Human Pose Estimation Using Landmark Points

CSD 316: Introduction to Machine Learning
Final Project

Aditya Jain, Manav Prabhakar, Prakhar Rathi, Salil Saxena

December 7, 2020

Abstract

The problem of human pose estimation has received extensive attention from the Computer Vision community, largely due to its numerous applications and potential to impact multiple industries. Even though, there has been over 15 years of research in the field, it continues to challenge researchers across the world. In this paper we, aim to build a classification method to distinguish between different human poses using landmark points, generated from annotated images. We also build an application which takes an image input and outputs the annotated image the pose-class it belongs to.

Contents

1 Introduction **2**

2 Literature Survey **3**
2.1 DeepPose: Human Pose Estimation via Deep Neural Networks (2014) 3
2.2 Articulated Pose Estimation: Pose estimation of arms using Discriminative Armlet Classifiers 4
2.3 OpenPose: Realtime Multi-person 2d pose estimation using PAFs 4

3 Method Description **5**
3.1 Preprocessing 5
3.2 Model Building 6
3.2.1 k-Nearest Neighbours Algorithm 7
3.2.2 Modified k-Nearest Neighbours Algorithm 7
3.2.3 Perceptron Learning Algorithm 8
3.3 Logistic Regression 8
3.4 Image Annotation 9

4 Experiments and Results **9**
4.1 Datasets 9
4.2 Experimental Setup 9
4.3 Training 10
4.4 Testing 12
4.5 Inference 13

5 Conclusions and Future Scopes **13**

6 Acknowledgement **14**

1 Introduction

Human Pose Estimation (HPE) is the process of estimating the orientation of a body and classifying it into one of the desired poses. The problem of human pose estimation is also defined as the problem of localization of human joints (also known as key-points, elbows, joints, wrists, etc) and has been discussed extensively in the computer vision community and has been studied for over 15 years, particularly owing to the abundance of applications which employ such technology and those that will benefit from it. Moreover, they also prove to be crucial for fields like human-computer interaction, activity recognition and Gait analysis.

Even after years of research, HPE remains a difficult and largely unsolved problem with many challenges hindering it's perfection. These include variable lighting conditions, differences in human physical structures, loss of information by representing a 3D structure onto 2d image projections, hidden objects due to image angles etc.

An important aspect of HPE is generating a Human Pose Skeleton which represents the orientation of a person in a graphical format using landmark points. It is, essentially, a set of points that can be connected with each other to form a skeleton like frame which can help in describing the pose of the person. Each point within the skeleton structure is called a joint or a landmark point, and an edge between two joints is called a limb or a pair. However, not all connections will yield a valid skeleton, thereby, making the problem quite complicated.

Figure 1: Left: COCO keypoint format for human pose skeletons. Right: Rendered human pose skeletons. [5]

Knowing the configuration of a human pose can lead to several real-life applications, some of which include:-

- **Third-Person Activity Recognition:** These include tracking the changes in the pose of a person over a duration and then predict what activity they are performing, perform gait analysis, track higher speed objects etc. These applications have many more specific use-cases which are outside the scope of this paper.
- **Motion Capture and Augmented Reality**: Currently, CGI is rendered over green screens and suits which is done manually by a team of video editors and designers. However, the graphics can be superimposed on characters automatically if their pose can be estimated as they move around in the frame. The same can be used for AR rendering.
- **Automation and Robotics:** Robots and automated vehicles are primarily programmed manually to follow trajectories but with advanced HPE techniques one could train robots to follow human pose skeletons performing an action by following the landmark points. This would lead to higher amount of training data and faster save time in programming them manually.

Problem Statement: In this project, we also try to solve the problem of human pose estimation using landmark points. We will also perform a comparative study of multiple algorithms to see which ones are doing better on this problem.

2 Literature Survey

2.1 DeepPose: Human Pose Estimation via Deep Neural Networks (2014)

The DeepPose paper by [1] forms the base for our work in Human Pose Estimation. They have proposed a method for human pose estimation which uses Deep Neural Networks

(DNNs). The body joints are taken as landmark points and the estimation problem is converted to a regression problem. These multiple regressors are then cascaded together to create a high-precision human pose estimator. The authors have relied on the recent developments in deep learning to propose a novel architecture which gave better results than then state-of-the-art (SOTA) models in pose estimation. They have leveraged the performance of DNNs on visual classification [2] and object localization [3]. Their work provides a simple formulation of holistic human pose estimation as a joint regression problem. They regress the location of each body joint and build a 7-layered generic convolutional neural network. On top of that they cascade several DNN pose predictors which convert the final task into a classification problem into multiple poses. They outperform the SOTA models four major components - Arms, Legs, Elbows, Wrists.

2.2 Articulated Pose Estimation: Pose estimation of arms using Discriminative Armlet Classifiers

In real-world everything is cluttered, the paper [6] proposes an approach for human pose estimation for such real-world scenes, focussing on the prediction of the pose of both arms of each person in the image. The approach was built upon the notion of poselets (classifiers to detect different body parts), and armlets(highly discriminative classifiers to differentiate among arm configurations). It also proposes a rich representation which integrates the features such as strong contours, skin color, contextual cues, with the addition of HOG features. The test of this approach was done on a large subset of images taken from the PASCAL VOC detection dataset, where critical visual phenomena, such as occlusion, truncation, multiple instances and clutter are the norm. This approach outperforms the state-of-the-art technique used in [4], with the figures of improvement being from 29.0% ti 37.5% PCP accuracy on the arm keypoint prediction task.

2.3 OpenPose: Realtime Multi-person 2d pose estimation using PAFs

The paper [7] proposes an approach for Realtime 2D pose estimation of multiple people in a single image which came out to be a key component in enabling the machines to understand the people better in an image. Proposal of this paper was a method which uses a nonparametric representation also referred to as Part Affinity Fields(PAFs), through which the learning of association of body parts to different individuals in an image is done. High accuracy was achieved with real time performance, using the bottom-up system, even for a good amount of people. The previous paper, the refinement of PAFs and body part location estimations was done simultaneously across different training stages, however this paper shows that much better results (Runtime performance and accuracy) are achieved when the refinement of PAFs only is done, rather than both the PAFs and body part location estimations. In addition to the above mentioned proposal, this paper also presents a detector of combined body and foot keypoint, which was based on an internal annotated foot dataset, publicly released by the authors of this paper. The combined detector reduced the inference time which was more in the case of sequential execution, and also maintains the accuracy of

each component.

3 Method Description

3.1 Preprocessing

This was the most important aspect of solving this machine learning problem. We had to represent our images as a set of landmark points and then convert it into a data format which would work well with our models. This included selecting the best features, representing them in a model-friendly format, preparing the annotated dataframe and applying the relevant scaling techniques.

After collecting the complete dataset from the source [9], we took a .mat file of annotations of around 17K images of over 400 human activities out of which we only took 2 classes - **walking** and **bicycling** because we observed that taking location of body landmarks from 2D images isn't a very good choice for classification (as distance and angles from a landmark to adjacent landmark will change chaotically, and due to this chaotic nature the separation of of classes on the basis of an input space will be very difficult). The .mat file contained X and Y co-ordinate of each landmark point and have been listed in Figure 2.

```
Columns:
        Index(['NAME', 'r ankle_X', 'r ankle_Y', 'r knee_X', 'r knee_Y', 'r hip_X',
       'r hip_Y', 'l hip_X', 'l hip_Y', 'l knee_X', 'l knee_Y', 'l ankle_X',
       'l ankle_Y', 'pelvis_X', 'pelvis_Y', 'thorax_X', 'thorax_Y',
       'upper neck_X', 'upper neck_Y', 'head top_X', 'head top_Y', 'r wrist_X',
       'r wrist_Y', 'r elbow_X', 'r elbow_Y', 'r shoulder_X', 'r shoulder_Y',
       'l shoulder_X', 'l shoulder_Y', 'l elbow_X', 'l elbow_Y', 'l wrist_X',
       'l wrist_Y', 'Scale', 'Activity', 'Category'],
      dtype='object')
```

Figure 2: The different features in the annotated dataset

The only features that seemed important were: landmark co-ordinate, Scale, Activity. So naturally we flushed out all the other columns and proceeded for further data exploration. Another thing which we observed was that the points that were not visible in the given image were marked as 1 but we saw that count of head's non-visible Y coordinates was greater than the X-coordinate, so we assumed it as a discrepancy and removed the head's X and Y coordinate as a feature. The neck coordinates were also removed after removing those of the head as we trivially assumed that the neck base coordinate would have little effect on classification without the head coordinates.

After keeping all the other land mark points, we came across Google's Posenet model [12], which only had below following common features with our dataset. Since, PoseNet would be required for the inference, we decided to select these features. ['left Elbow', 'right Elbow', 'left Wrist', 'right Wrist', 'left Hip', 'right Hip', 'left Knee', 'right Knee', 'left Ankle', 'right Ankle'].

After finalizing the landmark points which would be required for classification, we took the base 2 classes for classification which were Walking(1) and Bi-Cycling(-1). These were selected because they are the classes with closest prior probabilities and the most differentiable classes.

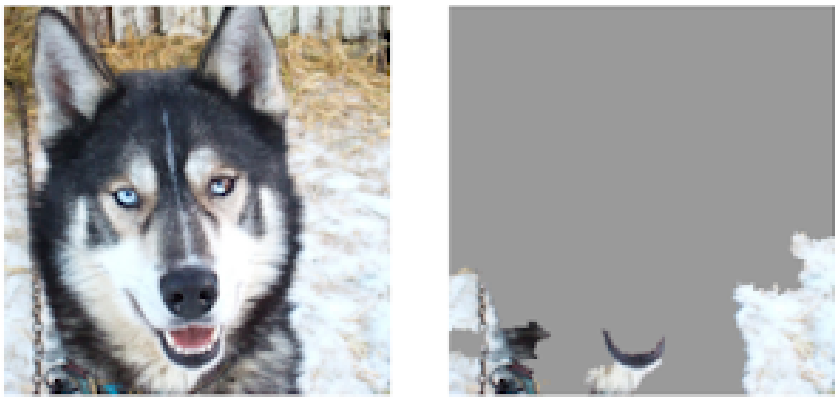
Feature Name	Description
l ankle_knee	distance between the left ankle and left knee
r ankle_knee	distance between the right ankle and right knee
l knee_hip	distance between the left knee and right hip
r knee_hip	distance between the right knee and right hip
l wrist_elbow	distance between the left wrist and left elbow
r wrist_elbow	distance between the right wrist and right elbow
l elbow_shoulder	distance between the left elbow and left shoulder
r elbow_shoulder	distance between the right elbow and right shoulder

Table 1: Features used for final modelling

The cleaning and filtering of the data had been achieved. Our next step was to process the input features for our models. There were two features that were decided upon. The first one was the angle and the second one was the distance between any two adjacent landmark points. Considering the angle as an input feature brought in some other complications, particularly due to the angle from which an image was captured. So making input space from 2D image landmark wasn't the best option. Other possible solutions have been discussed for this problem in Section 5.

3.2 Model Building

Once the data was completely processed and compiled into a structured format, we moved on to the modelling phase. The first task was to select the models we would be using. We were facing the performance-interpretability trade-off. On one hand, there were implementations which use deep neural networks for classification of these poses while not explaining much about why they chose those models or why they got the results in the manner they did. On the other hand, we could choose models which didn't have very high performance but really explain how we achieve those results. This is a very common problem in image classification tasks and can often lead to results like this:-



(a) Husky classified as wolf (b) Explanation

Figure 3: Explainability issues with black-box models [11]

In image 3, a black-box model with high performance but no explainability is displaying the important features in the image and turns out that it's actually not even looking at the animal and focusing on the snow instead. This often happens with black-box models which are not explained correctly. Keeping this aspect in mind, we decided to implement and test the following algorithms.

3.2.1 k-Nearest Neighbours Algorithm

Basic K-nearest-neighbour algorithm also known as the "Lazy learner" algorithm is used to classify the data by looking at the data points around it, or looking at the K number of nearest data points. This algorithm is very important in the field of image classification as it helps us identify the most similar points for a particular point.

The algorithm takes into account the data points of the testing set one by one and calculates their distances with respect to all other data points in the training set, and are sorted in ascending order. The first K distances are taken, which means the K number of nearest neighbours are taken, and the sum of all the occurrences of different class labels is found out. The Class label which has maximum occurrence, is taken as the class label to classify the data point.

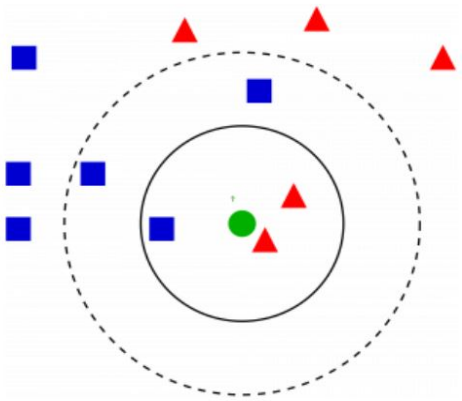


Figure 4: Depiction of how KNN algorithm works [10]

In Figure 4, the new point (*in green*) would be classified with red points for $k = 3$ and classified with blue points for $k = 4$.

One implementation of the algorithm starts by finding the optimal value of K by dividing the training set into a number of folds, so that one fold can be taken as the validation set, and others can be treated as the training data. This gives an optimal value for k. This is called cross-validation.

3.2.2 Modified k-Nearest Neighbours Algorithm

After training the model using k-Nearest Neighbours, we found that results were not very impressive. We realised that one of the issues with the classical kNN is that it depends heavily on our choice of 'k' value. If the value of k is very small, then algorithm would be

more susceptible to outliers and small disturbances and if the value is too large, then we end up getting points from different categories as well which reduced the result accuracy. Secondly, a majority vote does not give the best result in a case where closer points might give more information about the point that we are classifying. So, we tried to give higher weights to points which were closer because there was a large variation in the distances of the closest points. Our intuition behind the algorithm was to give more weight to the points which are closer to the test data point and lower weights to points which are further away. This was done using a function which allots decreasing weight values as the distance from point increases.

The function is based on [8]. Let the distances of the k Nearest Neighbours be in an ordered set = $\{d_1, d_2, ..., d_k\}$. For the i^{th} nearest neighbour, the weight is

$$w_i = \frac{d_k - d_i}{d_k - d_1}$$

The algorithm was built and implemented by us from scratch and no external library was used.

3.2.3 Perceptron Learning Algorithm

To discriminate between the points, we also employed the use of a Single Layer Perceptron. The weights for the Perceptron were intialized randomly and in each iteration of the algorithm, we updated the weights based on the error values. The perceptron algorithm makes a linear discriminant between the two classes of points. This algorithm was implemented from scratch by us without the use of any library. The working of the algorithm begins with weights and biases. We need to initialize them to random values and improving them gets us the best linear separator. After the intialization, the weights are multiplied with the inputs as follows:

$$a = \sum_{i=1}^{n} (w_i x_i) + b$$

where w_i is the weight and the x_i is the corresponding input. After this, we add the bias term, b to the sum. The bias and weights are then updated for each missclassification as follows:-

$$w_{i+1} = w_i + \eta * \text{true-class} * x_i$$

$$b_{i+1} = b_i + \eta * \text{true-class}$$

where x_i is the missclassified point and η is the learning rate. After multiple validation tests, $\eta = 0.01$ was decided.

3.3 Logistic Regression

Logistic regression is also a model for classification, mainly to classify between 2 classes only. This type of classification is used with the help of various functions, like Odd Ration (OR),

Logit function, Sigmoid function or Logistic function, and Cross-entropy or Log Loss. In our project we have used the Sigmoid function which is defined as follows:

$$\phi(x) = \frac{1}{1 + \exp^{-x}}$$

This function is used on the value calculated in the PLA(Perceptron Learning Algorithm), which comes out to be the dot product of trained weights and the features, in order to gain more accuracy. The function is so defined that it ranges only from [0,1]. So the classification is done according to the value calculated by the sigmoid function. If it is greater than or equal to 0.5, it is classified as one class, otherwise it is classified as another.

3.4 Image Annotation

While training, we are using annotated images, however, during the inference phase we will be using an unseen image and then marking landmark points on it. For that, we will be using PoseNet: A TensorFlow based Annotation Tool. The PoseNet model takes a processed camera image as the input and outputs information about keypoints. The keypoints detected are indexed by a part ID, with a confidence score between 0.0 and 1.0. The confidence score indicates the probability that a keypoint exists in that position. This keypoint estimation takes place in two phases:

1. An image is fed through a CNN.
2. A pose-algorithm decodes the poses, confidence scores, positions of various keypoints and the confidence of the positions

We select the points with highest confidence scores to mark our landmark points.

4 Experiments and Results

4.1 Datasets

The dataset that was used for this project was the MPII Human Pose dataset [9]. It is a state of the art benchmark for evaluation of human pose estimation solutions and algorithms. This dataset includes roughly 25,000 images and expresses roughly 410 human activities. It is a labelled dataset. For the scope of our project, we took up two different activities - **walking and cycling**. Our final dataset consisted of 1000 images which were expressed through 7 features.

4.2 Experimental Setup

The experiment was set up in three phases after the processed data was collected. This included sanity checks and making proper splits for each phase to make sure that we were preserving the data sanctity by separating testing and training data.

1. Training

2. Testing

3. Inference

4.3 Training

After the preprocessing of the data, we started training our models. The training data was built using 90-10 splits where 90% of the data was for training and rest was for testing.

We used both an iterative approach and cross-validation for training to see the best k-values. During the training phase, we also applied r-fold cross validation to identify the best k-values. The R-fold cross validation is done by dividing the training data into R number of folds (sets), of equal size which can be used to find the best value of K in a particular range. In these folds, one fold is selected one by one considering it as the testing set for the training set(which is the rest of the folds). The errors are calculated for each validation set, and saved for a particular value of k which is in the range taken before. Now this done for all values in the range, and the average of all the errors are taken which correspond to the same validation set. After these errors are averaged out, we get errors for every value in the range. Once again these multiple errors are averaged out to get a single error for each value in the range. The number corresponding to the minimum error in the total range, is taken as the optimal value of K, after which we can say the training of KNN algorithm is complete.This K value will be used to classify the unseen testing data. The classical KNN model gave better results than the modified KNN contrary to what we had had hypothesized. This could be primarily because of the nature of data. Though, the modified/weighted KNN algorithm was much more consistent when compared to the basic KNN. It is noteworthy that the data affects results for any classifier. The train results for the Nearest Neighbours algorithms can be seen in Figure 5

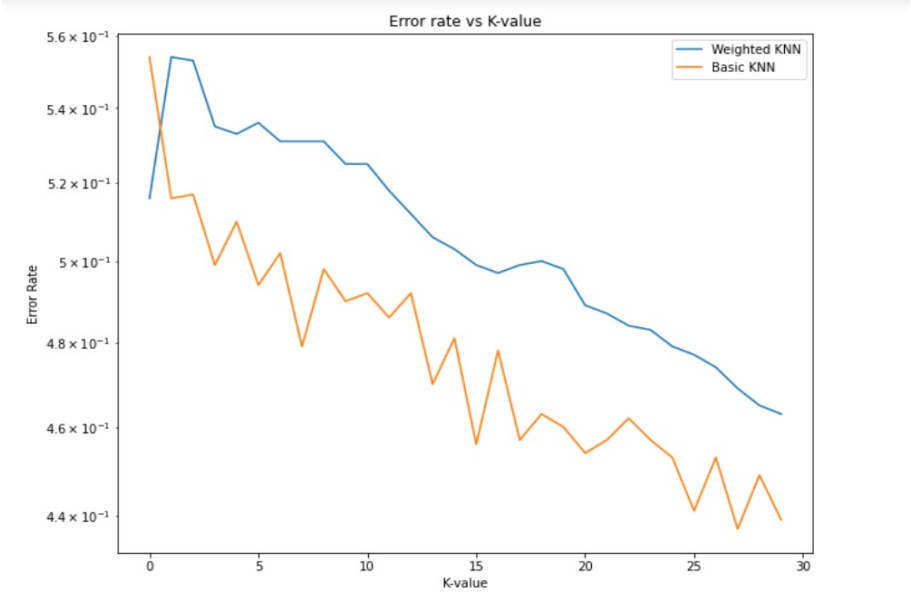


Figure 5: Training Error for the NN algorithms

The training loss and accuracy for the PLA algorithm can be seen in Figure 6 and 8 respectively.

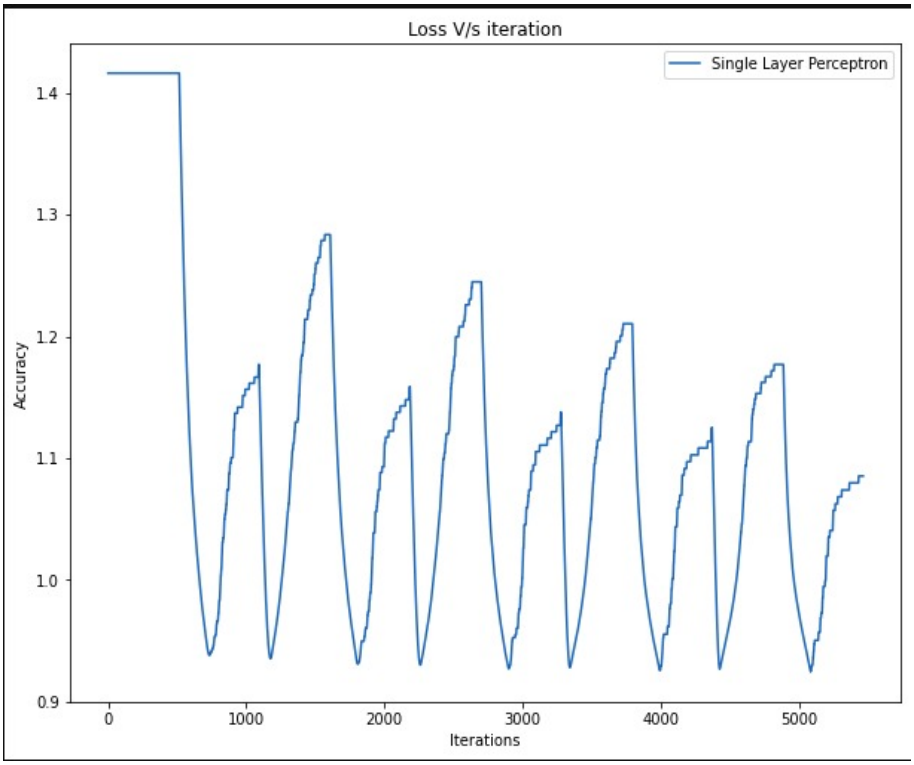


Figure 6: Training Loss for PLA algorithm

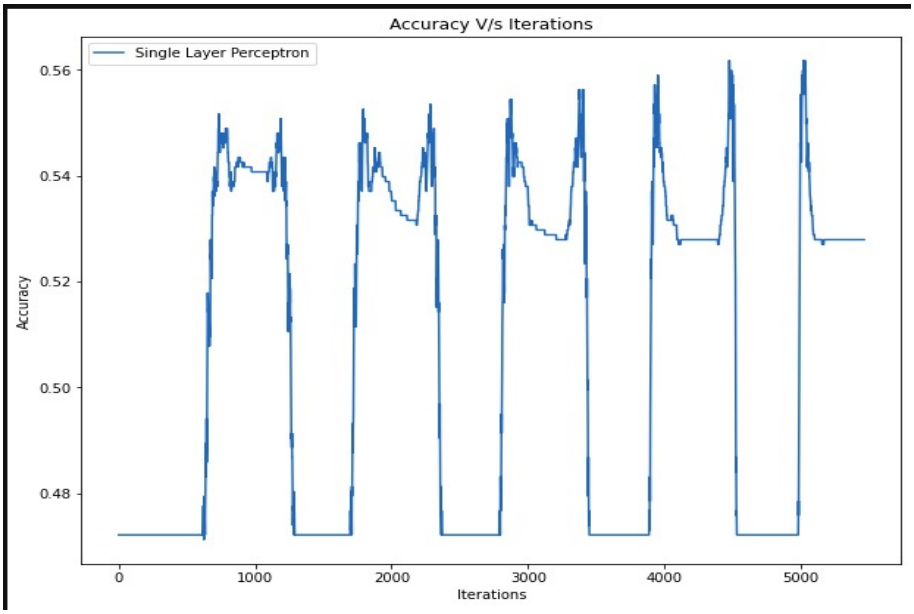


Figure 7: Training Accuracy for PLA algorithm

The training result for logistic regression are as follows:

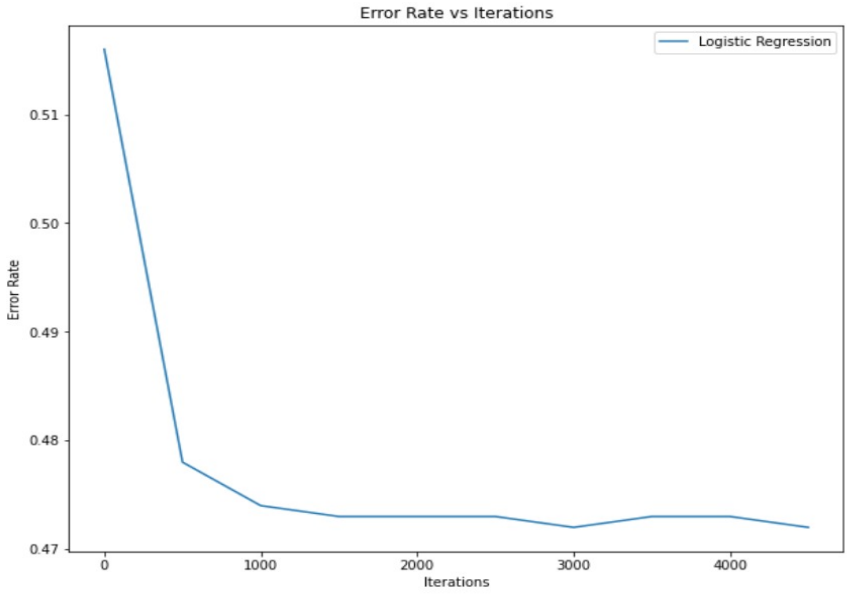


Figure 8: Training Error for Logistic Regression

4.4 Testing

In the testing phase, we used the remaining data and tested for different values in the NN algorithms to see if our output of best K-values conincided with the same here.

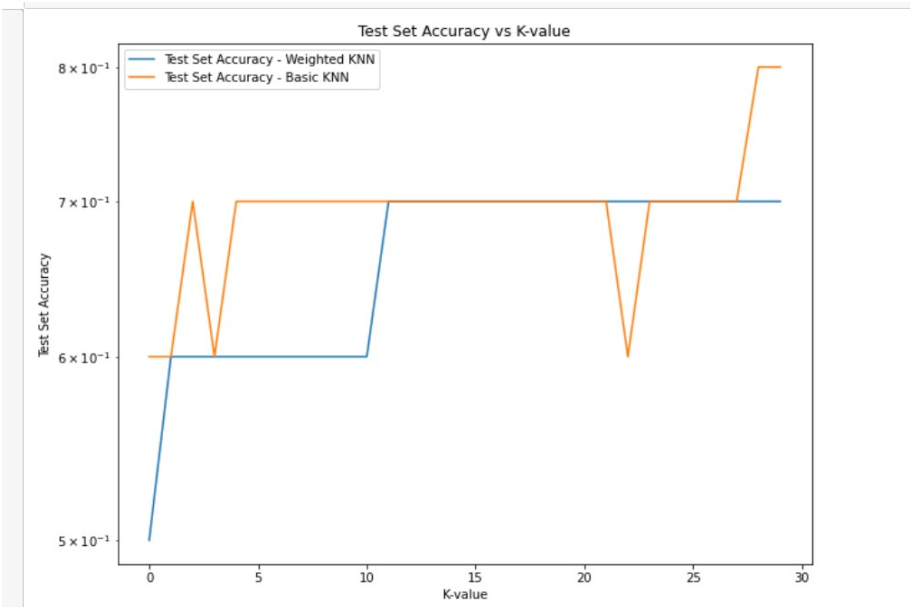


Figure 9: Testing Accuracy for the NN algorithms

4.5 Inference

In this section, we intend to start with an unseen image which is neither annotated nor processed. Our inference setup passes the image through a PoseNet and returns the image annotations. We then have automated processing and testing algorithms implemented. These algorithms extract the features that we want to input in our model's prediction function. The model then predicts the accurate class and returns it. The outcomes of our inference stage can be visualised in Figure 10.

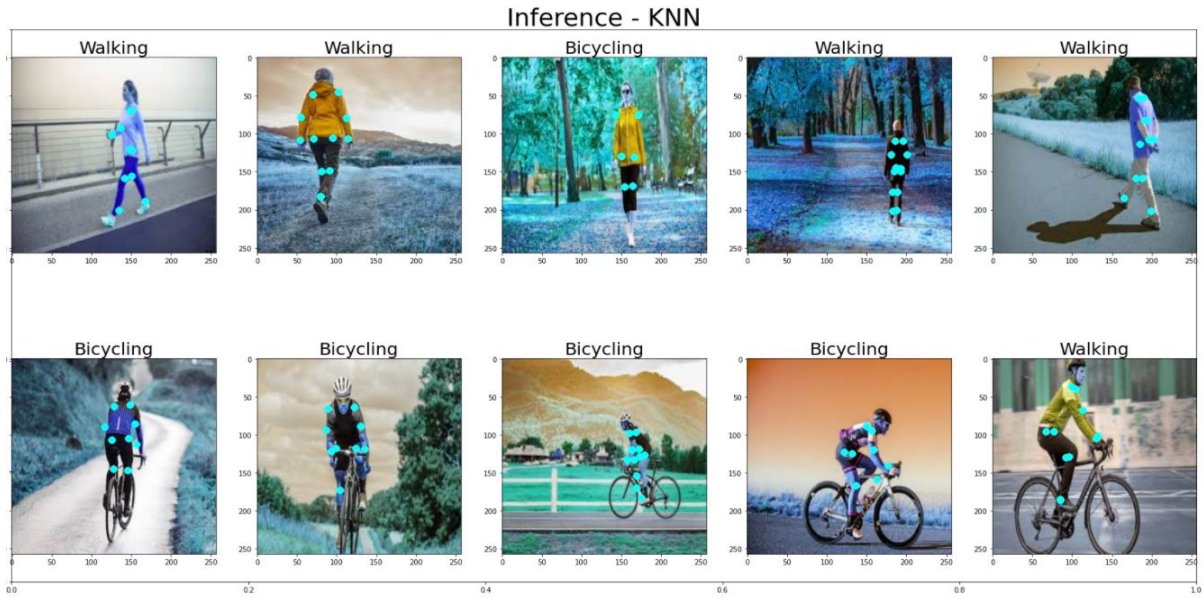


Figure 10: Inference Output for unseen test images using our algorithm (k=20)

5 Conclusions and Future Scopes

It is clear that this problem is a highly challenging one which has been puzzling Computer Vision Scientists for decades. This is mostly due to the challenges we discussed in earlier sections. We have attempted to make a dent in this sphere through the means of this project. We have a taken a subset of activities for this project and classified them using landmark points. The algorithms used were k-Nearest Neighbours, modified k-Nearest Neighbours, Perceptron Learning Algorithm and Logistic Regression. These algorithms were selected on the basis of the problem and keeping in mind that we wanted to separate the class points effectively using white-box models. The models showed an incremental results in accuracy which showed how our model selection was suited to the task. It should be noted that the classical knn performed better at one k-value on the test set but had highly unpredictable results while modified kNN was much more reliable and consistent in it's performance. The results were humbling and show that there is scope for improvement in our project. We have also refrained from using black-box models and have focused on explainability over performance. In the future, we intend to identify better features and classify more activities. We can also introduce powerful deep learning models and perform feature importance techniques

to identify important features in the human pose estimation problem. We could also look at 3D images for better annotations and to overcome the problem we faced in this task. 3D images can represent human poses in a much better format by increasing the concept of depth.

6 Acknowledgement

We would like to thank our professor, Dr. Snehasis Mukherjee for providing us with the opportunity to learn about a challenging problem in the field of Computer vision. We learnt many new concepts and tools to implement this project successfully. His focus on concepts made sure that we actually understood what we were implementing rather than just treating algorithms as a black-box. To complete this project, we had to read about past work, learn from their experiments, identify a field we could work on, design the experiment and then finally implement our learning. This course and it's project has been a holistic learning experience for all of us.

References

[1] Toshev, A., Szegedy, C. (2014). DeepPose: Human Pose Estimation via Deep Neural Networks. 2014 IEEE Conference on Computer Vision and Pattern Recognition, 1653-1660.

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

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

[4] Y. Yang and D. Ramanan. Articulated pose estimation with flexible mixtures-of-parts. CVPR, 2011.

[5] Raj, B. (2019, May 1). An overview of human pose estimation with deep learning. Medium. https://medium.com/beyondminds/an-overview-of-human-pose-estimation-with-deep-learning-d49eb656739b [last accessed: Nov 28, 2020]

[6] G. Gkioxari, P. Arbel´aez, L. Bourdev, and J. Malik. Articulated pose estimation using discriminative armlet classifiers. In CVPR, 2013.

[7] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei and Yaser Sheikh. OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields

[8] S. A. Dudani, "The Distance-Weighted k-Nearest-Neighbor Rule," in IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-6, no. 4, pp. 325-327, April 1976, doi: 10.1109/TSMC.1976.5408784.

[9] Andriluka, M., Pishchulin, L., Gehler, P., Schiele, B. (2014). 2D human pose estimation: New benchmark and state of the art analysis. 2014 IEEE Conference on Computer Vision and Pattern Recognition. https://doi.org/10.1109/cvpr.2014.471

[10] Srivastava, T. (2014, October 10). Introduction to k-nearest neighbors: A powerful machine learning algorithm (with implementation in Python R). Analytics Vidhya. https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/

[11] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

[12] A. Kendall, M. Grimes and R. Cipolla, "PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization," 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, 2015, pp. 2938-2946, doi: 10.1109/ICCV.2015.336.

[13] Google Research. (2015). Pose estimation. TensorFlow. Retrieved December 6, 2020, from https://www.tensorflow.org/lite/models/pose_estimation/overview