Mini Project Report

on

Computer Vision & ML in Posture Detection

Submitted by

20BCS006 - Adithya Hegde

20BCS045 - Esha

20BCS081 - Madineni Rohith

20BCS116 - Samuel Mathew

Under the guidance of

Dr. Prabhu Prasad B M

Asst. Prof., Dept. of Computer Science & Engineering

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY DHARWAD

08/05/2023

Contents

List of Figures

1 Introduction

In the world of basketball, free throws are crucial moments in a game where a player has the opportunity to score points for their team without interference from the opposing team. Therefore, the ability to consistently make free throws can be a significant advantage for a player and their team. However, making free throws consistently requires proper technique and form. Poor posture during a free throw can result in missed shots and lost points.

In recent years, machine learning techniques have been applied to various sports-related tasks, including video analysis of player movement and performance. In this project, we focus on using machine learning to predict whether a basketball player's posture during a free throw is good or bad based on video data.

We collected video data of basketball players performing free throws and annotated them with labels indicating whether the posture was good or bad. Using this labeled data, we trained and evaluated several machine learning and deep learning models, including Long-Short Term Memory (LSTM), Neural Networks and Shapelets, to predict the posture of a player during a free throw.

Our goal in this research is to demonstrate the feasibility of using machine learning to automatically evaluate the posture of a basketball player during a free throw. This project could have significant implications for coaching and training basketball players, as it could provide real-time feedback to players on their form during free throws and help improve their overall performance.

2 Related Work

In the paper "Key Kinematic Components for Optimal Basketball Free Throw Shooting Performance" by Dimitrije Cabarkapa, Andrew C. FryA, Kevin M. Carlson, John P. Poggio, Michael A. Deane et al. the authors recruited 11 experienced basketball players (7 males and 4 females) to participate in the study. The research team used a motion capture system to gather kinematic data during free throw shooting, and analyzed it statistically to determine the key components that contribute to optimal performance. Their study makes several contributions, including identifying these key components and demonstrating their importance through performance testing. The study concludes that coaching cues that focus on proper preparation and completion of the free throw shooting motion can significantly affect the outcome of the shot and differentiate between proficient and non-proficient free throw shooters. Proficient shooters displayed greater knee and elbow flexion, lower relative elbow height, and smaller forearm angle values relative to vertical.[1]

The paper "Sitting Posture Recognition Based on OpenPose" by Kehan Chen et al. constructed a dataset of sitting postures by cutting images to a size of 60x60 pixels, labeling correct and incorrect postures, applying data enhancement, and normalizing the images. They used a CNN model with 19 layers, to train the dataset and stored the results. OpenCV and OpenPose were used to extract posture features and a CNN was built with Keras to train the data set and achieve an accuracy of up to 90% on test and verification sets after 100 epochs of training. This system successfully recognized correct and incorrect sitting postures from real-time video using OpenPose, even when some joint information was missing or occluded. But the quality of the images used in training the model can greatly affect the accuracy of the model. If the images are low quality or contain a lot of noise, the model may not be able to learn the correct features, leading to lower accuracy.[2]

The paper "A Basketball Training Posture Monitoring Algorithm Based on Machine Learning and Artificial Intelligence" by Shihao Hou, Bizhen Lian, Wenhao Li and Hong Tang et al. talks about how human gesture recognition requires classification training. Different classification algorithms are evaluated during the study to select the optimal classifier, and four common algorithms are described in the article - decision tree, naïve bayes, SVM, ANN. The authors

analyze different classifiers, compare the performance of different body movements' data classification, and establish a suitable classification algorithm for training, concluding that the proposed method has practical significance in recognizing basketball gestures. The study on basketball position recognition method based on action division is limited by the lack of data to check the accuracy of the algorithms used.[3]

The paper "Sports Analytics With Computer Vision" by Colby T. Jeffries et al. constructs a program which uses *OpenCV* and *OpenPose* to automatically record and analyze a basketball player's body pose during a free throw. The program starts by generating a reference pose and then checks every frame against it to determine if the player is about to shoot. If a shot is detected, the program records 90 frames and extracts the angles between the shooter's body parts, which are saved to a CSV file for analysis. The program prompts the user to determine if the shot was made. In this study, the author tried to detect small differences in shot forms between successful and unsuccessful free throws by analyzing various angle measures. However, it was found that successful shots did not show many similarities, while unsuccessful shots were more clustered. The results of the shot estimation system did not provide any conclusive evidence regarding the relationship between shot form and outcome due to poor data quality. The authors suggest that obtaining better data, either in terms of quantity or quality, or by exploring different types of data, such as ball trajectory, could help provide more meaningful insights.[4]

The paper "Time Series Shapelets: A New Primitive for Data Mining" by Lexiang Ye and Eamonn Keogh et al. the authors propose a method for extracting shapelets from a set of time series. Shapelets are short, representative subseries that capture the most discriminative patterns in the data. They use a brute-force algorithm to find shapelets by comparing each candidate shapelet to every subsequence in the data. They can be used for feature extraction and classification of time series data. They also proposed a distance measure called the "distance transform" that captures the similarity between a time series and a shapelet. The distance transform considers the minimum distance between any point in the time series and the shapelet. They have shown with extensive experiments that we can find the shapelets efficiently, and that they can provide accurate, interpretable and fast classification decisions in a wide variety of domains.[5]

3 Data Collection and Preprocessing

3.1 Data

We collected video data manually from two groups of basketball players - professional players and inexperienced ones, with the help of a smartphone. We amassed 71 videos from the professional players and 55 videos from the inexperienced ones, with an average length of 2 seconds that shot the start of a free throw till the end of the ball's trajectory through or bouncing off the net. With the clips being recorded at 30 FPS, we got an average of 60 frames per video. We then labelled each of these videos as having good or bad posture, depending on the free throw taker, and proceeded with data extraction.

3.2 Data Extraction

Based on our literature review, we determined the angles that were integral for optimal free throw shooting performance.[] The first angle is the knee angle, which is the internal angle made by bending the knees during a shot. The second is the hip angle, which is the angle formed by two line segments that connect three key joints - the shoulder, hip, and knee. The third angle is the shoulder angle, which is the angle between the long axis of the arm segment and an imaginary vertical line that passes right through the center of the glenohumeral joint at the time point of the ball release. The fourth angle is the elbow angle, which is the angle made between the shoulder, elbow and wrist joints during the free throw. *Figure 1* pictorially shows how these angles have been measured.

3.3 OpenCV and MediaPipe

OpenCV (Open Source Computer Vision) is an open-source library for computer vision and machine learning tasks. It provides a wide range of functions for image and video processing, feature detection, object recognition, deep learning, and more. We used OpenCV to access our system's webcam and take the live video-feed, as well as capture frames from a video on the disk.

MediaPipe is also an open-source framework for building real-time computer vision and

Figure 1. Representation of measured angles: (a) knee angle (b) hip angle (c) shoulder angle (d) elbow angle

machine learning applications. One of the most popular use cases of MediaPipe is for capturing body posture from video data. The framework provides a pre-trained pose estimation model that can accurately identify the position and orientation of key joints in the human body, such as the shoulder, elbow, hip, and knee joints.

To use MediaPipe for capturing body posture in Python, you can install the MediaPipe package and import the Pose module. Then, you can use the Pose module to create a Pose object and pass in the video data as input. The framework will automatically detect and track the position of the key joints in real-time, and return the results as a set of landmarks and their corresponding 2D coordinates. You can then use these landmarks and coordinates to compute various metrics related to body posture, such as the angle between the shoulder-hip and hip-knee joints, the distance between the feet, or the alignment of the torso. These metrics can be used as features for machine learning models that can classify body posture as good or bad.

We made use of these two libraries to extract data from each frame of the videos and stored the same in a *Pandas Dataframe*.

3.4 Data Preprocessing

We analyzed the time series data collected in the previous step for each individual player by the use of graphical methods and plotted the same using the *Matplotlib* library for *Python*. This led to the observation that the data was misaligned along the time axis for the different angles that we had captured. The same can be observed in *Figure 2*.

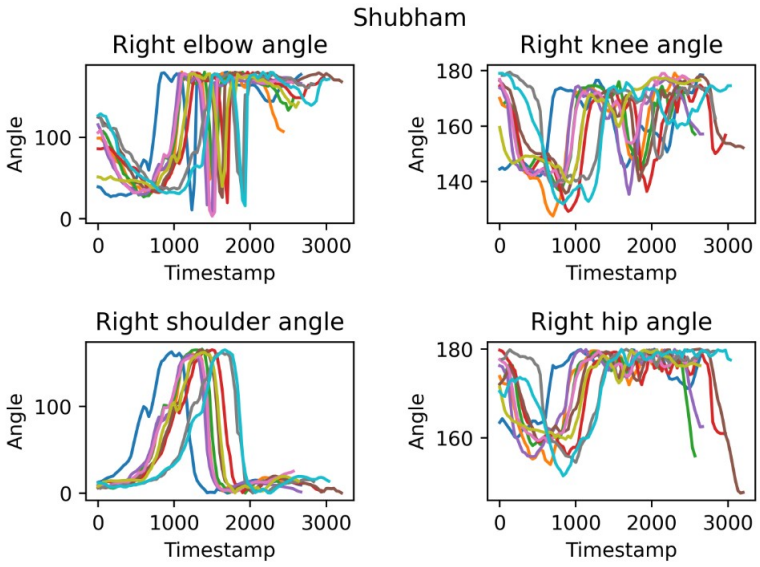

Figure 2. Misaligned time series data

After applying certain geometrical techniques and shifting the plots along the time axis, we were able to get the perfectly aligned data that would make it possible to apply further ML techniques on the synced time series data. The same can be observed in *Figure 3*.

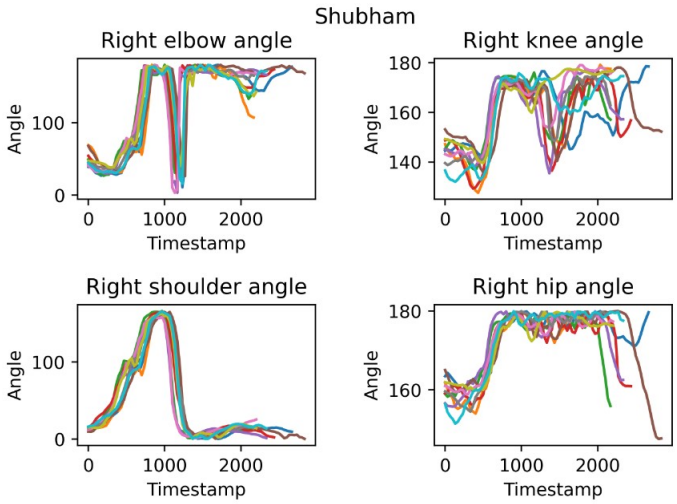

Figure 3. Aligned time series data

4 Machine Learning Models

4.1 Neural Networks

Once we completed the data preprocessing steps, we employed ML and DL algorithms to classify our labelled dataset values to "Good Posture" and "Bad posture". Time series classification algorithms are large and varied. While we have dedicated algorithms like Distance based KNN with *Dynamic Time Warping*, Dictionary based methods like *BOSS/cBOSS, Shapelet Transform classifier* etc., that tend to perform better than traditional methods that take each time stamp as a separate features to classify the data.

In our case, we decided to look at both these approaches to the problem. First, going the traditional route, we tried different ensemble models that include *RandomForestClassifier*, *AdaBoosting* on *Decision Tree Classifier* among aother methods based on [6]. We found that these methods did not perform well on our performance metrics. They tended to obtain very high precision values, however the tradeoff was too high to consider these models satisfactory.

Finally implemented a **Neural Network** model with two hidden dense layers with 32 amd 16

neurons each as shown below. Our classification would be obtained from the sigmoid probability values of the output layer.

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_33 (Dense)            (None, 64)                576

 dense_34 (Dense)            (None, 32)                2080

 dense_35 (Dense)            (None, 16)                528

 dense_36 (Dense)            (None, 1)                 17

=================================================================
Total params: 3,201
Trainable params: 3,201
Non-trainable params: 0
_________________________________________________________________
```

Figure 4. Layers of the Neural Network

When we trained our Neural Network model, we ran into the problem of *target leakage*. When we split our dataset which has continually varying datapoints, randomly into train and test datasets, these splits will have very little variation between them. Since the subset of the points in each of these splits will be very similar, the model is observed to perform very well, even giving a 98% accuracy on the test dataset. However this is not representative of the actual capability of the model, since it does not perform well when exposed to new data that is dissimilar to our original dataset. To fix this problem of data leakage, we decided to split our data subject wise, and let the model "see" data belonging to only a certain subsection of our subjects. Training and testing splits were made by assigning some subject videos to only train and others to only test. The results obtained after training the same model were as follows

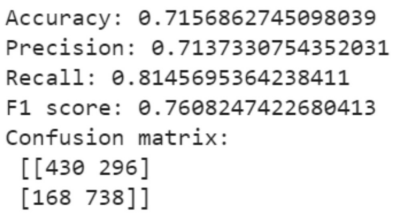

Figure 5. Metrics of the Neural Network

Since our requirement was to implement live posture tracking and feedback, we decided to include this model in our website. Although it has only 71% accuracy, we used this in our final output as we required live predictions. The other models that will be enumerated below required the data of the entire video before classification was possible, and so we have used them in the "Upload video" section of our website. To combat the problems faced by 71% accuracy as well as to reduce flickering in our posture predictions we used rolling prediction averaging.

Rolling prediction averaging is a technique used in video classification to improve the accuracy of predictions made by a classifier over multiple frames. In this technique, instead of predicting the class of each frame independently, the classifier makes predictions for a window of frames(we've considered 5 frames), and then the predicted probabilities are averaged over the window to produce a single prediction. The window is then shifted to the next set of frames, and the process is repeated to get our final predictions.

4.2 Shapelets

The second approach involved a **Shapelet-Based Classifier** technique: a method dedicated for time series data. A shapelet is a subsequence, or sub-shape, of a time series that represents a characteristic pattern of the class. Through this method local patterns of motion and shape can be identified and used to classify videos into different categories. By using something similar to a sliding window mechanism, shapelets are chosen and then enumerated over the entire time series to find a discriminative shapelet that will serve as a representation of the feature. The presence of certain shapelets can make one class more likely than another, and these shapelet features can be used to interpret the shapelet-based classifier. The Shapelet Transform Classifier first selects the top k shapelets that are representative sub-sequences of time series that can

help distinguish between different classes of data.

Next, for each time series in the dataset, k features are computed. These features are calculated as the distance between each time series and each one of the k shapelets, resulting in k columns of features, one for each shapelet. These computed features can then be used to train a classifier to predict the class of a given time series based on its similarity to the set of shapelets. Any vector-based classification algorithm can be applied to the shapelet-transformed dataset, for this purpose we used *SVM*, and *Random forest classifier* on these feature vectors. SVM gave an accuracy of 63% where as Random forest performed significantly better at the classification with an accuracy of 81%.

4.3 LSTM

we built a two-layered **LSTM** model using the *Keras* deep learning framework to predict the success of a basketball free throw based on the angles of the player's elbow, knee, hip and shoulder joints. We preprocessed the dataset by dropping unnecessary columns (like unnamed, timestamps, etc) and encoding the video titles by using label encoder so as to convert categorical data to numerical data. We also split the dataset into groups based on the video title. Each group represents a set of basketball free throw attempts from the same video. We then trained the LSTM model on each group of data by taking the angles of the elbow, knee, hip and shoulder joints as input and the target variable (success or failure) as output. The model was trained using binary cross-entropy loss and RMSprop optimizer. To evaluate the performance of the model, we calculated the accuracy of the model on each group of data and averaged the accuracy across all groups. The resulting accuracy is a measure of how well the model can predict the success of a basketball free throw based on the angles of the player's joints.

5 Implementation and Website

5.1 Website

We utilized a micro web framework called **Flask** to implement our machine learning (ML) models onto our website. Flask is a lightweight framework that provides only the necessary features for building a web application, and it ignores complex functionalities that are not required for our project. Compared to other web frameworks like Django, Flask is a simpler option that is more suited for smaller projects. One of the key advantages of Flask is its ability to support various functionalities such as *OpenCV*, databases, and ML model integration. We selected Flask because it provided us with the necessary features to host our ML model.

To implement our ML models using Flask, we started by initializing all the necessary functions, such as Flask, render template, request, and response. We organized all of our ML models into separate files for ease of access and simplicity. We then imported the ML functions into different Python modules. Next, we initialized a Flask object and bound it to the variable "app." We assigned specific routes to each ML model and created separate functions for each model. For example, we implemented live posture detection by encoding the data into bytes using simlejpeg module for faster encoding and sending it to the frontend for decoding and display to the user.

We also implemented posture detection of a video using form actions, allowing the user to select a video file from a given directory. The selected video file is then processed by the ML model, which displays the good or bad posture for every frame on a computer vision (CV) console, as seen in *Figure 6* and *Figure 7*. We utilized CSS for designing and styling the websites.

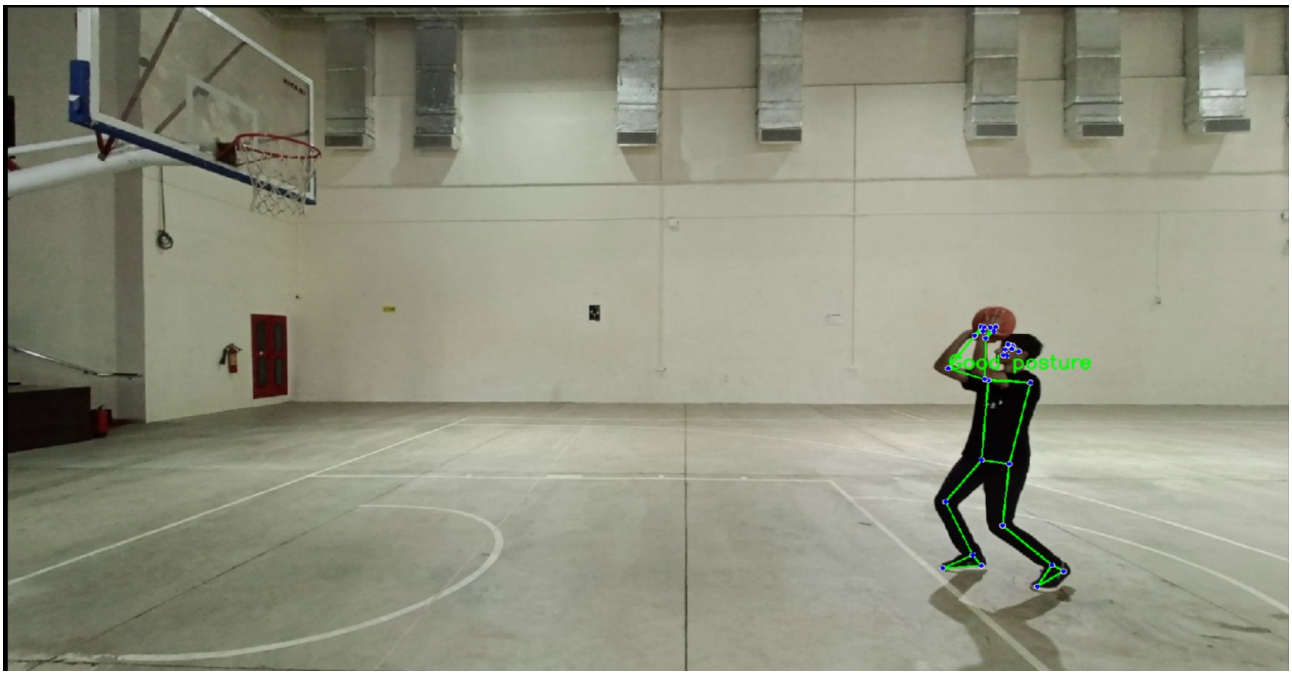

Figure 6. Good posture detection

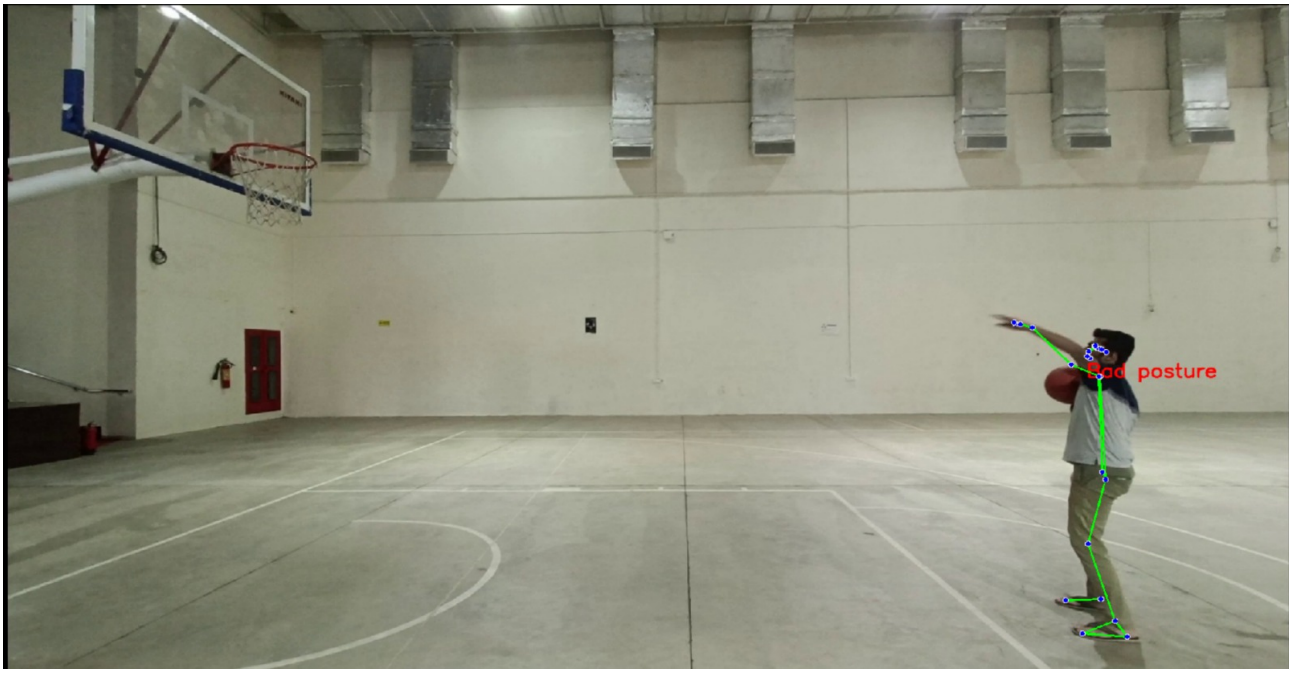

Figure 7. Bad posture detection

5.2 Turtle Graphics

In our research, we employed the **K-means clustering** algorithm to group data points into K-*clusters* based on their similarities. K means clustering is a popular machine learning algorithm that partitions a dataset into k distinct clusters, where each cluster is defined by a centroid, which is the mean of all the data points in that cluster. We utilized this algorithm as part of an image segmentation approach.

One of the benefits of the K-means clustering algorithm is that it does not require prior training. Instead, a single video can be used as a reference for another video. The K-means clustering algorithm is then used to cluster the frames and identify the closest frames for a trainer and trainee video, allowing us to find the difference between them. The difference and the original trainer video are then used to visually show the difference between the video being compared and the video it is being compared with.

To create 2D graphics for our project, we used the turtle graphics library in Python. This library enabled us to generate images and graphics. However, one of the drawbacks of this method is that the movement is not always smooth, as a turtle object has to move every time a different object is created.

It is worth noting that this ML model is different from others we used in our research, as it does not require training, and it can use a single video as a reference. However, we encountered some limitations in our implementation. For example, the turtle graphics library draws graphics based on the front angle, which can lead to some skewed diagrams. Additionally, the clustering could sometimes fail to consider some frames, resulting in skewed outputs. Despite these limitations, our implementation of the K-means clustering algorithm using the turtle graphics library proved to be a valuable tool for our research.

6 Results and Discussions

6.1 LSTM

The accuracy of the two-layered LSTM model we built for predicting the success of a basketball free throw based on the angles of the player's joints was found to be 0.5057. This indicates that the model was not effective in predicting the success of a free throw. There could be several reasons for the poor performance of the two layered LSTM. One possible reason could be the limited amount of data available for training the model. Another reason could be the complexity of the task, as the angles of the player's joints may not be the only determining factor for the success of a free throw. Additionally, there could be noise or variability in the data that is not captured by the model, or the model architecture and hyperparameters may not have been optimized for the task at hand. Overall, these results suggest that further exploration and refinement of the model and data collection methods may be necessary to improve the accuracy of predicting basketball free throw success based on joint angles.

6.2 Neural Network and Shapelets

Our Neural networks model gave a better classification accuracy of 71%, and was thus selected as the model for our implementation and website. The Shapelets classifier in combination with SVM only gave an accuracy of 63%, while in combination with Random Forest Classifier gave an accuracy of 81%. However, this model could not be used for live-feed prediction and was thus only used in predicting clips of basketball free throws.

7 Conclusion

Posture tracking technology has emerged as an innovative and effective approach to improving the postures of basketball players during training and gameplay. Poor posture can have a significant impact on the player's performance, including their ability to execute shots accurately and efficiently. The use of posture tracking technology can provide coaches and players with real-time feedback on their posture and allow for adjustments to be made to improve their alignment, stability, and balance.

A combination of computer vision techniques and machine learning (ML) models has been used to develop posture tracking technology for basketball players. The technology involves the use of computer vision modules to track the player's posture and detect any deviations from optimal alignment. ML models are then used to analyse the data collected and provide feedback to the player and coach on areas where improvements can be made.

The results of studies on posture tracking technology for basketball players have been promising. This technology has been found to be effective in improving the posture of players and helping them to perfect their shots. ML models have been shown to be particularly effective in judging and improving posture, leading to better performances on the court.

In addition to improving performance, posture tracking technology has the potential to reduce the risk of injury for basketball players. Poor posture can increase the likelihood of injury, particularly to the back and shoulders. By monitoring and adjusting posture, players can reduce the stress on their muscles and joints and lower the risk of developing chronic injuries.

However, there are some limitations to the use of posture tracking technology in basketball. For example, the technology may be expensive and require specialized training to use effectively. Additionally, it may be difficult to incorporate the technology into gameplay without disrupting the flow of the game.

Despite these limitations, posture tracking technology has the potential to revolutionize the way basketball players train and compete. With continued research and development, the technology

could become a standard tool for coaches and players looking to improve their performance and reduce the risk of injury.

References

[1] Dimitrije Cabarkapa, Andrew C. FryA, Kevin M. Carlson, John P. Poggio, Michael A. Deane et al, 2021. Key Kinematic Components for Optimal Basketball Free Throw Shooting Performance. *Central European Journal of Sports Sciences and Medicine vol. 36.*

[2] Kehan Chen et al, 2019. Sitting Posture Recognition Based on OpenPose. *doi:10.1088/1757-899X/677/3/032057.*

[3] Shihao Hou, Bizhen Lian, Wenhao Li and Hong Tang et al, 2022. A Basketball Training Posture Monitoring Algorithm Based on Machine Learning and Artificial Intelligence. *DOI:10.1155/2022/2264659.*

[4] Colby T. Jeffries et al, 2018. Sports Analytics With Computer Vision. *Senior Independent Study Theses. Paper 8103.*

[5] Lexiang Ye and Eamonn Keogh et al, 2009. Time Series Shapelets: A New Primitive for Data Mining. *https://doi.org/10.1145/1557019.1557122.*

[6] Mustafa Chasmai, Nirjhar Das, Aman Bhardwaj & Rahul Garg, 2022. A View Independent Classification Framework for Yoga Postures. *SN Computer Science volume 3, Article number: 476.*