Journal of High School Science

Martial art move detection and validation using the K-nearest neighbor algorithm

Mahajan A[1], Jain R[2]

Submitted: May 16, 2022, revised: version 1, August 16, 2022, version 2, September 5, 2022
Accepted: September 24, 2022

Abstract

A live-action algorithm method is proposed that tracks various human movement sequences which are then processed and classified as an action. The action recognition method detects human movements such as sitting, falling, walking, or more specialized sports actions like those associated with karate, running, or jumping. Live-action recognition methods are more complex because they have to analyse and process real-time data streams within a short time span. The proposed real-time live-action algorithm method recognizes and evaluates karate moves using a smart phone camera. The method has multiple advantages such as providing real-time feedback on how accurately a particular move was executed and the capability to self-train. The algorithm relies on pose detection, which is combined with the K-nearest neighbour (KNN) algorithm to estimate pose accuracy. This, along with the repetition counting algorithm, allows for the accurate counting of karate actions. Other use cases include no-contact based training and greater adoption as the algorithm can be run on any smartphone without the need for an instructor. In regions where facilities are unavailable or inaccessible, or restrictions imposed on certain segments of the population, this type of training system can be particularly useful.

Keywords

K nearest neighbour (KNN), Motion tracking, Motion estimation, Karate move detection, Media pose.

---

[1]Corresponding author, Aarush Mahajan, Lotus Valley International School, Expressway, Sector 126, Noida, Uttar Pradesh 201313, India.
aarushmahajan01@gmail.com
[2]Reetu Jain, Mentor, On My Own Technology Pvt. Ltd. 1018/909, B Wing, Samarth Aishwarya, New Link Road, Lokhandwala, Oshiwara, Mumbai 400053, Maharashtra, India.
reetu.jain@onmyowntechnology.com

Introduction

Pose estimation is a computer vision (CV) technique that detects human figures in images and videos so that the relative position of the body parts in the image or video can be determined. Pose estimation only estimates where key body joints are and does not recognize who is in the image (6). The pose estimation is performed by finding the location of key points for the given object's body. Based on these key points, various movements and postures can be compared. Human activity can be classified into three categories:

a. Atomic actions - movements of a person describing a certain motion that may be part of more complex activities.

b. Gestures - primitive movements of the body parts of a person that may correspond to a particular action of that person.

c. Human-to-object or Human-to-human interactions - human activities that involve two or more persons or objects (7).

This work aims at automatically recognizing sequences of complex karate movements and quantitating the quality of the movements performed. Since this is a problem which intrinsically requires a 3D model, in this work a solution is proposed, which takes as input, the sequences of skeletal motions that can be derived from both motion capture hardware and consumer-level, off-the-shelf, depth-sensing systems. The proposed system is constituted of four different modules: skeleton representation, pose classification, temporal alignment, and scoring. The system is tested on a set of different punch, kick and defence karate moves executed, starting from the simplest case, i.e., fixed static stances (Heiko Dachi), up to sequences in which the starting stances are different from the ending one. The dataset was recorded using a single Microsoft Kinect®. The dataset includes the recordings of both male and female athletes with different skill levels, ranging from novices to masters (8).

Action recognition tasks are solved with many popular and efficient classifiers. The choice of classification method is often determined by feature selection which frequently utilises derivatives of body joint positions. This work concentrates mainly on body joint-based features (9).

Literature review

Action recognition from video has been a wide area of research. Provided with the complexity of the task involved, various researchers have used an array of techniques to try and solve this problem. Action recognition using video frame data as a direct input to the model is one of the approaches used. Ramanathan et. al. (1) provided an overview of various video action recognition techniques and listed common problems faced by those techniques. A few of the major problems include variation in viewpoint, in which the person for whom the action is being recognised can be standing in any orientation relative to the camera. The execution rate is the next problem. The same action can be performed at different rates which can cause the trained model to be unable to predict the action accurately. Camera motion, cluttered background and anthropometric variations are some other problems faced by video action recognition techniques. Zhu et. al. (2) provided a comprehensive review of all the video action recognition techniques in chronological order. They also highlighted the set of available data sets for action recognition. Ullah et. al. (3)

proposed a recurrent neural network (RNN) and a Long-Short term memory (LSTM) based method to train the model. They reduced the model complexity by sampling every video's 6^{th} frame. This data was then input into a convolutional neural network (CNN) to extract the deep features. Then the sequential information was input into a densely connected - bidirectional LSTM to increase the depth of the model. The authors reported that the model was capable of learning long-term sequences and processing lengthy videos. Majd et al. (4) proposed a method in which the video features were handcrafted. They worked with an extended version of LSTM called C2LSTM which allowed the motion data perception to be as good as Spatial-temporal features. They compared the model to two well-known benchmarks, UCF101 (an action recognition data set of 13320 realistic action videos, collected from YouTube, having 101 action categories, at https://www.crcv.ucf.edu/data/UCF101.php) and HMDB51 (a human motion database at https://serre-lab.clps.brown.edu/resource/hmdb-a-large-human-motion-database/)

A number of machine learning algorithms have been presented for classification of human activity. These predominantly use IMU sensors. Chen et. al. (10) introduced a random forest (RF) algorithm to classify human activities. The algorithm used an accelerometer with a gyroscope to capture the data. It was applied on a dataset consisting of 27,681 samples and reached a total accuracy of 83.59%. Sangeetha et. al. (11), have discussed the idea of using a Neighborhood component in the analysis of human activity. They used two different algorithms, random forest and KNN, for every movement. They collected 47,000 element vectors for the accelerometer and the gyroscope, the KNN with a K = 23 had a movement detection precision of 78% and it reduced the K – value. The random forest model resulted in a precision of 75.85% using 23 trees in the RF classifier. Mohsen et. al. (12), discussed the application of the KNN algorithm for human activity tracking using the acceleration and gyroscope. Their results indicated that the KNN algorithm could classify human activities with a high degree of precision. Their weighted average precision, recall, F1- score, and the area under the micro-average precision-recall curve for the KNN were 90.96%, 90.46%, 90.37%, and 96.5%, respectively, while the area under the receptor-operator curve (ROC) was 100%. Ahmed et. al. (13) presented results on the classification of human activity based on Angle Variance analysis. They used a Poincare Plot, that resulted in 88% accuracy, compared with a 58% accuracy using the count-based approach and a support vector machine (SVM) classifier with 10-fold cross-validation. In the classification of the variants of dynamic activities with the MHealth dataset (body motion and vital signs recordings for ten volunteers while performing several physical activities at http://archive.ics.uci.edu/ml/datasets/mhealth+dataset), the accuracy for angle variance-based and count-based methods was 100%, in both cases, using a fivefold cross-validation with SVM classifiers.

Current methods of video action recognition involve using pose detection first on the video data and then training the model on the poses from each video. This method is preferable as the amount of data that is used for training is reduced significantly, allowing for less computational load and short training times. Ge

et. al. (5) proposed a convolutional LSTM-based method whose approach was to use Google Net to obtain the pose-based features first, then perform pre-processing on the pose features with subsequent input into an LSTM to learn the sequence-based features. They adopted temporal coherence to reduce redundant features and training time. A custom spatial transformer was used for attention.

Hachaj et. al. (6) considered the problem of joint segmentation and classification of various common-live actions in the framework of conditional random field (CRF) models. Model training was performed by means of an efficient likelihood maximisation algorithm, and inference was based on the familiar Viterbi algorithm. Yang et. al. (7) proposed a method to recognize human actions using 3D skeleton joints recovered from 3D depth data of RGBD cameras. The authors designed an action feature descriptor for action recognition based on differences in skeleton joints, i.e., Eigen Joints which combined action information including static posture, motion property, and overall dynamics. Accumulated Motion Energy (AME) was then proposed to perform informative frame selection, which was able to remove noisy frames and reduce computational cost. Non-parametric Naive-Bayes-Nearest-Neighbour (NBNN) was used to classify multiple actions.

Methods

Human pose estimation is a computer vision technique to track the movements of a person or an object. This is usually performed by finding the location of key points for the given objects. Based on these key points, various movements and postures can be compared. Pose estimation is actively used in the field of augmented reality, animation, gaming, and robotics. Some of the methods for pose estimation are Open pose, Pose net, Blaze pose, Deep pose, Dense pose and Deep cut.

Blaze Pose GHUM 3 was selected for the real-time video analysis and landmark identification. This pose detection method is provided by the Media Pipe library. Media Pipe is an open-source cross-platform framework (https://google.github.io/mediapipe/) for building multimodal machine learning pipelines. It can be used to implement cutting-edge models like human face detection, multi-hand tracking, hair segmentation, object detection and tracking. Figure 1 shows the flowchart of the algorithm used in the study.

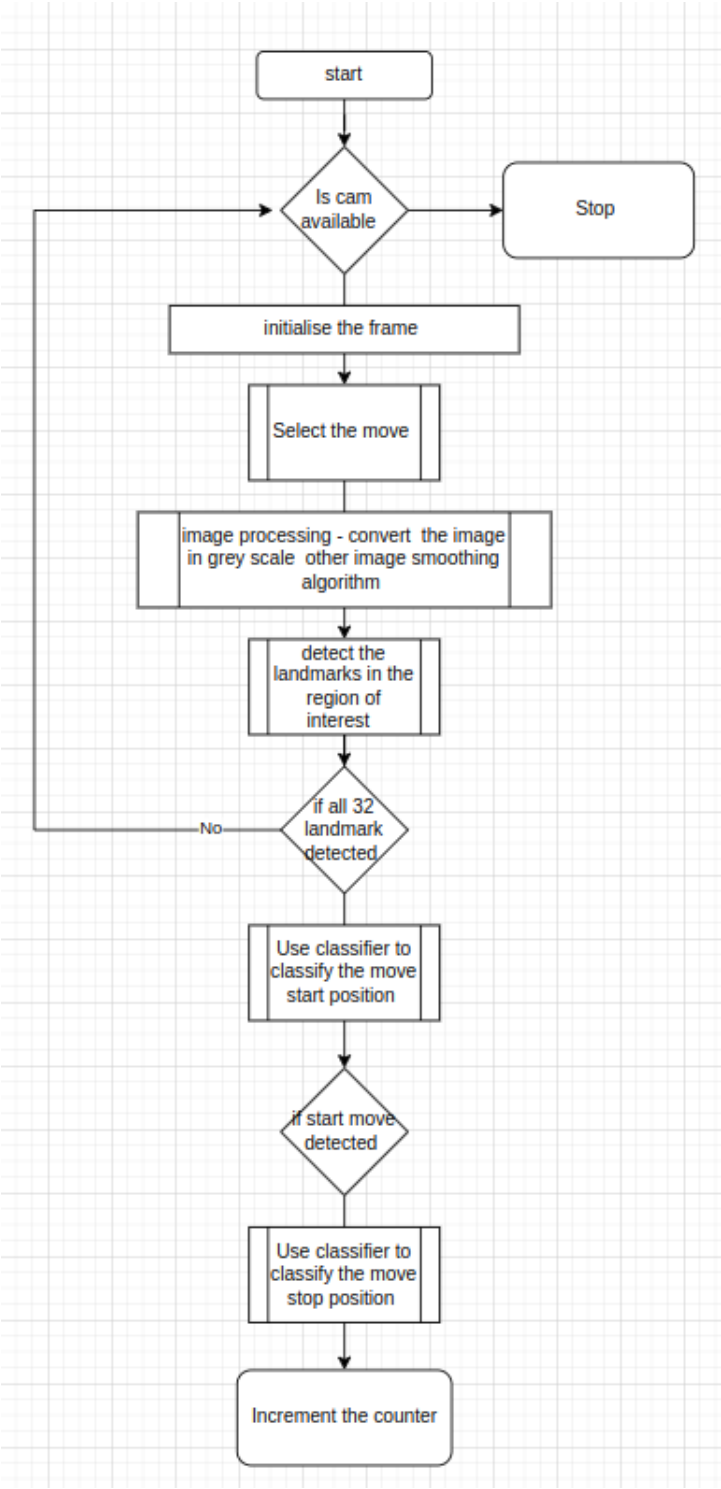

Figure 1: Flowchart/ Algorithm used in the study

The data output from the Media Pipe is 33 points constituting the various important landmarks of the human body, in the form of an array where each row has the x, y and depth data of each landmark. The landmarks in themselves are of no use as the x and y data is relative to the video resolution being used. Hence, the data was converted to a more meaningful data representation format by first calculating a set of distances between the important landmarks. The landmarks across which the distances needed to be calculated were decided after considering the actions being performed and those sequences which would classify the actions accurately. Figure 2 shows the different body landmark distances that were used.

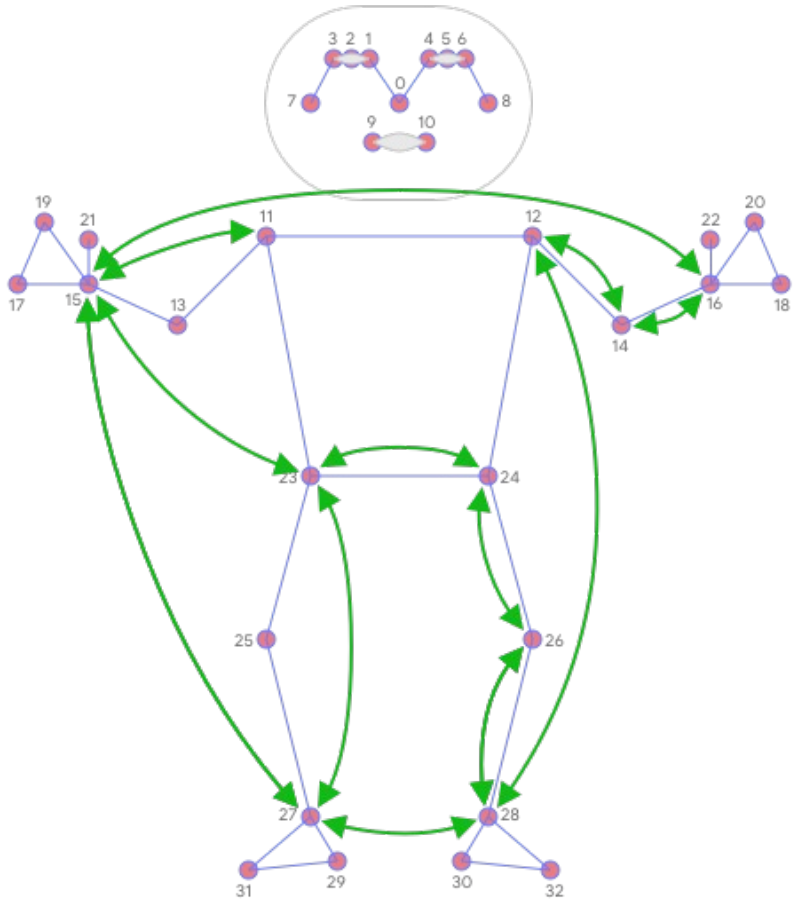

Figure 2: Distance metric for various body landmarks

The second step was to normalise the data. We used the torso distance to normalise the data. The normalisation step is important because different persons have different body sizes and can be standing at different distances from the camera. This normalisation step allowed us to remove those biases. The third step was to use the distance data to train a supervised learning

KNN (k nearest neighbour) model. Provided a set of pre-known data points which represent different classes, the KNN algorithm allowed us to classify a data point as one of the previously defined classes. For visualization, we used 2D data, but KNN is equally applicable to multidimensional data as well. Figure 3 shows a plot of data belonging to 2 different classes. This encompasses the training dataset for the model.

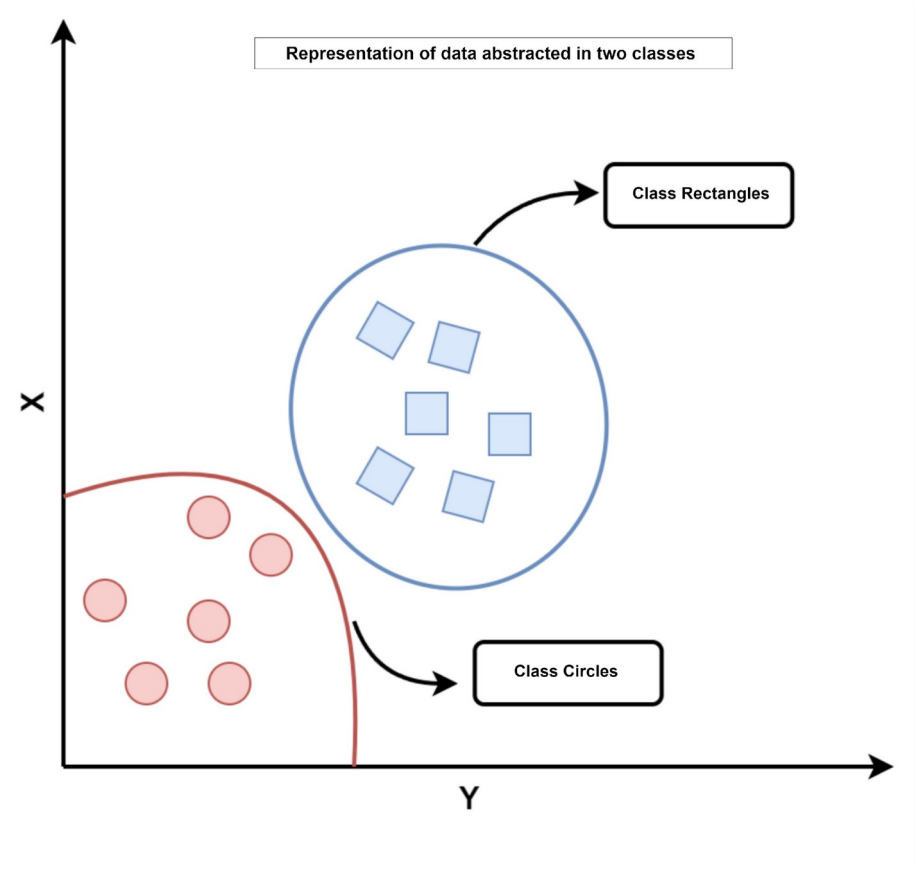

Figure 3: KNN classification

Now if we introduce new data, the model will predict the class it belongs to. First, the distance between the new data point and all the known data points was computed by using the Euclidean distance formula. For the KNN algorithm to work we needed to define the value of K which determines how many closest data points should be considered. Hence, once the distance was calculated, we selected the first K data points which have the shortest distance to the given data point. Then the number of data points belonging to different classes were counted and the class with the highest number of data points was assigned to the new data point. Figure 4 shows how the K values were used to select the first 3 data points which have the shortest distance to the new data point. Figure 5 shows that the Circle class is assigned to the new data point because there are more circles than rectangles in the given K range.

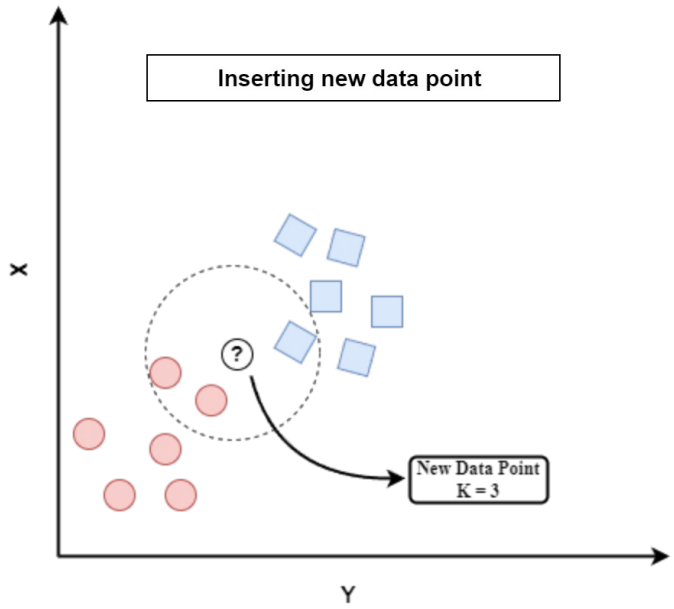

Figure 4: KNN New data point classification

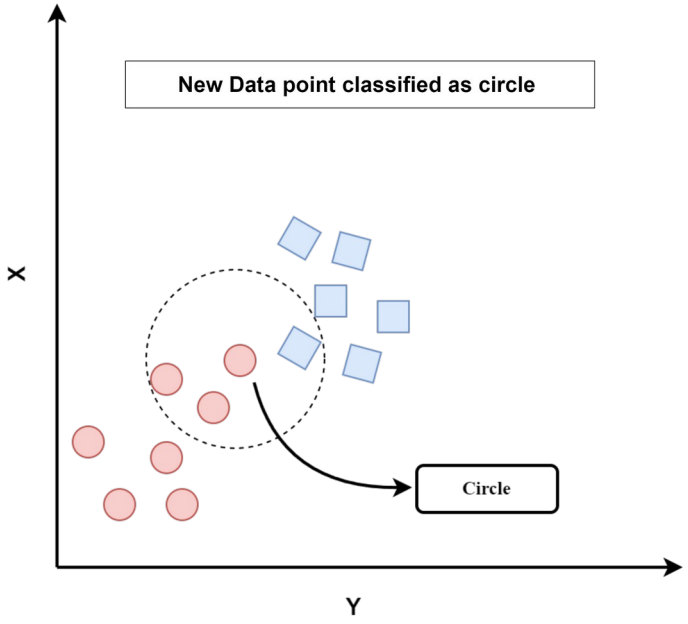

Figure 5: KNN New data point classification

We calculated 23 distances using the 33-landmark data from the Media Pipe library. These distances were then normalised to remove bias. Consider one action being performed - a left-hand punch. We created 2 classes using the above-mentioned method, one representing the punch start and the other representing the punch stop. The data was in the form of camera frames that were used to generate the above-mentioned normalised data for both classes. This completed the model training process. Subsequently, when a new frame was input, it was first run through the Media Pipe library to generate the 33 landmarks. Then it was pre-processed and normalized to generate the 23-distance metric. Since solving a problem with 23 data points is slow and inefficient using the KNN algorithm, the data points were reduced to one variable. This was achieved by subtracting each distance metric from the corresponding distance metric in the training dataset. The absolute value was used to calculate the distance of the new data point from all the other data points. A K value of 30 was used, implying that the classes of the data points in the first 30 points with the least distance were selected, and the class with the maximum count was assigned to the new data point.

Counting repetitions

The model outputs the confidence of prediction based on the number of classes that exist in the 30 samples. As shown in Figure 6, a high and low threshold was set for each action pose.

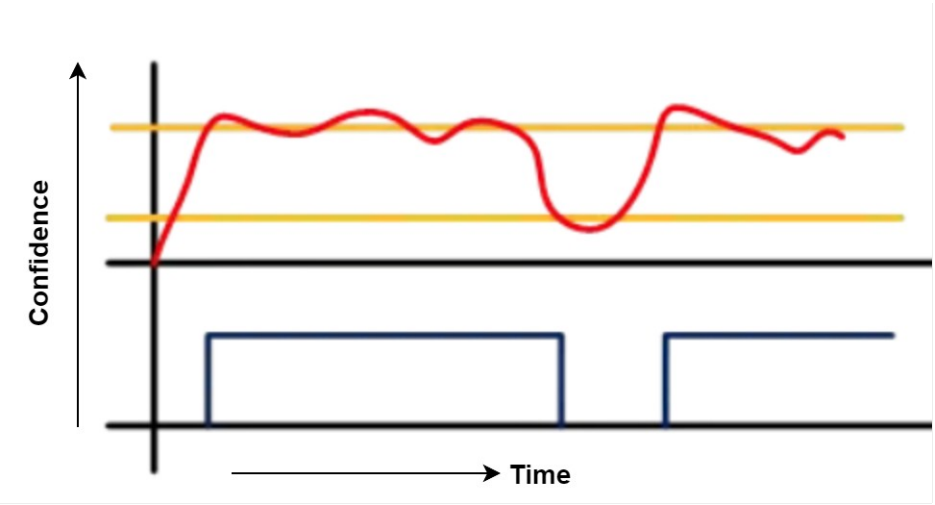

Figure 6: Start and Stop phase thresholds for Punching

Every action was divided into a start and a stop phase. The action could exist only in one of the given phases and was assigned to every phase. A punch's start phase was entered when the action pose confidence output a value above a given threshold, but this did not increment the counter. The counter was only incremented when the stop action phase was input. The thresholds were selected and calibrated for each individual action separately. The model

was only able to predict and count the pre-selected action being performed. The graph displayed is the model's prediction confidence for a particular phase of the action. The count being triggered was visually displayed on the graph. The graph also shows the confidence level of the model when a smoothing function is applied to the input data.

Data Augmentation

Data sets containing karate actions are not readily available. Those videos that are available are not adequately labelled. Therefore, we created our own data set using a qualified professional. The videos of different moves were recorded when the person was facing the camera. However, since the video does not account for all the possible ways of camera orientation, we decided to augment the data by using techniques like scaling, rotation, and perspective transformation. Data augmentation not only increased the amount of data available for training but also increased the accuracy of the model. Data augmentation was a necessary experimental design add-on in our case since the video data was not used directly to train the model. Instead, the data was input through the Media Pipe library which in turn converted the video into 33 landmarks. Hence, applying the transforms to the video caused a significant improvement in the quality of the data. Figure 7 is a representation of transforms that were applied to the video.

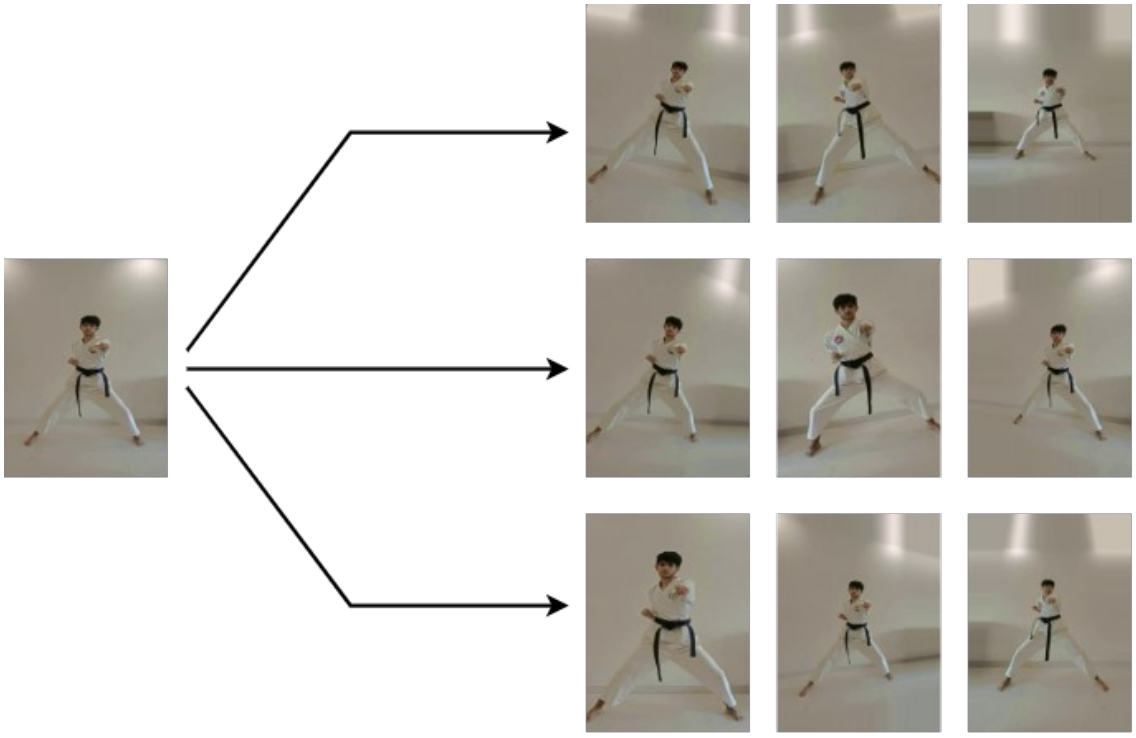

Figure 7: Data augmentation

Data augmentation was performed for every frame in the video. Since the data was being used for posing classification, we used the data augmentation step for the first 3 and the last 3

frames. This allowed us to capture the images from a different perspective. This emulated the different possible camera placements as well as the different body aspect ratios during actual implementation. This is important as the entire training set was made using a single individual.

Results.

Table 1 shows the performance parameters for the model trained on the various moves. The double punch and dynamic block were predicted with 100% accuracy, whereas the lefthand block and straight kick had accuracies of 99% and 81% respectively.

Table 1. Performance parameters of the KNN model for different moves

Move	Accuracy	Precision	Recall	AUC	F1	True negative	False positive	False negative	True positive
Double Punch	1	1	1	1	1	125	0	0	130
Dynamic block	1	1	1	1	1	126	0	0	129
Left hand block	0.99	1	0.99	1	1	131	0	1	123
straight kick	0.81	0.80	0.89	0.89	0.83	71	28	14	114

As stated above we trained the model for four different moves, the data set was created using the data augmentation and trained using the above algorithm, the below chart shows the data point for the double punch start and stops for each augmented data. The algorithm can detect double punches with an accuracy of 100%, the algorithm can work on the recorded video as well as the live video for estimation purposes. The designed algorithm is not symmetric when estimating the pose, hence if the right leg kick is selected, the algorithm will not detect a kick-starting with the left leg and ending with the left leg. Similarly, the double punch; when started with a right hand and ended with the left hand, cannot be detected.

Figures 8 and 9 show the ROC curve and the confusion matrix for the dynamic block, figures 10 and 11 are those for the left-hand block, and figures 12 and 13 are those for the straight kick. Figures 14 and 15 are the actual normalized length data for different images used for the training data set.

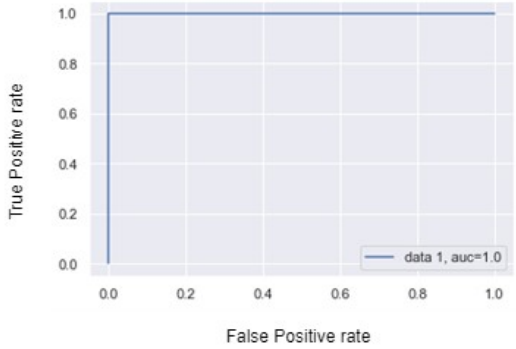

Figure 8: ROC curve for dynamic block

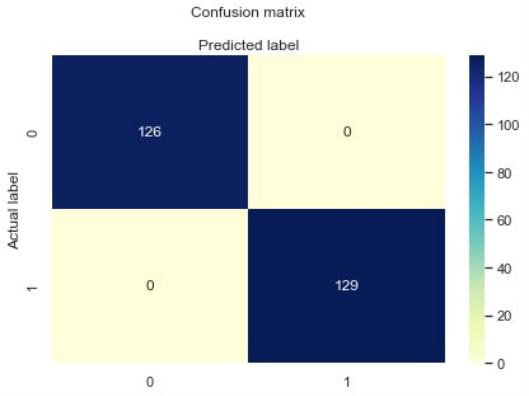

Figure 9: Confusion Matrix for dynamic block

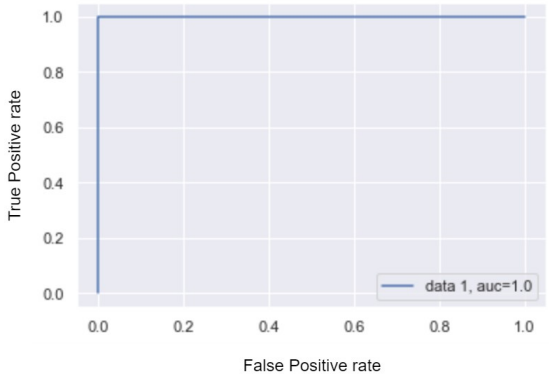

Figure 10: Roc Curve for the left-hand Block

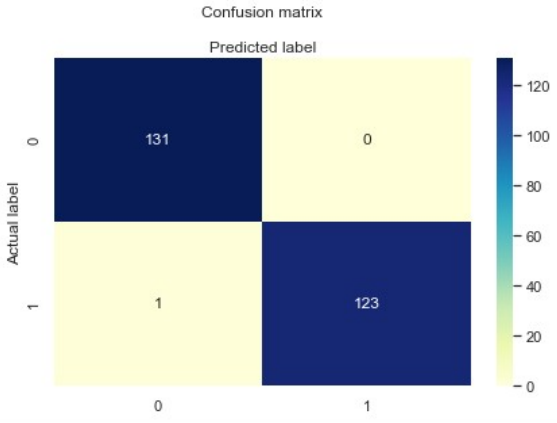

Figure 11: Confusion Matrix for the left-hand Block

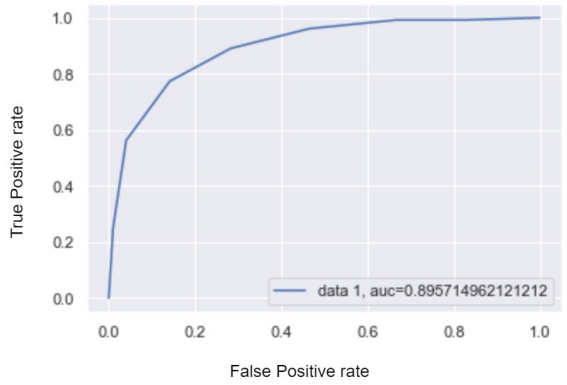

Figure 12: Roc Curve for Straight Kick

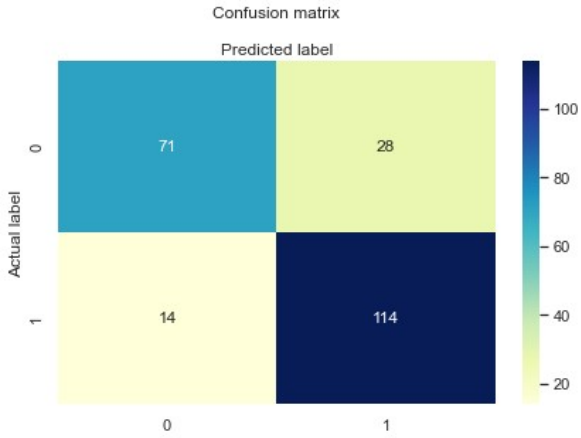

Figure 13: Confusion Matrix for Straight Kick

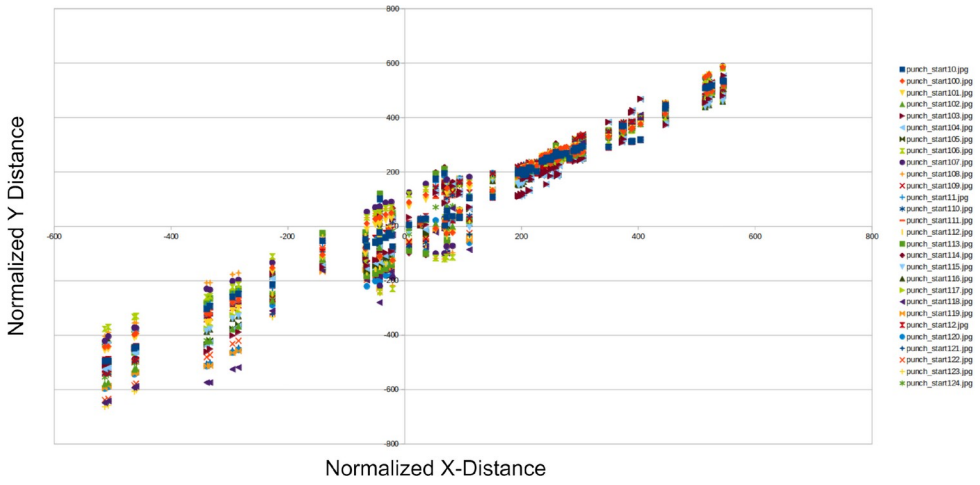

Figure 14: Actual normalized length data for different moves in the training data set

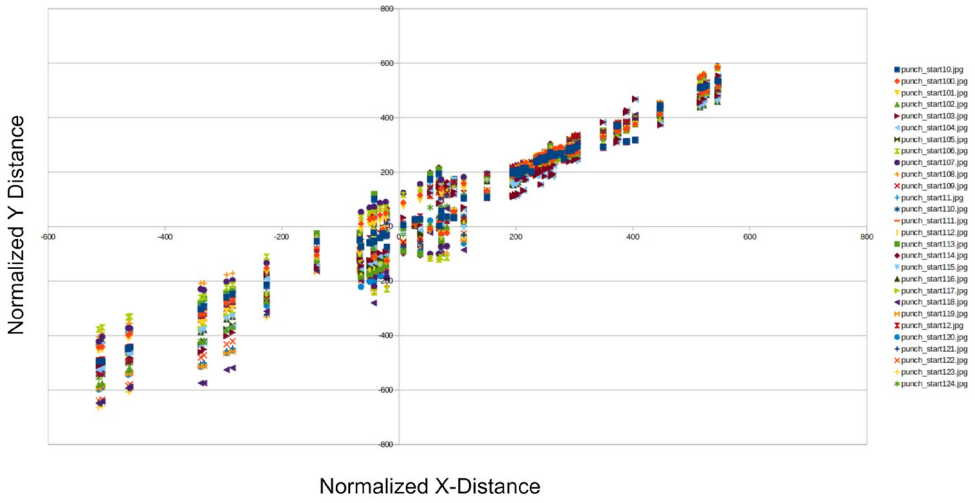

Figure 15: Actual normalized length data for different moves in the training data set

Figures 16 and 17 show the result of how the algorithm detects and counts the punching action. The embedded graph shows the confidence of the model regarding the current pose. Since the model was trained to detect the punching action, the model predicts the punch start and the punch stop action. The Y axis represents the confidence score of the model regarding the two labels, punch start, and punch stop. When the model is confident that the pose is a punch start position, the graph registers a confidence score of 1. Conversely, a confidence score of 0 indicates that the person is in the punch-stop position. Using the confidence score, the algorithm counts the action repetitions accurately.

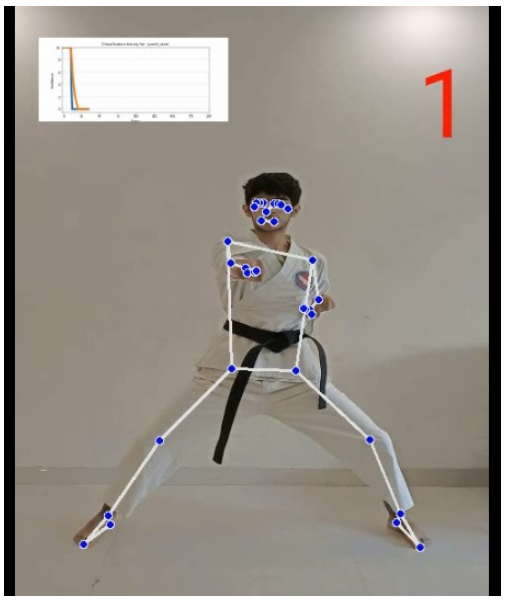

Figure 16: Double punch start detection

The blue line of the graph represents the confidence score of the smoothed pose detection. Pose smoothing is important, as without it there would be a lot of jitters in the input pose which might cause the repetition counting to not function accurately. Hence, the repetition count is designed to process the smoothed poses.

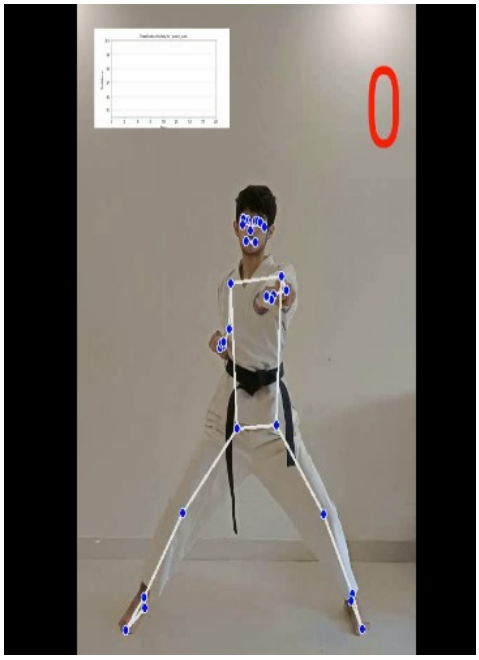

Figure 17: Double punch stop detection

Discussion

The KNN algorithm can detect and track selected karate moves with 98% accuracy and a ROC of 100%. The algorithm can potentially be used to monitor and track other actions such as those associated with yoga, golf, training at the gym and other fitness moves. For dynamic moves in yoga, it is necessary to first recognize individual pose checkpoints to identify and then validate the moves, rather than only monitoring the start pose and the end pose.

The proposed algorithm only detects the move from one start point to another, and therefore cannot work for symmetric moves. For example, in the example of the double punch move, this punch can be executed by starting with the right hand and ending with the left or vice versa. A separate algorithm is required for the latter executed move. If the algorithm can be designed to filter and identify the start and end move poses, it can be used for symmetric moves as well.

Conclusion

The KNN model was used to predict various karate moves at K=10 and K = 30. The model was found to be 100% accurate for the double punch and single-hand block, 99% accurate for the left-hand block and 81% accurate for the straight kick.

This model can be implemented on a web-based interface where any user can access the

model and practice the karate moves. The video is downloadable and includes a graph and count for further user evaluation. The accuracy of the model can potentially be increased by increasing the amount of training set data. Training data from multiple camera angles can be input thereby potentially increasing the accuracy even when the person performing the action is not facing the camera.

Acknowledgement

We would like to express heartfelt gratitude to all the tutors and mentors of On My Own Technology Pvt. Ltd. for extending their help in carrying out this project.

Data

The data used to train the double punch model can be found at https://github.com/aarushmahajan01/karate/tree/main/Data

References

1. Ramanathan, M., Yau, W.Y. and Teoh, E.K., "Human action recognition with video data: research and evaluation challenges". *IEEE Transactions on Human-Machine Systems*, *44*(5), pp.650-663, 2014.

2. Zhu, Y., Li, X., Liu, C., Zolfaghari, M., Xiong, Y., Wu, C., Zhang, Z., Tighe, J., Manmatha, R. and Li, M., "A comprehensive study of deep video action recognition". *arXiv preprint arXiv:2012.06567*, 2020. https://doi.org/10.48550/arXiv.2012.06567

3. Ullah, A., Ahmad, J., Muhammad, K., Sajjad, M. and Baik, S.W., "Action recognition in video sequences using deep bi-directional LSTM with CNN features". *IEEE access*, *6*, pp.1155-1166, 2017.

4. Majd, M. and Safabakhsh, R., "Correlational convolutional LSTM for human action recognition". *Neurocomputing*, *396*, pp.224-229, 2020.

5. Ge, H., Yan, Z., Yu, W. and Sun, L., "An attention mechanism based convolutional LSTM network for video action recognition". *Multimedia Tools and Applications*, *78*(14), pp.20533-20556, 2019.

6. Hachaj, T., Ogiela, M.R. and Koptyra, K., "Application of assistive computer vision methods to Oyama karate techniques recognition". *Symmetry*, *7*(4), pp.1670-1698, 2015.

7. Yang, Y., Saleemi, I. and Shah, M., "Discovering motion primitives for unsupervised grouping and one-shot learning of human actions, gestures, and expressions". *IEEE transactions on pattern analysis and machine intelligence*, *35*(7), pp.1635-1648, 2012.

8. Bianco, S. and Tisato, F., "March. Karate moves recognition from skeletal motion". In *Three-Dimensional Image Processing (3DIP) and Applications,* Vol. 8650, pp. 154-163, SPIE, 2013.

9. Zhu, Y., Chen, W. and Guo, G., “Fusing multiple features for depth-based action recognition”. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *6*(2), pp.1-20, 2015.

10. Chen, Y. and Shen, C., “Performance analysis of smartphone-sensor behavior for human activity recognition”. *IEEE Access*, *5*, pp.3095-3110, 2017.

11. Hashim, B.M. and Amutha, R., “Machine Learning-based Human Activity Recognition using Neighbourhood Component Analysis”. In *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*, IEEE, pp. 1080-1084, 2021, April

12. Mohsen, S., Elkaseer, A. and Scholz, S.G., “Human activity recognition using K-nearest neighbor machine learning algorithm”. In *Proceedings of the International Conference on Sustainable Design and Manufacturing*, Springer, Singapore, pp. 304-313, September, 2021.

13. Ahmed, S., Bhuiyan, T.A., Kishi, T., Nii, M. and Kobashi, S., 2021. Human Activity Classification Based on Angle Variance Analysis Utilizing the Poincare Plot. *Applied Sciences*, *11*(16), p.7230, 2021. https://doi.org/10.3390/app11167230