,198.56	134,194	2,86
230.56,109.86	231,110	2.85
185.85,123.56	187,129	3.62
110.56,106.56	111,109	2.43
156.58,109.67	157,111	2.54

185.67,154.86	182,156	2.45
110.45,94.28	109,97	4.67
145.78,169.45	142,166	11.54
104.57,145.58	104,145	0.98
125.87,189.54	125,188	0.81
118.47,109.45	123,111	2.34
128.37,110.89	127,110	3.12
182.39,118.94	183,120	4.67
98.27,115.98	99,118	2.46
134.86,229.78	135,226	3.09
105.67,115.89	108,119	5.01
109.38,88.56	112,91	4.26
138.49,119.56	141,123	4.86
189.14,134.68	187,135	4.56
150.45,167,13	145,163	8.25
148.34,167.55	149,165	2.15
134.16,167.59	135,165	2.37
140.18,156.89	143,156	2.53
123.14,127.12	124,128	2.51
134.16,146.18	134,143	4.67
145.18,156.89	146,152	6.87
112.56,156.67	101,154	7.34
112.76,109.67	111,109	1.86
104.57,113.56	104,119	5.65
118.45,156.78	121,159	4.32
113.56,109.45	118,109	4.87
145.89,106.56	143,108	2.24
123.76,109.56	121,109	1.94
118.67,164.69	122,157	9.17
116.45,118.17	113,119	6.27
113.46,102.56	112,106	5.43

127.56,113.56	124,110	4.67
123.78,119.37	121,119	2.15
145.67,114.76	141,112	3.46
134.66,187.87	136,185	4.59
98.13,127.16	101,126	3.56
114.56,118.67	114,118	4.66
123.56,156.47	124,148	12.45
112.56,143.56	111,142	1.98
121.56,132.56	120,128	3.56
138.56,144.23	136,143	3.46
154.34,128.14	152,127	5.32
110.24,118.85	108,117	2.76
134.76,145.15	127,138	6.86
120.67,114.52	123,118	5.88
132.56,148.76	131,144	2.65
123.86,119.79	123,119	3.42
108.56,231.56	108,231	4.36
154.87,123.25	154,123	3.59
118.29,107.27	109,106	1.65
113.89,118	112,118	2.34
107.56,103.12	106,102	1.87
182.34,117.87	181,116	1.97
121.25,134.17	118,132	3.65
143.29,158.59	142,156	2.45
132.45,124.56	131,126	2.85
109.23,89.94	111,102	7.67
118.23,113.46	116,113	2.46
123.45,115.47	121,115	2.54
107.34,111.12	105,110	1.97
176.73,183.45	174,181	2.88
157.13,162.45	156,161	1.78

112.13,109.34	110,108	2.45
118.45,104.35	118,105	1.87
98.17,113.45	97,114	1.79
127.67,132.45	125,134	2.88
118.57,138.98	115,142	5.02
154.76,181.95	155,176	5.08
134.74,104.57	134,105	1.02
117.43,157,47	116,154	3.45
111.18,145.07	111,145	1.08
134.13,106.57	134,106	1.06
164.76,119.86	156,118	7.89
152.56,149.75	149,151	3.45
128.34,133.56	132,135	4.17
154.18,137.43	154,136	1.15
118.81,127.53	121,126	3.07
125.46,137.15	125,137	1.12
140.56,133.58	139,133	1.07
96.06,101.37	101,103	5.67
102.87,112.34	102,113	1.27
87.56,118.74	101,121	5.08
134.27,118.34	136,119	2.45
129.15,137.26	129,137	1.08
119.56,102.14	119,102	1.07
116.34,108.34	115,109	1.95
125.67,119.87	125,118	0.97
106.32,119.79	106,120	1.12
113.76,118.92	113,117	1.23
120.65,126.86	120,127	1.31
134.86,116.52	132,118	2.88
112.54,136.98	113,139	1.78
113.16,109.16	112,109	1.67

123.45,108.56	121,108	2.56
113.45,117.65	112,118	1.87
98.45,102.56	99,101	1.76
118.45,103.45	117,104	1.92
97.65,118.45	96,119	1.88
127.74,136.54	126,137	1.83
118.13,145.33	126,154	14.78
123.18,127.56	121,119	13.26
145.24,193.17	142,191	3.87
156.35,118.87	155,119	1.84
134.18,143.87	132,141	2.54
129.67,132.45	131,137	5.36
136.87,110.86	138,112	2.88
132.67,102.34	131,103	1.97
119.56,111.36	118,109	1.99
143.86,126.56	137,127	6.02
123.18,194.27	122,192	2.08
106.87,118.15	108,119	2.11
119.56,108.54	119,111	3.03
111.34,176.45	111,176	1.65
118.45,109.62	117,112	3.56
104.28,113.45	103,117	5.01
118.43,103.67	112,104	5.08
134.65,176.45	132,176	2.25
118.23,110.56	118,110	1.03
103.56,102.87	103,101	1.07
112.34,109.82	111,108	1.21
134.56,143.37	132,144	2.52
143.18,152.67	137,151	6.05
112.56,108.45	110,109	2.46
118.97,109.12	115,107	3.85

Chapter 6: Conclusion

Although our model of patch classification works fairly well on our dataset, but our model does not explicitly identify, the type of joint it represents i.e. whether it is head, wrist, elbow or shoulder. Moreover, it cannot detect whether the person in the picture or video is looking at the camera or has his back in front of the camera. Also, there may be the case, when a person is standing in front of the mirror, the joint locations of right side and left side are reversed. So, the detected pose in those cases might be ambiguous. Apart from these, the number of training samples in this methodology is extremely high which takes up a long time for training and testing as well because, we are also segmenting the test image to classify each individual patch obtained from it.

However, the main reason we implemented this model in our work is the fact that since, we are treating each joint as an object, irrespective of the image from which it is obtained, generally too much of background information is not captured which reduces overfitting significantly.

Although, the dataset we used are 2D images framed from videos, the results can be used to make a 3D projection of the 2D pose and thus enabling researchers to detect joints in 3D images or videos.

References

[1] Yap Wooi Hen and Raveendran Paramesran, “Single camera 3D human pose estimation: A Review of current techniques”, in *International Conference for Technical Postgraduates(TECHPOS),*2009, pp. 1-8.

[2] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake, “Real-Time Human Pose Recognition in Parts from Single Depth Images”, in *CVPR 2011*,2011, pp. 1297-1304.

[3] Greg Mori and Jeetendra Malik, “Estimating human body configurations using shape context matching,” in *European conference on computer vision.* Springer, 2002, pp. 666-680.

[4] Tomas Pfister, Karen Simonyan, James Charles and Andrew Zisserman, “Deep convolutional neural networks for efficient pose estimation in gesture videos,” in *Asian Conference on Computer Vision,* Springer, 2014, pp. 538-552.

[5] Alexander Toshev and Christian Szegedy, “Deeppose: Human pose estimation vis deep neural networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* 2014, pp. 1653-1660.

[6] A. Krizhevsky, I. Sutskever and G. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks” in *NIPS*,2012

[7] Le Hou, Dimitris Samaras, T.M. Kurc, J.E. Davis, and J.H Saltz, "Patch-based Convolutional Neural Network for Whole Slide Tissue Image Classification” in *IEEE Conference on Computer Vision and Pattern Recognition,* 2016.