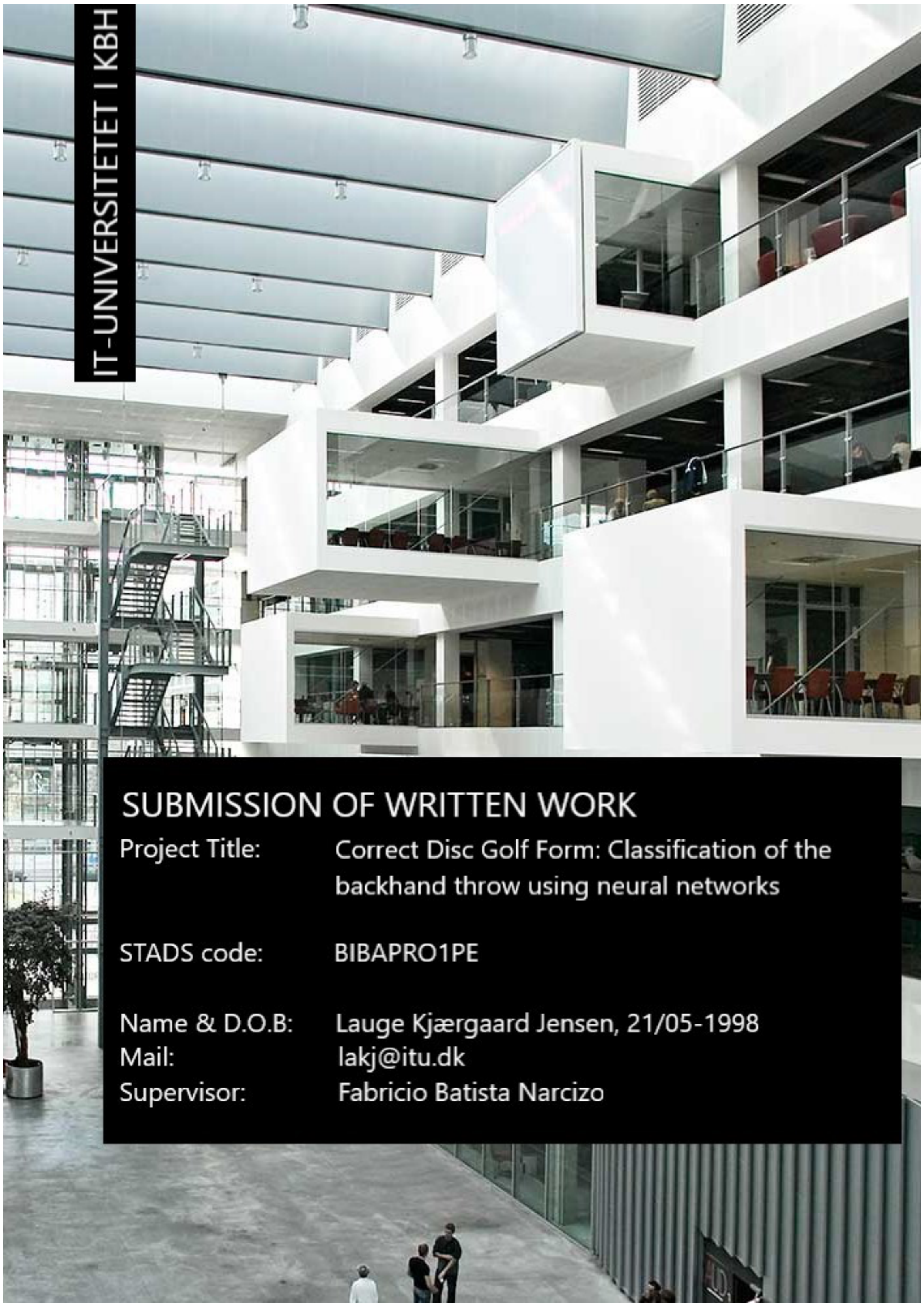


# SUBMISSION OF WRITTEN WORK

| | |
|---|---|
| Project Title: | Correct Disc Golf Form: Classification of the backhand throw using neural networks |
| STADS code: | BIBAPRO1PE |
| Name & D.O.B: | Lauge Kjærgaard Jensen, 21/05-1998 |
| Mail: | lakj@itu.dk |
| Supervisor: | Fabricio Batista Narcizo |

# Contents

## Abbreviations

**Disc** A frisbee designed for disc golf. Weighs more and can fly further and faster than a regular frisbee.

**Pull** The motion of pulling the disc along your body before throwing it.

**Backhand** The most commonly used throw in disc golf. Derives its name from tennis because the motion is reminiscent of the tennis backhand stroke.

**Plant Foot** For a right handed player this is the right foot which is planted on the ground just before pulling the shot.

**Form** The motion of body poses that defines form in sports.

**RNN** Recurrent neural network

**LSTM** Long short-term Memory neural network

**PCA** Principal Component Analysis

## Abstract

Form is essential when analyzing and reviewing a backhand disc golf throw. The form defines if the throw is performed correctly and the poses of the body define the form. By looking at the body poses the throw can be classified, critiqued, and improved upon. The form consists of different motions which are analyzed using 3D data collected using machine learning solutions on a data set of recorded disc golf throws. By processing the 3D data from recorded throws the form is classified into three classes that represent the start, mid, and end of the throw. The three classes are shown as clusters using Principal Component Analysis (PCA). The PCA showed more overlapping clusters for the start and middle of the throw compared to the end. Classification solutions include a variation of trained LSTM networks and a solution using MediaPipe Pose Classification. The paper concludes that LSTM models perform faster and more accurately than the solution using MediaPipe Pose Classification when analyzing disc golf throws. However, the classification only provides insight for classifying the different forms and not the quality of form.

# 1 Introduction

Analyzing sports has been around for as long as sports have however the use of big data is changing the way that sports are being analyzed. Sports analytics can provide major insights for all parties involved in a given sport [7]. It can help the coaches and players with training as well as improve decision-making for other parties involved. It can also improve fan engagement by providing a better live game experience using software like shot tracers and providing statistics to the viewer. Big data and machine learning solutions rely heavily on having access to tons of data. And the number of statistics and videos collected in sports in recent times has skyrocketed due to general improvements in technology. The way amateur and professional athletes evolve their game has changed due to these improvements. Nowadays athletes are constantly processing their statistics so that they can focus on and improve in areas where they would be underperforming. In a golf setting a player might be below average when comparing putting on the greens. Or it could be having bad accuracy and distance when driving from the tees, resulting in missed fairways. Whatever the problem the solution is often analyzing what went wrong and where it could be improved upon. In modern times many shots are recorded in slow-motion which allows one to analyze the form and break it down into the parts that define it. For all sports, there are specific techniques that i.e. define how one should position one's body and perform the required technique. When learning a new technique in a sport it is important that it is learned properly since unlearning bad technique can be very hard. However, when the technique is learned properly it is also important to review it to ensure consistency across time.

Disc golf is a sport much like golf however instead of swinging a golf club at a ball the throw is a motion where a frisbee is pulled across the body before being released. This motion is especially intriguing since it can be broken down into classes that define the throw. Form breakdown is the most essential for revealing what allows one person to throw 200-300 feet accurately and another to throw 500-600 feet accurately. And since golf is a mental game it can be a weapon to have the ability to outdrive your opponent on the golf course to get the mental edge. Therefore form critique is essential for any player to improve their game or ensure that they stay consistent across time. However, is it possible to break down the form of a disc golf shot and classify it using modern machine learning solutions to get potential knowledge that could allow one to improve one's shot and ensure that it doesn't degrade?

This project will collect data from recorded disc golf shots and do classification of three classes of the backhand disc golf throw. The project provides multiple trained LSTM models which can review videos of disc golf shots and classify the frames which are a part of the classes defined in the model. The LSTM models will be compared to a solution from MediaPipe[5] which provides a pose classification solution. Both solutions will be trained using the same footage.

The thesis is structured in the following sections: “Background” presents background information about the theory of throwing a disc, the APIs used for processing the videos, LSTM network, and MediaPipe Pose Classification. “Data & Results” presents the data collection, data processing, training the network, results of PCA, results from LSTM network models, and results of processed videos. “Analysis & Discussion” includes analysis and discussion of sections from “Data & Results”. The thesis is concluded with the section “Conclusion” and “Reflection”.

## 1.1 Thesis Statement

This project aims to classify the motion of performing a backhand disc golf throw to research what correct form is. The motion of the shot will be analyzed using ML framework MediaPipe Pose. By dividing the motion of the shot into phases before, during and after the shot this project aims to research what body poses are correct when performing a backhand throw.

# 2 Background

## 2.1 Performing the throw

When performing a backhand throw in the sport of disc golf there are many factors that influence the quality of the shot. Among these factors is the grip of the disc, the angle of the disc, the run-up, the timing, and most importantly the pull from the reachback through the powerpocket to the followthrough. This is called a player's disc golf form. Every player has small tweaks to the form that make their style unique. It might be having a slightly lower reachback or higher release through the followthrough. But in its essence, every ideal disc golf form is very similar. From the many factors that define a great shot this project will mainly work with the pull of the disc.

### 2.1.1 Reachback

The reachback is defined from the moment that the disc is reached all the way back and your body is ready to begin pulling through [6]. This means that your throwing arm is fully extended at the same moment as your plant foot is planted on the ground. The head is turned with the shoulders and the reachback is in a straight line. The key is timing your plant foot hitting the ground at the same time your reachback is fully extended. From this position, you start to pull the disc across your body in a straight line.

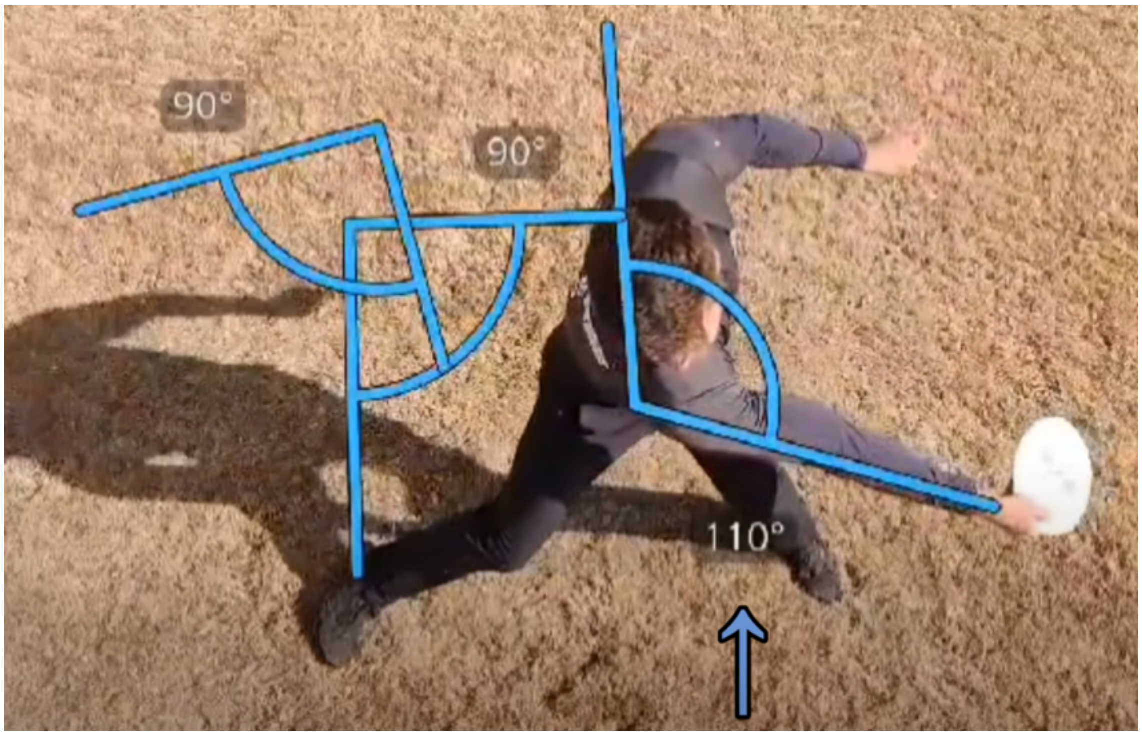


Figure 1: Reachback

The reachback consists of a sequence of frames that start when the reachback is initiated until the powerpocket is reached (see figure 2).

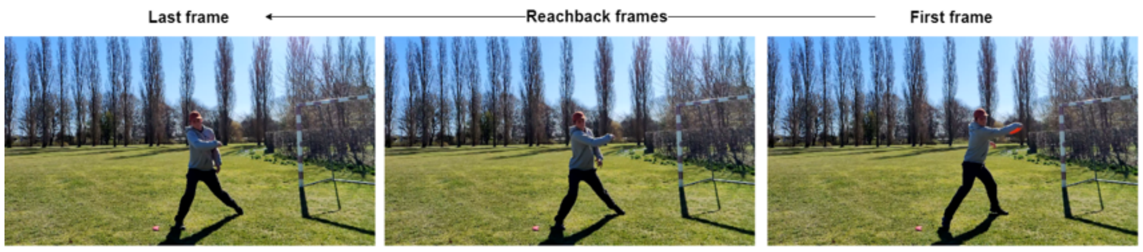


Figure 2: Reachback sequence of frames

### 2.1.2 Powerpocket

The powerpocket is defined from the moment that your right elbow and both shoulders create a 90-degree angle. This is also called the hitbox or hitpoint where the shoulders, right elbow, and the disc in hand create a perfect box. It is from the powerpocket that the explosive behavior of the shot is created. To create extra power the left arm is tugged down and close to the body while pushing your shoulders forward [2].

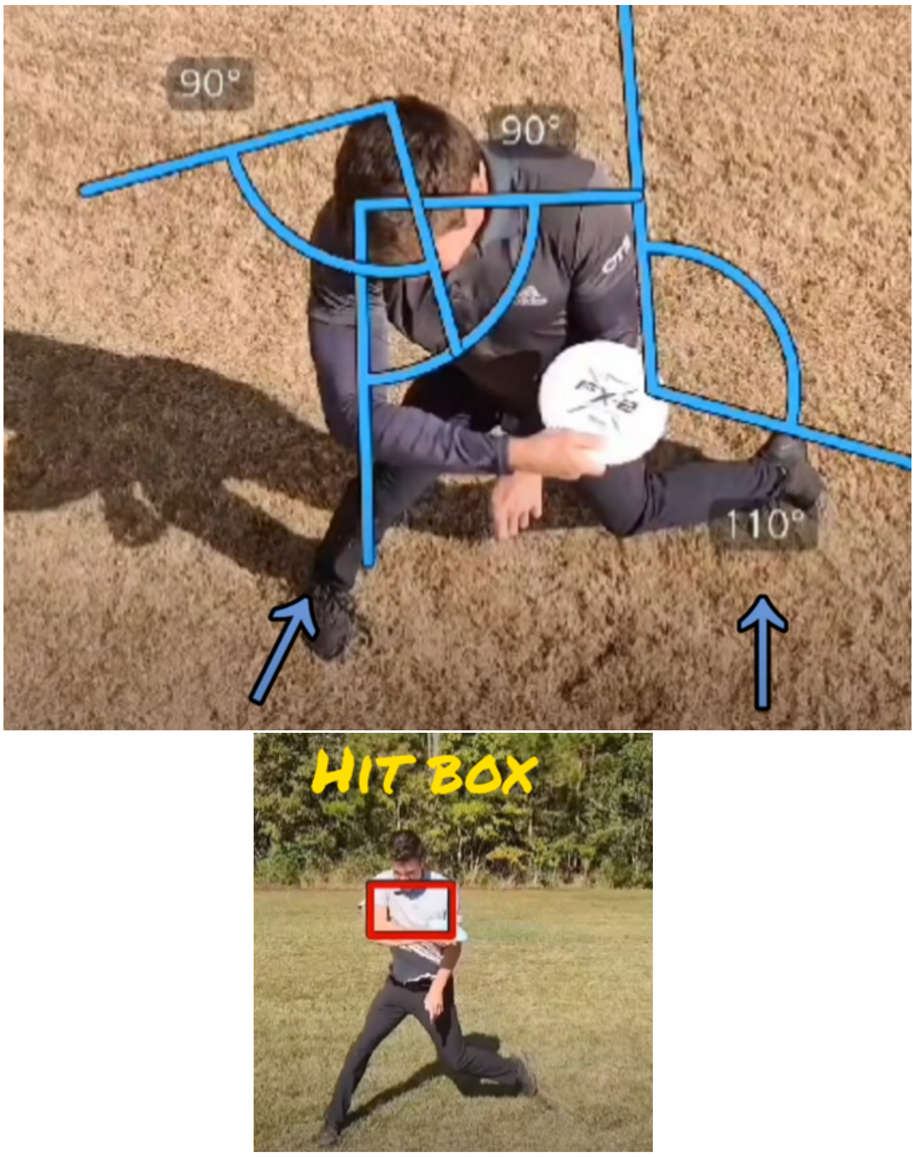


Figure 3: Powerpocket and perfect box

The powerpocket is the fastest part of the throw and consists of the fewest frames compared to the reachback and the followthrough. The sequence of frames that represent the powerpocket starts with the perfect box and ends when releasing the disc from the hand (see figure 4).

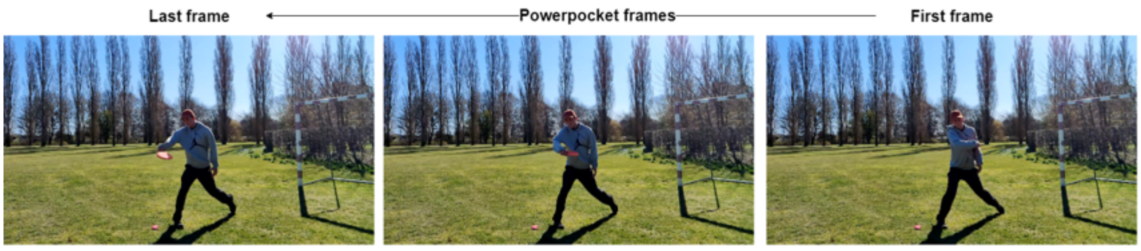


Figure 4: Powerpocket sequence of frames

### 2.1.3 Followthrough

The followthrough is defined from the moment that the disc leaves your hand. At this point, you rotate on your plant foot and completely follow through with your arm and whole body. This ensures that you are following through on the explosive behavior and not putting unnecessary strain on the body by abruptly stopping the rotation.

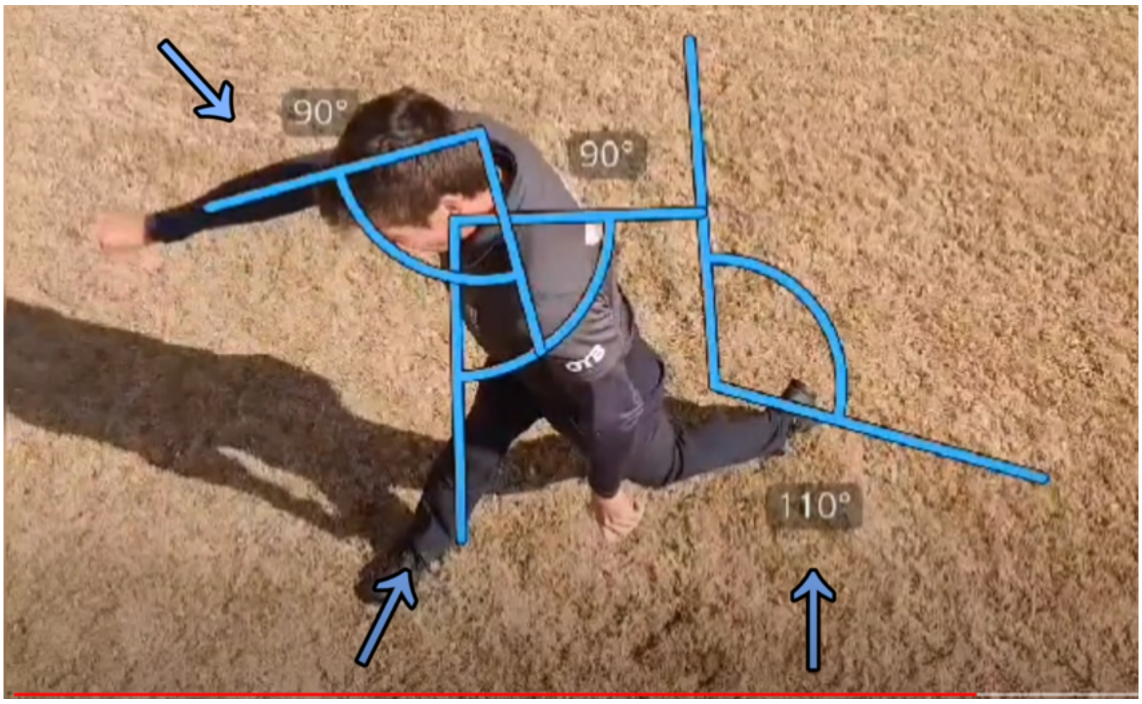


Figure 5: Followthrough

The sequence of frames that represent the followthrough is defined from the moment the disc leaves the hand until the rotation of the body has come through. The followthrough is the longest part of the throw and consists of the most amount of frames (see figure 6).

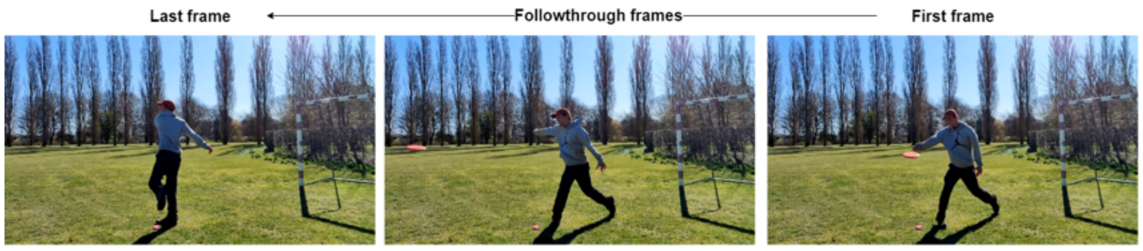


Figure 6: Followthrough sequence of frames

### 2.1.4 Other crucial factors

The pull of the disc is performed at the last step of the run-up most commonly called the x-step. The x-step is a three-step motion that puts the body in the ideal position for performing the throw. The power which is generated from the x-step (running motion) is transferred into the shot by performing the throw from the reachback through the powerpocket to the followthrough. The x-step is as followed:

1. One step with dominant foot at 45 degree angle
2. Non dominant foot crosses behind first step creating a X of the lower body (see figure 7).
3. Dominant foot steps in front with a 90 degree angle which is timed with the reachback. The pull is initiated.

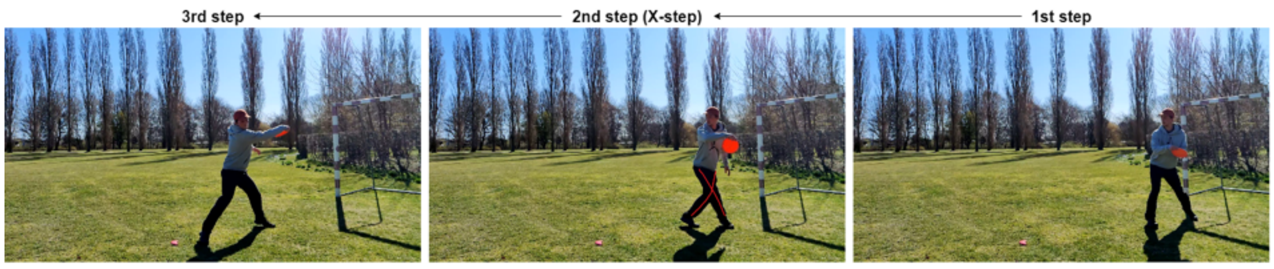


Figure 7: X-step

Besides how the shot is performed the type of disc which is used has a huge impact on the flight of the throw. In regular golf the golf ball is always the same however there are many types of clubs. For disc golf it is the opposite. The discs come in different categories: driver, fairway-driver, midrange, and putter. For each shot, the player is allowed to choose whatever disc they want. The driver is used for longer drives and the putter for more control at short distances. Discs also have different stabilities that define how it flies when released from the hand. A shot thrown with the backhand of a right-handed player will naturally finish left due to the spin direction of the disc produced from the throw. Overstable discs want to finish early whereas understable discs will flip and turn before wanting to finish in their natural direction. Stable discs naturally want to go more straight (see figure 8). The flight of each disc is represented using flight numbers (see figure 9) that are available for every disc on the market. The four numbers define the speed, glide, turn, and fade of the disc (see appendix 9 for a more detailed explanation).

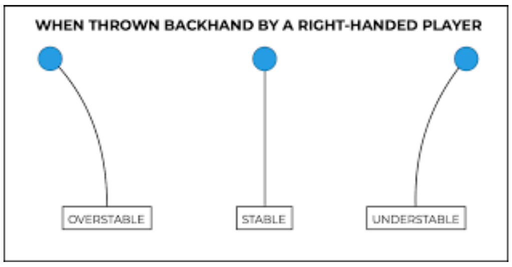


Figure 8: Stabilities of discs

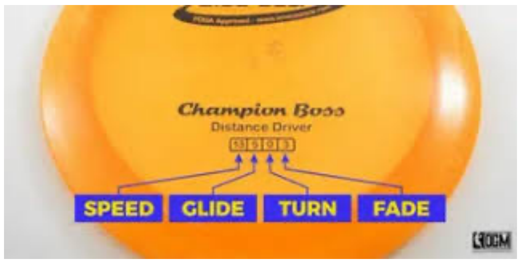


Figure 9: Flightnumbers

The angle at which the disc is released is described using the disc golf terminology hyzer, flat, and anhyzer (see figure 10). Using different discs and angles, shots can be shaped to specific lines. Fairways are often designed such that specific shots are required for the player to score and avoid out of bounds.

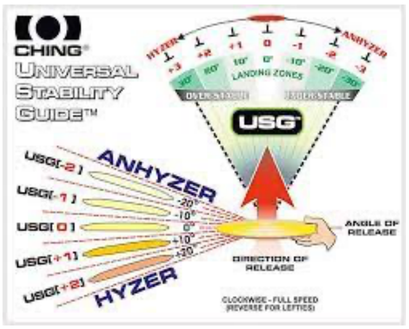


Figure 10: Release angles

## 2.2 OpenCV & MediaPipe Pose

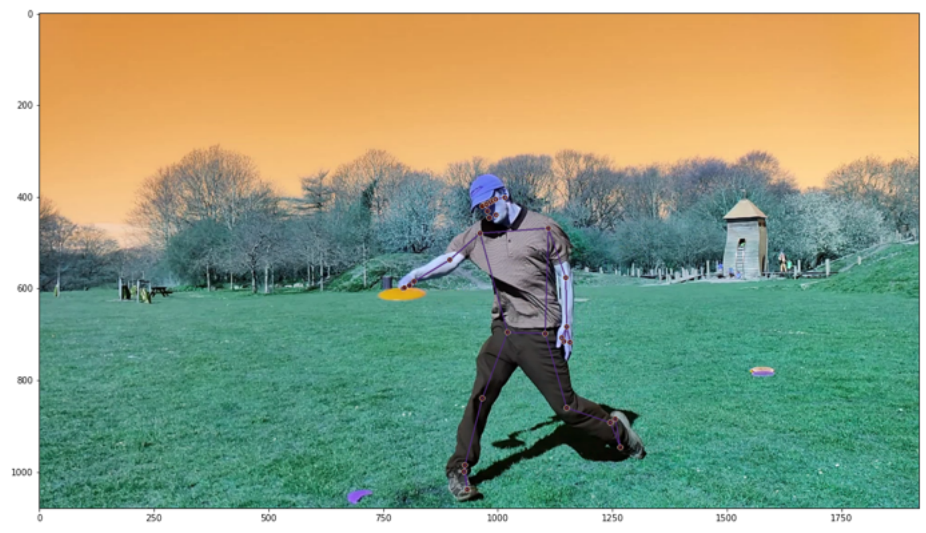


Figure 11: Pose landmarks

On the frame from figure 11, there are defined 33 landmarks that are classifying the human body pose. The 33 landmarks are collected from the MediaPipe Pose ML solution that predicts the 33 pose landmark [4]. Each landmark contains an x, y, z, and visibility value. The x and y values define a coordinate normalized to [0.0, 1.0] by the image width and height. The z value defines the depth where the midpoint of the hips is the origin. Lastly, the visibility value defines the probability of the landmark being visible. OpenCV is used in this project for processing recorded videos of disc golf throws. Every frame is read from the video and processed using the solution from MediaPipe Pose. The 33 pose landmarks are drawn on the frame and connected to visualize the position of the body.

## 2.3 RNN & LSTM Network

The three classes defined for the pull of the disc golf shot consist of a movement of the body that is performed over a short amount of time. Since the shots are recorded the movement of a given class is represented as a sequence of frames holding the positional data from MediaPipe Pose. Therefore the Neural Network should take into account the previous data from previous frames when trying to classify what part of the shot is occurring. This is where the structure of the RNN network is extremely useful. The RNN network has a loop on itself that allows the information to go through steps of the network, which allows the information to persist. However, the gap can grow too large and in that case, RNNs become unable to connect the previous information when trying to predict the new output. When using an LSTM network that problem is avoided since the LSTM network is designed to avoid the long-term dependency problem [1].

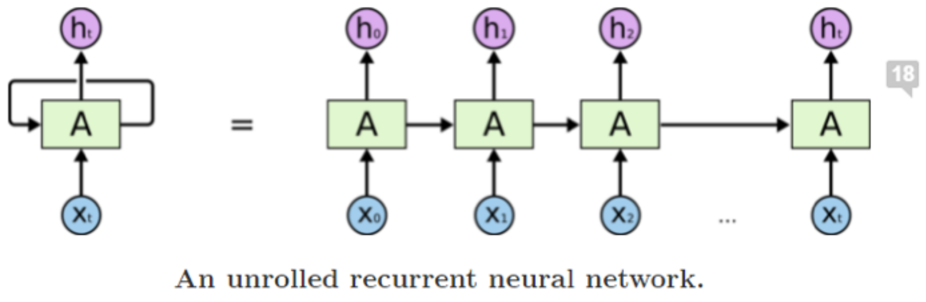


Figure 12: Unrolled RNN

### 2.3.1 LSTM network

An LSTM network is a structure based on RNN but instead of having one single neural network layer, it has four. The four layers add a gating structure so that each LSTM unit has a forgetting gate, input gate, and output gate that can remove and/or add information to the cell state. The cell state can be seen as a conveyor belt running straight through the recurring network. In the cell state information can flow through but also take minor linear interactions.

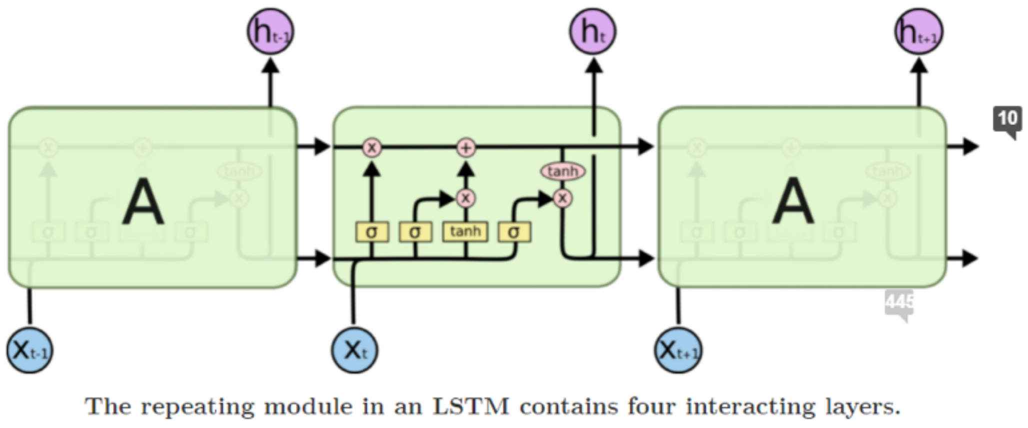

h_{t-1}
h_t
h_{t+1}
A
σ
σ
tanh
σ
tanh
A
x_{t-1}
x_t
x_{t+1}
The repeating module in an LSTM contains four interacting layers.

Figure 13: LSTM network

The forget gate layer decides what information to discard from the cell state. It does so by looking at the previous input and the current input and outputting a value between 0 and 1. 1 meaning discard all information while 0 meaning keep all information. Next up the input gate decides what values to update. The updated values are passed through a sigmoid layer. The input gate layer also has a tanh layer that produces new candidate values, which could be added to the cell state. Then the update to the cell state is made with the values from the forget gate layer and the input gate layer. At last, the output gate layer will output a filtered version of the cellstate. It does so by deciding what values to output by putting the cellstate through a sigmoid layer. Then it applies a tanh operation to the cellstate and multiplies it with the output from the sigmoid layer. That results in only outputting the parts that were decided [1].

## 2.4 MediaPipe Pose Classification

MediaPipe also provides a solution for pose classification and repetition counting that can be used instead of creating and training a neural network. The solution uses the k-nearest neighbor's algorithm (k-NN). K-NN classifies an object's class based on the closest samples from the training set. The training set for the k-NN classifier uses x, y, and z values from pose landmarks. A good training set is defined by the solution as having a few hundred samples for each terminal state. The terminal states for a disc golf throw are reachback, powerpocket, and followthrough (as defined in 2.1). The solution invokes k-NN search twice with different metrics for a better classification result. First, it picks top-N samples by a max distance (set to 30) which allows removing samples where the pose is almost the same but where joints are bent in other directions. Second, it picks top-N samples by mean distance (set to 10) to get the samples that are closest on average. Afterward, it performs exponential moving average (EMA) smoothing to remove noise from the classification [5].

# 3 Data & Results

## 3.1 Data Collection

### 3.1.1 Initial Dataset

The initial dataset consists of 30 shots from 3 professional players (Eagle McMahon, Calvin Heimberg, Paul McBeth) who are among the current top 5 rated players in the world [3]. The dataset was collected from GateKeeperMedias Youtube channel where various footage of slow-motion form checks is uploaded. The videos were filmed at unknown camera angles and distances. The footage was shot on a GoPro Hero 8 at 240 fps and slowed down to 25 percent and therefore playing at 60 fps (see appendix 9). As a start to the project, five of the shots were divided into the classes reachback, powerpocket, and followthrough. The five shots were used in the early data processing. The shots were divided into three classes using a video editor. By going frame by frame one shot was divided into three videos where each video had the sequence of frames that defined the part of the throw (as described in 2.1). The dataset had the potential to be expanded since GateKeeperMedia had a total of six videos of around 15 minutes in length with slow-motion form checks.

### 3.1.2 Final Dataset

The final dataset consists of 55 shots from two amateur players who have played for over two years. The dataset was collected 19. April 2022 using the camera on a OnePlus 8. The videos were shot in 1080p with 240 fps. The camera was placed on a pod at a height of 105 cm. The distance from the camera to the player was measured at 400 cm (see figure 14). For each shot from the dataset, the video was divided into the target classes reachback, powerpocket, and followthrough as described 2.1. By going frame for frame with a video editor each shot was divided and cropped to the three target classes and placed in the following folder structure (see appendix 8 under train_test_videos/final_dataset/).

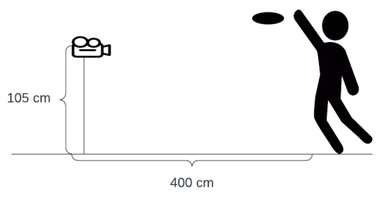


Figure 14: Setup footage

## 3.2 Data Processing

### 3.2.1 Initial Dataset

Since the initial dataset consisted of videos with only 60 fps that resulted in a shorter sequence of frames for the classes reachback, powerpocket, and followthrough. Therefore the value for maximum size for the sequence of frames could only be four since the powerpocket occurred in around 4-7 frames in the dataset. From the initial dataset, five shots were processed using MediaPipe Pose and the landmarks were written to a .csv file for early analysis. A PCA graph with the five shots was created to analyze the clustering of the different classes (3.5).

### 3.2.2 Final Dataset

When processing a shot from the final dataset each frame was written to "shots_train_v4.csv" (see appendix 8 for file) with pose landmarks and the according class (reachback, powerpocket, and followthrough). Each entry in the .csv file also has a "series_id". The "series_id" describes the sequence of frames that a given frame is a part of. The final dataset is processed such that each sequence consists of ten frames. The sequence is moved one frame forward when incrementing the "series_id" (see figure 15).

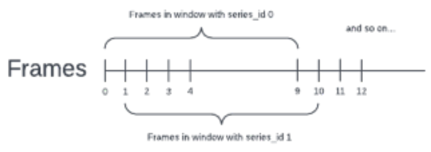
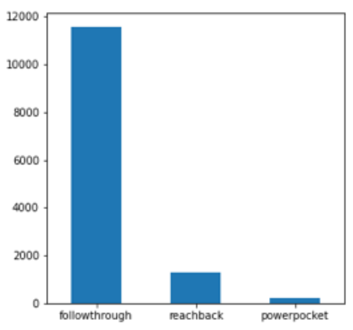

Frames in window with series_id 0
and so on...
Frames
0
1
2
3
4
9
10
11
12
Frames in window with series_id 1

Figure 15: Window of frames

A movement of the body is defined as a continuous series of frames (a sequence) with the according pose data from the landmark list. The data produced from the recorded shots show that the amount of series_id's differs depending on the class. That is because the sequence of frames that define the reachback is shorter because the movement is shorter. The reachback movement is only represented in around 30 frames of data for each shot. The powerpocket is even shorter and only represented in around 10-15 frames. On the other hand, the followthrough is a much longer movement and is represented in around 300 frames of data. So when processing a movement of 14 frames (i.e. a powerpocket) from a shot from the dataset, only 5 sequences of data is written to the .csv file. Therefore it is visible that the amount of data for each class differs (see figure 16).



```
followthrough    11569
reachback         1298
powerpocket        210
Name: class, dtype: int64
```

Figure 16: Value counts for dataset

A processed shot is represented by the time series plotted with the values from pose landmarks. The classes reachback, powerpocket, and followthrough that are specified in the class column of the .csv file occur in specific time intervals. To research if the classes are clustered, dimensionality reduction is performed using Principal Component analysis. The PCA reduces the data to 2 dimensions. If there is reasonable clustering the data can be fit to a neural network. The time series can also be plotted with data written with the series id that defines the window size (see figure 19).

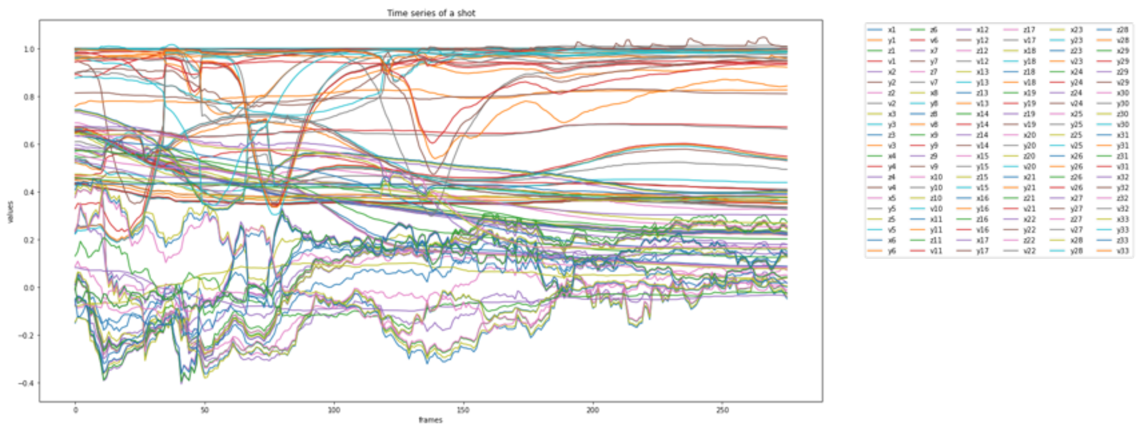


Figure 17: Time series with no windows

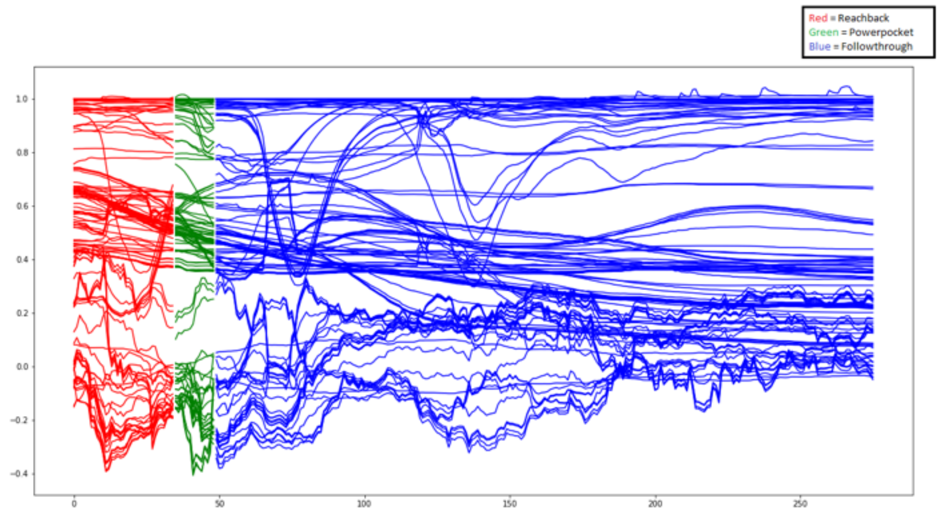


Figure 18: Time series from reachback, powerpocket, followthrough

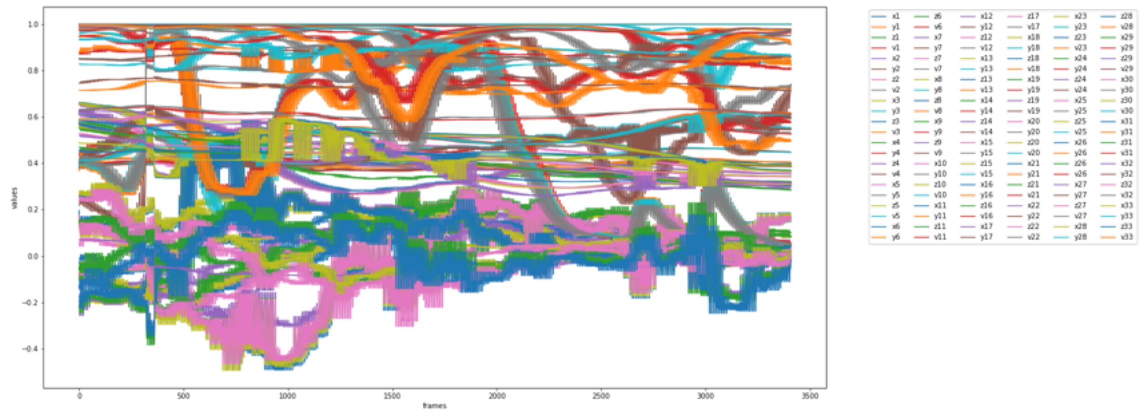


Figure 19: Time series with series_id

## 3.3 TensorFlow & LSTM network

The neural network for classifying disc golf shots is built using Keras TensorFlow open-source platform for machine learning [9]. Using a Sequential model the network is defined with the layers depicted in figure 20.

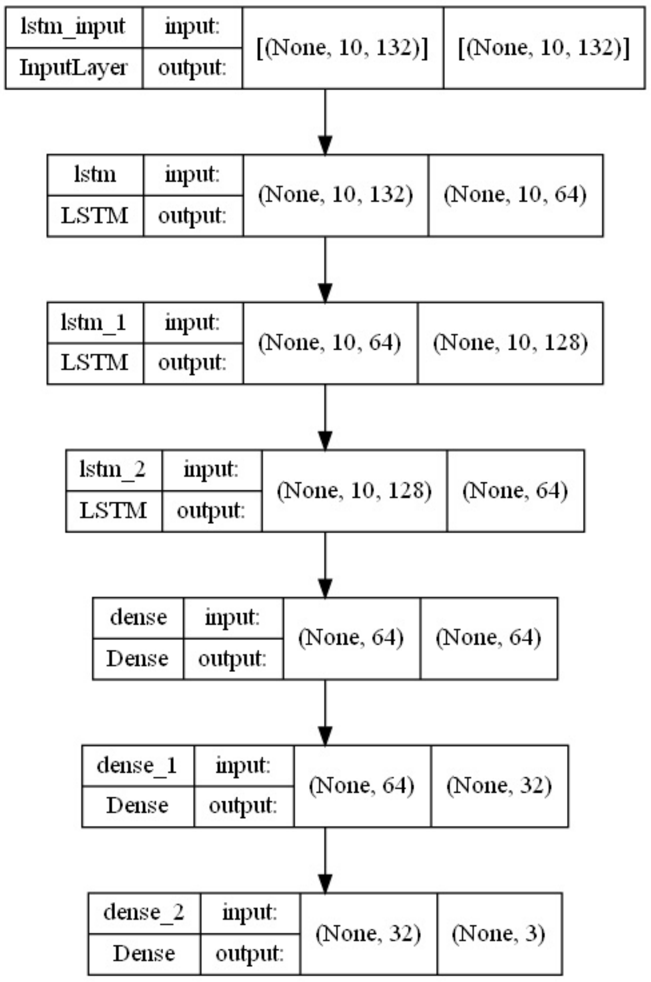


Figure 20: TensorFlow Sequential Model

The sequential model is used to stack layers that only have one input tensor and one output tensor. The first layer in the model is the input layer and is an LSTM layer that takes 10 inputs of 132 values. This corresponds to a sequence of 10 frames of positional data from MediaPipe Pose. In some of the other LSTM models, the input layer is changed to take a sequence of five frames and only x, y, and z values from MediaPipe Pose. The input layer has 64 units which is the number of neurons that the input_shape is connected to (see figure 21). The whole model is defined using three LSTM layers, followed by three Dense layers. The last layer (output layer) has three units and uses the activation function 'softmax' for multinomial probability distribution. Meaning the output layer will return three values. Each value describes the probability for each class of the disc golf throw (as defined in 2.1).

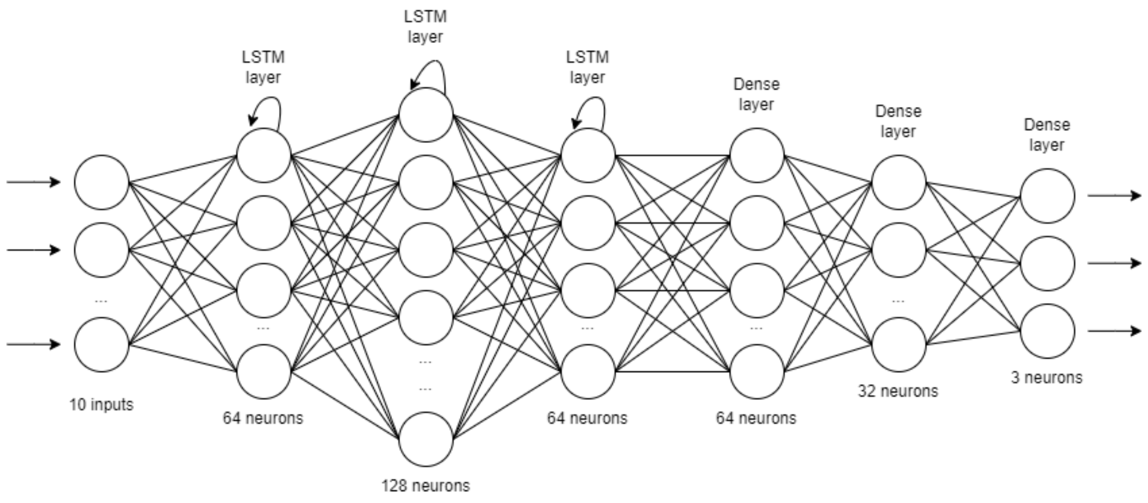
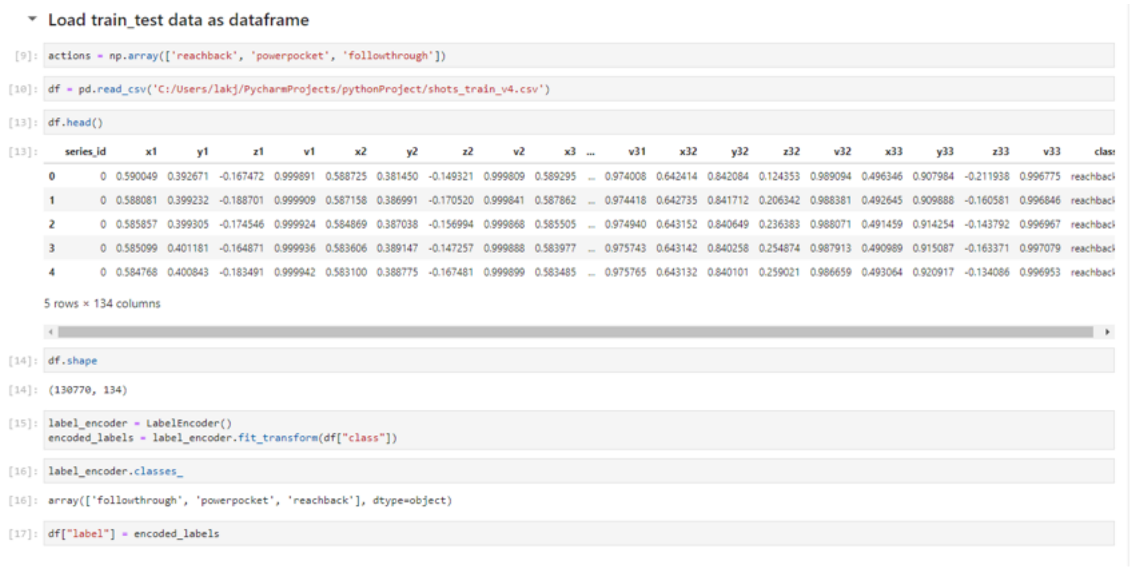


Figure 21: LSTM neural network

### 3.3.1 Training the models

Before training the models, the data from the shots_train_v4.csv (see PredictionModel.ipynb on GitHub 9) file is processed to fit the input shape for the given model. The labels denoting the target class are encoded to an integer representation of the class which is required for the model defined in TensorFlow. The sequences for the input variables (X) and the encoded labels for the output variables (y) are done by grouping on the series_id and appending the landmarks data to a sequence array and the encoded labels to a feature array. Using the train_test_split method from sklearn.model_selection[8] the input variables (X) and the output variables (y) are split into train and test sets for X and y (X_train, X_test, y_train, y_test). The data is split such that 25 percent is used for testing and the rest is used for training (see figure 22, figure 23).



**Load train_test data as dataframe**

```
[9]:  actions = np.array(['reachback', 'powerpocket', 'followthrough'])

[10]: df = pd.read_csv('C:/Users/lakj/PycharmProjects/pythonProject/shots_train_v4.csv')

[13]: df.head()
```

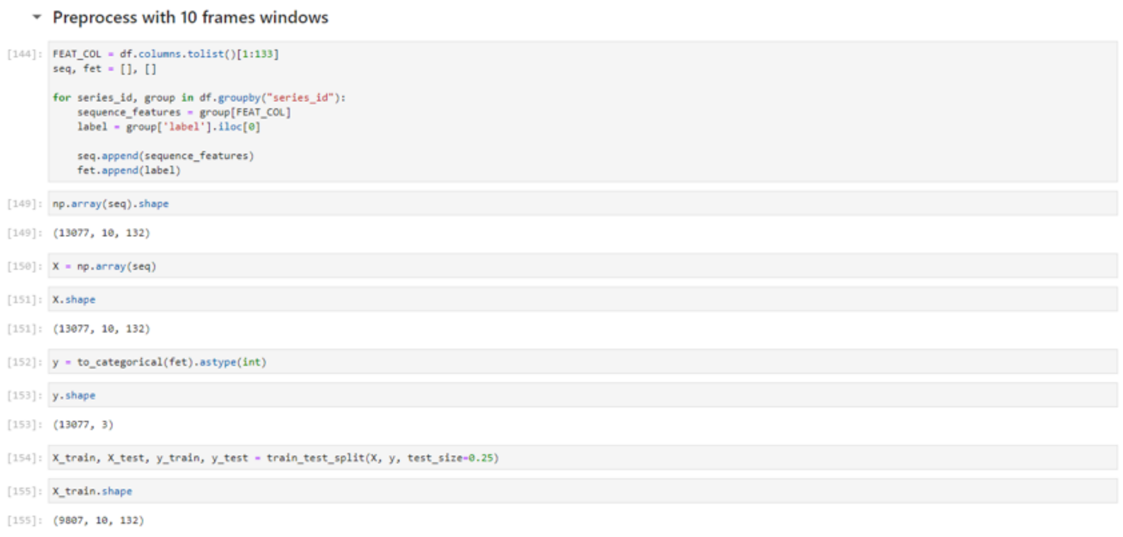
[13]:

| | series_id | x1 | y1 | z1 | v1 | x2 | y2 | z2 | v2 | x3 | ... | v31 | x32 | y32 | z32 | v32 | x33 | y33 | z33 | v33 | clas |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0.590049 | 0.392671 | -0.167472 | 0.999891 | 0.588725 | 0.381450 | -0.149321 | 0.999809 | 0.589295 | ... | 0.974008 | 0.642414 | 0.842084 | 0.124353 | 0.989094 | 0.496346 | 0.907984 | -0.211938 | 0.996775 | reachback |
| 1 | 0 | 0.588081 | 0.399232 | -0.188701 | 0.999909 | 0.587158 | 0.386991 | -0.170520 | 0.999841 | 0.587862 | ... | 0.974418 | 0.642735 | 0.841712 | 0.206342 | 0.988381 | 0.492645 | 0.909888 | -0.160581 | 0.996846 | reachback |
| 2 | 0 | 0.585857 | 0.399305 | -0.174546 | 0.999924 | 0.584869 | 0.387038 | -0.156994 | 0.999868 | 0.585505 | ... | 0.974940 | 0.643152 | 0.840649 | 0.236383 | 0.988071 | 0.491459 | 0.914254 | -0.143792 | 0.996967 | reachback |
| 3 | 0 | 0.585099 | 0.401181 | -0.164871 | 0.999936 | 0.583606 | 0.389147 | -0.147257 | 0.999888 | 0.583977 | ... | 0.975743 | 0.643142 | 0.840258 | 0.254874 | 0.987913 | 0.490989 | 0.915087 | -0.163371 | 0.997079 | reachback |
| 4 | 0 | 0.584768 | 0.400843 | -0.183491 | 0.999942 | 0.583100 | 0.388775 | -0.167481 | 0.999899 | 0.583485 | ... | 0.975765 | 0.643132 | 0.840101 | 0.259021 | 0.986659 | 0.493064 | 0.920917 | -0.134086 | 0.996953 | reachback |

5 rows × 134 columns

```
[14]: df.shape

[14]: (130770, 134)

[15]: label_encoder = LabelEncoder()
      encoded_labels = label_encoder.fit_transform(df["class"])

[16]: label_encoder.classes_

[16]: array(['followthrough', 'powerpocket', 'reachback'], dtype=object)

[17]: df["label"] = encoded_labels
```

Figure 22: Load .csv file and encode labels



**Preprocess with 10 frames windows**

```
[144]: FEAT_COL = df.columns.tolist()[1:133]
       seq, fet = [], []

       for series_id, group in df.groupby("series_id"):
           sequence_features = group[FEAT_COL]
           label = group['label'].iloc[0]

           seq.append(sequence_features)
           fet.append(label)

[149]: np.array(seq).shape

[149]: (13077, 10, 132)

[150]: X = np.array(seq)

[151]: X.shape

[151]: (13077, 10, 132)

[152]: y = to_categorical(fet).astype(int)

[153]: y.shape

[153]: (13077, 3)

[154]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)

[155]: X_train.shape

[155]: (9807, 10, 132)
```

Figure 23: Split data into X_train, X_test, y_train, y_test

The model is trained with an EarlyStopping callback to prevent overfeeding the network. The EarlyStopping callback is monitoring the loss of the training with a patience of three. So if the loss is not decreasing for three epochs the model will halt training. Four models are trained using the following sequence size and input shape:

| **Model Name** | **sequence size** | **input shape** |
|---|---|---|
| LSTM_model_10 | 10 frames | (10, 132) |
| LSTM_model_10_XYZ | 10 frames | (10, 99) |
| LSTM_model_5 | 5 frames | (5, 132) |
| LSTM_model_5_XYZ | 5 frames | (5, 99) |

## 3.4 MediaPipe Pose Classification

The training set for MediaPipe Pose Classification defines the classes reachback, powerpocket, and followthrough as defined in 2.1. MediaPipe Pose Classification does not use temporal information like the LSTM network and therefore the k-NN classifier takes only one frame as an input. The training dataset is specified from the final dataset such that x, y, and z values from each frame are written to three separate .csv files. Each .csv file defines the data for one of the classes reachback, powerpocket, or followthrough. The data is read from the three .csv files (see appendix 8 under csv_files/mediapipe_pose_classification_data/). The 55 shots from the final dataset produce the following amount of data that is used for the MediaPipe Pose Classification solution:

| **Class/target** | **Number of frames** |
|---|---|
| Reachback | 1793 |
| Powerpocket | 705 |
| Followthrough | 12064 |

MediaPipe Pose Classification is done in Google Colab (see appendix 2). The solution expects image samples of the different poses which then a bootstrap helper converts to a .csv file for each target class. Instead of using the bootstrap helper the target .csv files were created by reading each frame from the final dataset and writing the 3D data to shots_train1f_MP.csv (see csvToMP_pose_class_form.ipynb from GitHub, appendix 9). The solution from MediaPipe Pose Classification expects a single .csv file for each target class that was created by selecting all rows that fit one class and writing it to a single .csv file.

## 3.5 Results of PCA

### 3.5.1 Results from PCA graph computed from the inital dataset

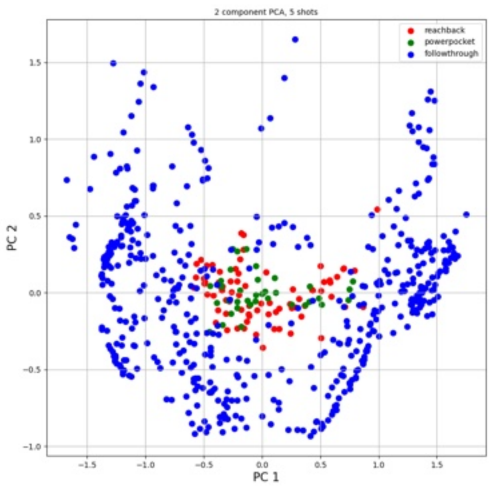


Figure 24: PCA of 5 shots no window

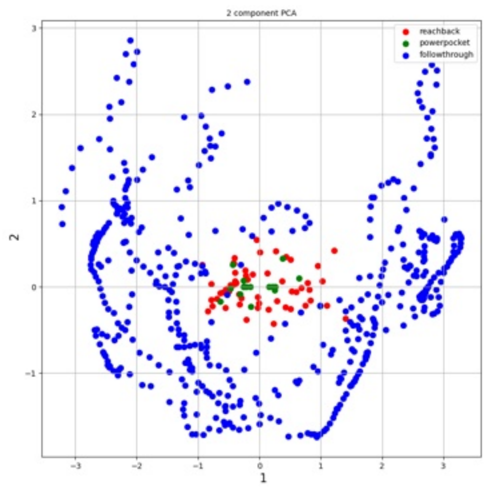


Figure 25: PCA of 5 shots window of 4 frames

Principal Component Analysis performs dimensionality reduction on the pose landmarks data that were written to the .csv file for each frame in a given shot. In figure 24 the 132 values from the 33 landmarks represented in each frame are reduced to two-dimensional data using the PCA algorithm. From the results plotted in figure 24 there is noticeable clustering of the classes reachback, powerpocket, and followthrough. However, there is some overlapping between the reachback and the powerpocket. The same goes for figure 25 where the PCA algorithm performs dimensionality reduction on sequences of four frames of landmarks instead of one. The results are produced from five shots from the initial dataset.

### 3.5.2 Results from PCA graph computed from the final dataset

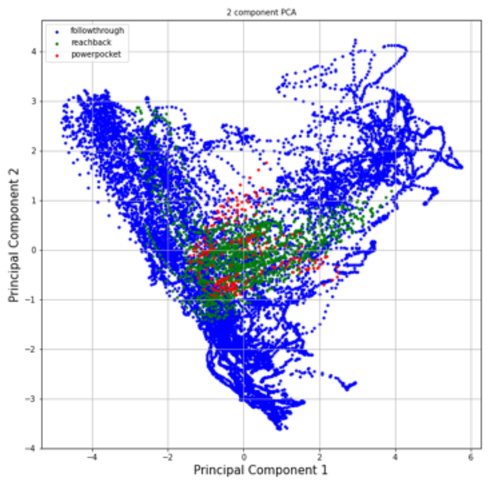


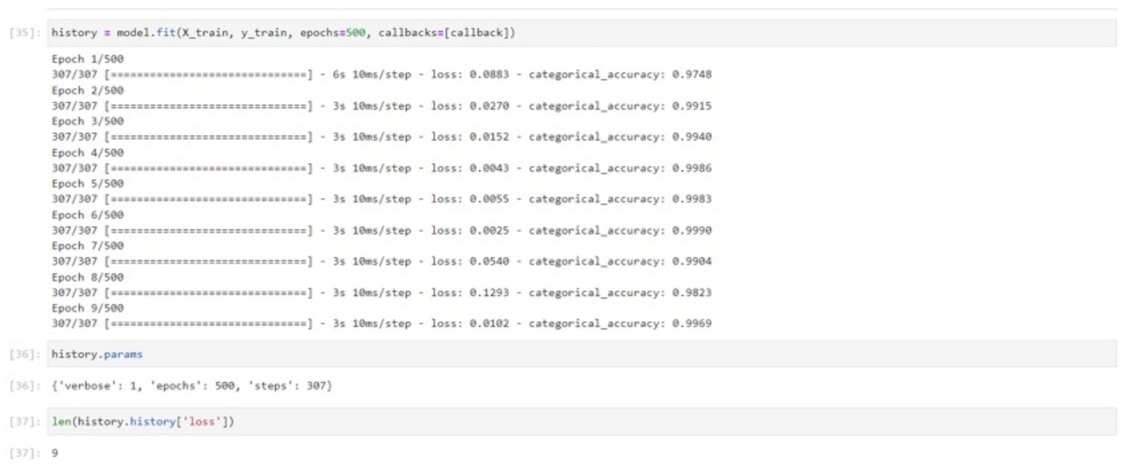
Figure 26: PCA of final dataset

Figure 26 shows all PCA values from all shots from the final dataset ("shots_train_v4.csv"). The PCA algorithm is fed a sequence of 10 frames of pose data. It is also noticeable that there is considerable clustering between the classes. The reachback and the powerpocket are still the classes with the most overlapping clusters.

## 3.6 Results of LSTM network

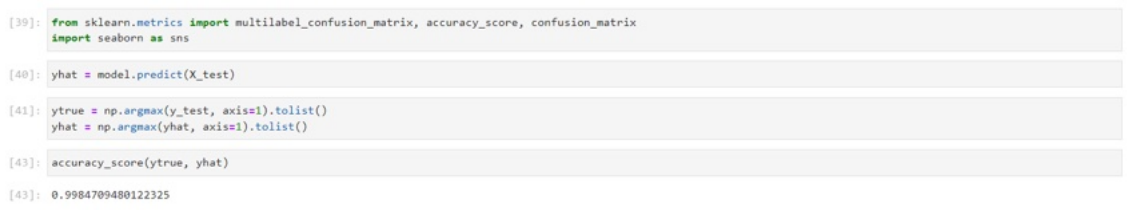
### 3.6.1 Model with 10 frames input (X, Y, Z, Visibility)

The model halts training after 9 epochs because the loss stops decreasing for 3 epochs as specified by the EarlyStopping callback. The model is evaluated with X_test (3.3.1) on which it performs great. In over 99 percent of the test cases, the model predicts the correct class. As shown in figure 29 the model predicts all inputs of followthrough correctly. However, for the powerpocket and the reachback, the model mispredicted for five of the cases. Once where it predicted followthrough for a powerpocket and the rest between powerpocket and reachback. It is shown in 3.5.1 and 3.5.2 that the powerpocket and the reachback have the most overlapping clusters computed with PCA. Therefore it is also expected that the model would have a harder time predicting between those two classes. The results are shown by the following:



```
[35]: history = model.fit(X_train, y_train, epochs=500, callbacks=[callback])

      Epoch 1/500
      307/307 [==============================] - 6s 10ms/step - loss: 0.0883 - categorical_accuracy: 0.9748
      Epoch 2/500
      307/307 [==============================] - 3s 10ms/step - loss: 0.0270 - categorical_accuracy: 0.9915
      Epoch 3/500
      307/307 [==============================] - 3s 10ms/step - loss: 0.0152 - categorical_accuracy: 0.9940
      Epoch 4/500
      307/307 [==============================] - 3s 10ms/step - loss: 0.0043 - categorical_accuracy: 0.9986
      Epoch 5/500
      307/307 [==============================] - 3s 10ms/step - loss: 0.0055 - categorical_accuracy: 0.9983
      Epoch 6/500
      307/307 [==============================] - 3s 10ms/step - loss: 0.0025 - categorical_accuracy: 0.9990
      Epoch 7/500
      307/307 [==============================] - 3s 10ms/step - loss: 0.0540 - categorical_accuracy: 0.9904
      Epoch 8/500
      307/307 [==============================] - 3s 10ms/step - loss: 0.1293 - categorical_accuracy: 0.9823
      Epoch 9/500
      307/307 [==============================] - 3s 10ms/step - loss: 0.0102 - categorical_accuracy: 0.9969

[36]: history.params

[36]: {'verbose': 1, 'epochs': 500, 'steps': 307}

[37]: len(history.history['loss'])

[37]: 9
```

Figure 27: LSTM model 10 frames x,y,z,v fit



```
[39]: from sklearn.metrics import multilabel_confusion_matrix, accuracy_score, confusion_matrix
      import seaborn as sns

[40]: yhat = model.predict(X_test)

[41]: ytrue = np.argmax(y_test, axis=1).tolist()
      yhat = np.argmax(yhat, axis=1).tolist()

[43]: accuracy_score(ytrue, yhat)

[43]: 0.9984709480122325
```

Figure 28: LSTM model 10 frames x,y,z,v accuracy on test

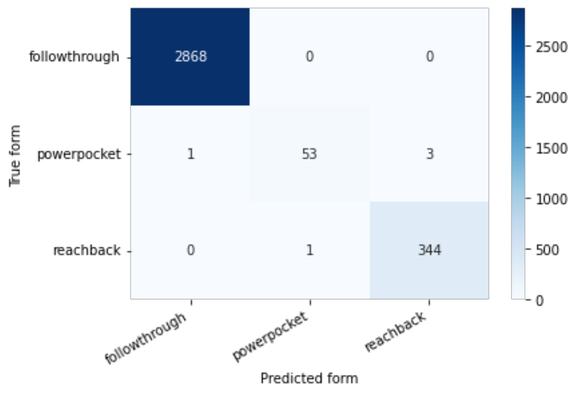
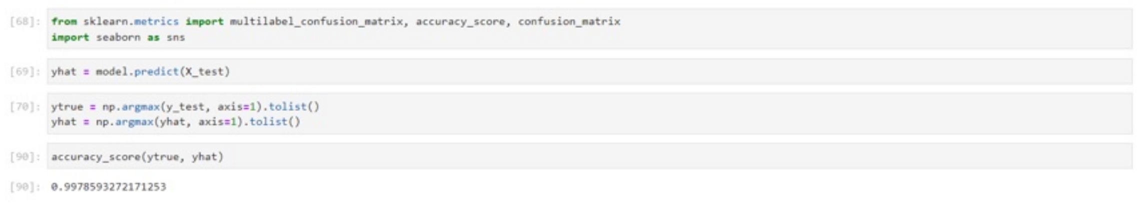


Figure 29: LSTM model 10 frames x,y,z,v confusion matrix

### 3.6.2 Model with 10 frames input (X, Y, Z)

The model trained with an input shape of (10, 99) also performs well and has an accuracy score of over 99 percent. As in 3.6.1, the model performs very well for the followthrough and only mispredicts a small amount between the reachback and the powerpocket. The results are shown by the following:



```
[68]: from sklearn.metrics import multilabel_confusion_matrix, accuracy_score, confusion_matrix
      import seaborn as sns

[69]: yhat = model.predict(X_test)

[70]: ytrue = np.argmax(y_test, axis=1).tolist()
      yhat = np.argmax(yhat, axis=1).tolist()

[90]: accuracy_score(ytrue, yhat)

[90]: 0.9978593272171253
```

Figure 30: LSTM model 10 frames x,y,z accuracy on test

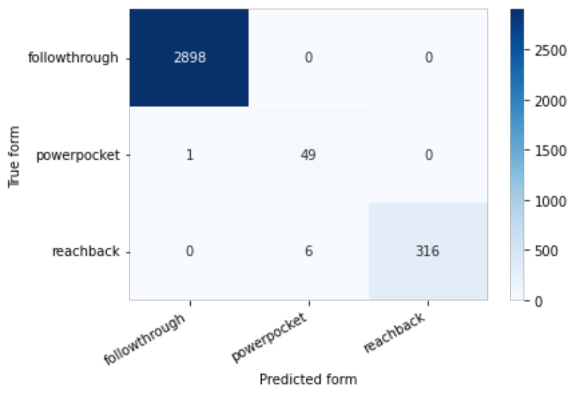
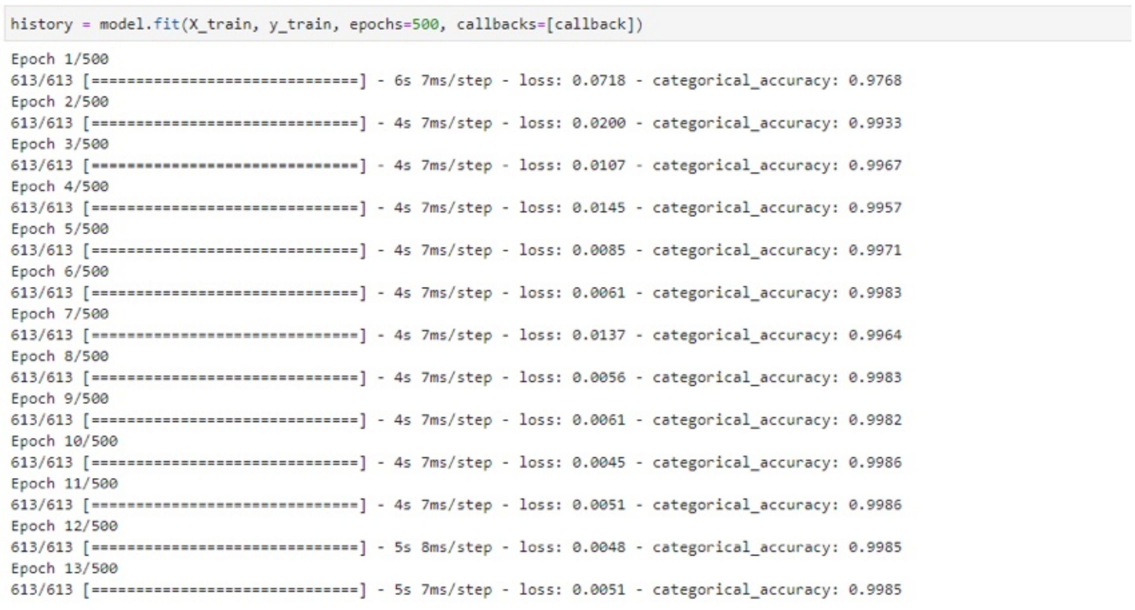


Figure 31: LSTM model 10 frames x,y,z confusion matrix

### 3.6.3 Model with 5 frames input (X, Y, Z, Visibility)

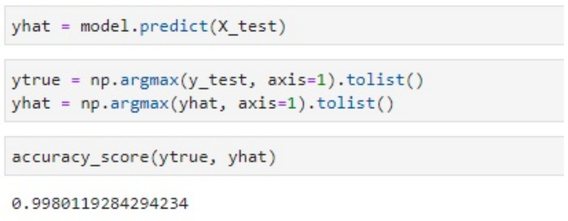
The model trained for 13 epochs before halting. The accuracy of the model is over 99 percent on the test data. The model mispredicted seven inputs from the test data. The results are shown by the following:



```
history = model.fit(X_train, y_train, epochs=500, callbacks=[callback])
```

```
Epoch 1/500
613/613 [==============================] - 6s 7ms/step - loss: 0.0718 - categorical_accuracy: 0.9768
Epoch 2/500
613/613 [==============================] - 4s 7ms/step - loss: 0.0200 - categorical_accuracy: 0.9933
Epoch 3/500
613/613 [==============================] - 4s 7ms/step - loss: 0.0107 - categorical_accuracy: 0.9967
Epoch 4/500
613/613 [==============================] - 4s 7ms/step - loss: 0.0145 - categorical_accuracy: 0.9957
Epoch 5/500
613/613 [==============================] - 4s 7ms/step - loss: 0.0085 - categorical_accuracy: 0.9971
Epoch 6/500
613/613 [==============================] - 4s 7ms/step - loss: 0.0061 - categorical_accuracy: 0.9983
Epoch 7/500
613/613 [==============================] - 4s 7ms/step - loss: 0.0137 - categorical_accuracy: 0.9964
Epoch 8/500
613/613 [==============================] - 4s 7ms/step - loss: 0.0056 - categorical_accuracy: 0.9983
Epoch 9/500
613/613 [==============================] - 4s 7ms/step - loss: 0.0061 - categorical_accuracy: 0.9982
Epoch 10/500
613/613 [==============================] - 4s 7ms/step - loss: 0.0045 - categorical_accuracy: 0.9986
Epoch 11/500
613/613 [==============================] - 4s 7ms/step - loss: 0.0051 - categorical_accuracy: 0.9986
Epoch 12/500
613/613 [==============================] - 5s 8ms/step - loss: 0.0048 - categorical_accuracy: 0.9985
Epoch 13/500
613/613 [==============================] - 5s 7ms/step - loss: 0.0051 - categorical_accuracy: 0.9985
```

Figure 32: LSTM model 5 frames x,y,z,v fit



```
yhat = model.predict(X_test)
```

```
ytrue = np.argmax(y_test, axis=1).tolist()
yhat = np.argmax(yhat, axis=1).tolist()
```

```
accuracy_score(ytrue, yhat)
```

```
0.9980119284294234
```

Figure 33: LSTM model 5 frames x,y,z,v accuracy on test

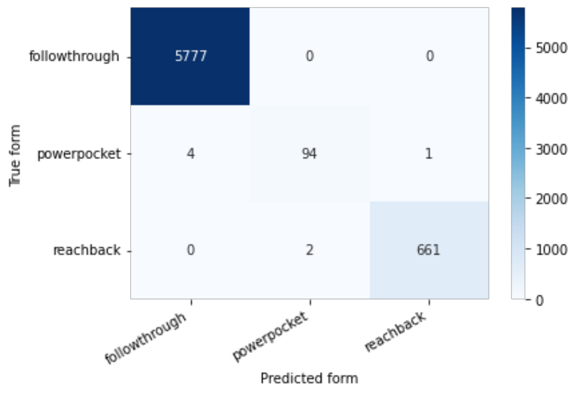
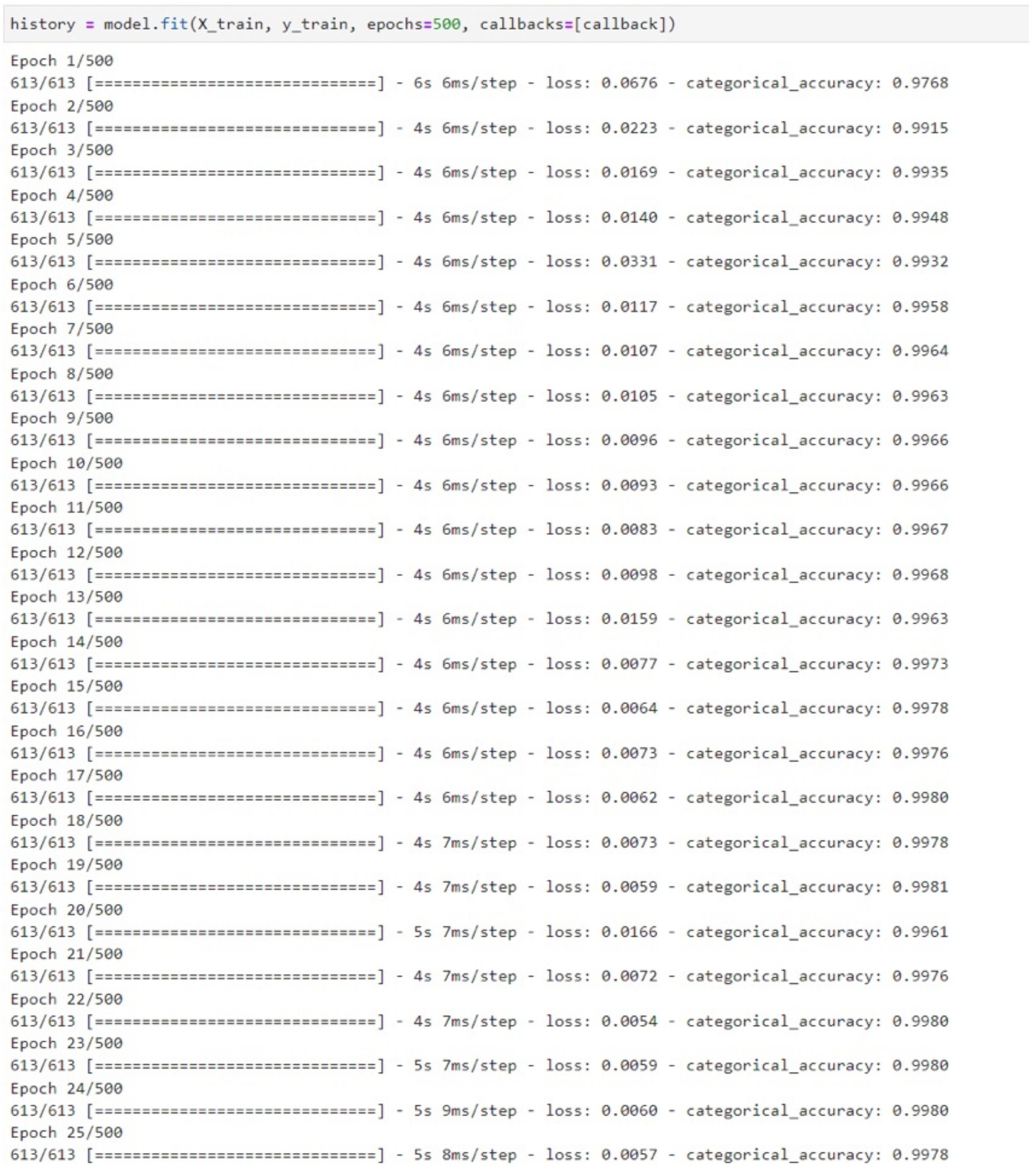


Figure 34: LSTM model 5 frames x,y,z,v confusion matrix

### 3.6.4 Model with 5 frames input (X, Y, Z)

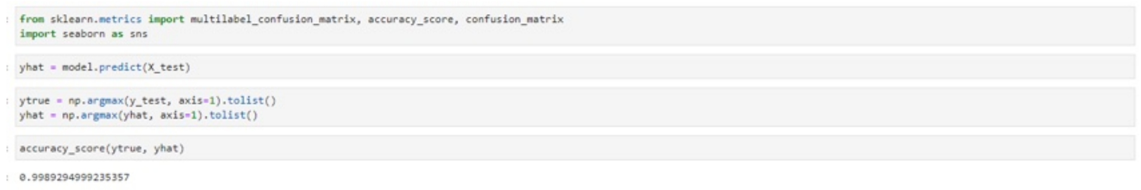
The model trained for 25 epochs before the earlyStopping callback was called. The model with a window of five frames with 3D data from the 33 pose landmarks also performs at over 99 percent accuracy on the test data set. The results are shown by the following:



```
history = model.fit(X_train, y_train, epochs=500, callbacks=[callback])
```

```
Epoch 1/500
613/613 [==============================] - 6s 6ms/step - loss: 0.0676 - categorical_accuracy: 0.9768
Epoch 2/500
613/613 [==============================] - 4s 6ms/step - loss: 0.0223 - categorical_accuracy: 0.9915
Epoch 3/500
613/613 [==============================] - 4s 6ms/step - loss: 0.0169 - categorical_accuracy: 0.9935
Epoch 4/500
613/613 [==============================] - 4s 6ms/step - loss: 0.0140 - categorical_accuracy: 0.9948
Epoch 5/500
613/613 [==============================] - 4s 6ms/step - loss: 0.0331 - categorical_accuracy: 0.9932
Epoch 6/500
613/613 [==============================] - 4s 6ms/step - loss: 0.0117 - categorical_accuracy: 0.9958
Epoch 7/500
613/613 [==============================] - 4s 6ms/step - loss: 0.0107 - categorical_accuracy: 0.9964
Epoch 8/500
613/613 [==============================] - 4s 6ms/step - loss: 0.0105 - categorical_accuracy: 0.9963
Epoch 9/500
613/613 [==============================] - 4s 6ms/step - loss: 0.0096 - categorical_accuracy: 0.9966
Epoch 10/500
613/613 [==============================] - 4s 6ms/step - loss: 0.0093 - categorical_accuracy: 0.9966
Epoch 11/500
613/613 [==============================] - 4s 6ms/step - loss: 0.0083 - categorical_accuracy: 0.9967
Epoch 12/500
613/613 [==============================] - 4s 6ms/step - loss: 0.0098 - categorical_accuracy: 0.9968
Epoch 13/500
613/613 [==============================] - 4s 6ms/step - loss: 0.0159 - categorical_accuracy: 0.9963
Epoch 14/500
613/613 [==============================] - 4s 6ms/step - loss: 0.0077 - categorical_accuracy: 0.9973
Epoch 15/500
613/613 [==============================] - 4s 6ms/step - loss: 0.0064 - categorical_accuracy: 0.9978
Epoch 16/500
613/613 [==============================] - 4s 6ms/step - loss: 0.0073 - categorical_accuracy: 0.9976
Epoch 17/500
613/613 [==============================] - 4s 6ms/step - loss: 0.0062 - categorical_accuracy: 0.9980
Epoch 18/500
613/613 [==============================] - 4s 7ms/step - loss: 0.0073 - categorical_accuracy: 0.9978
Epoch 19/500
613/613 [==============================] - 4s 7ms/step - loss: 0.0059 - categorical_accuracy: 0.9981
Epoch 20/500
613/613 [==============================] - 5s 7ms/step - loss: 0.0166 - categorical_accuracy: 0.9961
Epoch 21/500
613/613 [==============================] - 4s 7ms/step - loss: 0.0072 - categorical_accuracy: 0.9976
Epoch 22/500
613/613 [==============================] - 4s 7ms/step - loss: 0.0054 - categorical_accuracy: 0.9980
Epoch 23/500
613/613 [==============================] - 5s 7ms/step - loss: 0.0059 - categorical_accuracy: 0.9980
Epoch 24/500
613/613 [==============================] - 5s 9ms/step - loss: 0.0060 - categorical_accuracy: 0.9980
Epoch 25/500
613/613 [==============================] - 5s 8ms/step - loss: 0.0057 - categorical_accuracy: 0.9978
```

Figure 35: LSTM model 5 frames x,y,z fit



```
from sklearn.metrics import multilabel_confusion_matrix, accuracy_score, confusion_matrix
import seaborn as sns
```

```
yhat = model.predict(X_test)
```

```
ytrue = np.argmax(y_test, axis=1).tolist()
yhat = np.argmax(yhat, axis=1).tolist()
```

```
accuracy_score(ytrue, yhat)
```

```
0.9989294999235357
```

Figure 36: LSTM model 5 frames x,y,z accuracy on test

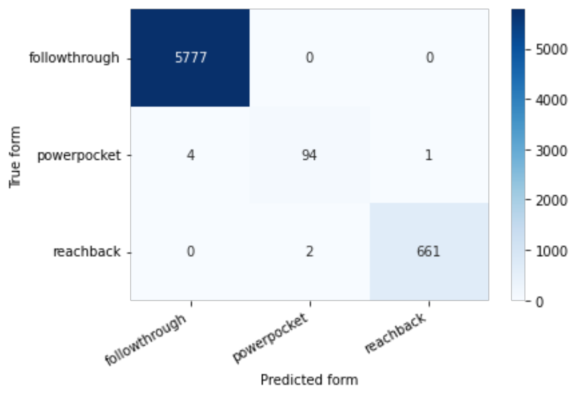


Figure 37: LSTM model 5 frames x,y,z confusion matrix

## 3.7 Processed Video Results of LSTM models and MediaPipe Pose Classification

When classifying disc golf throws using the different LSTM models the prediction is visualized on each frame of the output video. Depending on the model the input for the prediction differs (i.e. (10, 132), (10, 99), (5, 132), (5, 99)). For the first model (3.6.1) following counts:
The model accepts an input of (10, 132) and therefore it is necessary to first read 10 frames and run the MediaPipe Pose to get the Pose landmarks. The 132 values from the pose detection are written to a sequence array that is used to run predictions on. When the sequence array has a length of 10 the model tries to predict the class. When further appending a frame to the array, the array is sliced to only hold the last 10 frames. If the prediction with the highest probability is higher than the threshold (0.5) the prediction is added to the history (see section Test in Realtime at PredictionModel.ipynb from GitHub, appendix 9).
The MediaPipe Pose Classification solution only targets one class at a time. As shown in (see classified videos at appendix 7) the classification works well and can even implement the repetition counter somewhat successfully when targeting a class for which it iterates. However, the classification differs more in confidence depending on the class and test videos. Sometimes the threshold for the counter is not reached and therefore it increments incorrectly for some classes more than others. The classification confidence also drops wrongly for some predictions resulting

in the counter incrementing wrongly.

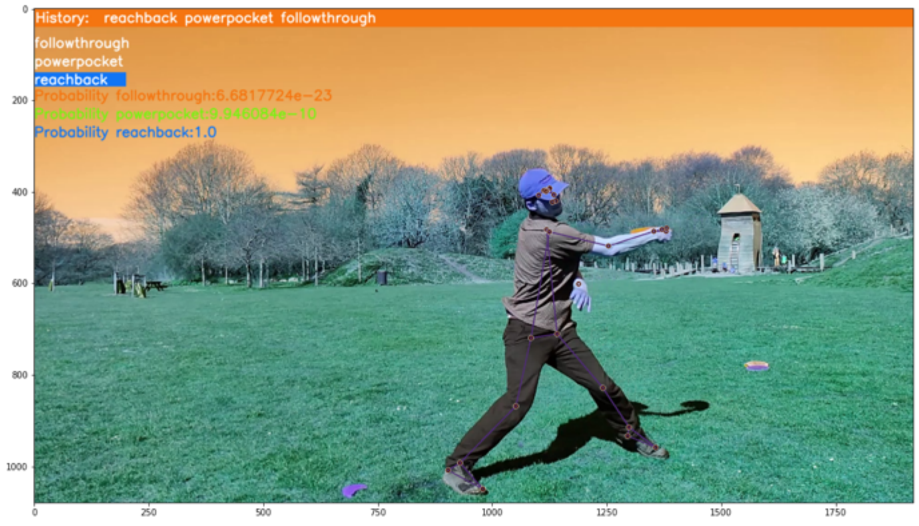
The following results are produced from the different LSTM models and MediaPipe Pose Classification:

| **Name** | **video test final dataset** | **video test initial dataset** |
|---|---|---|
| Model_10_XYZV | LSTM_model_10f_xyzv_ams.mp4 | LSTM_model_10f_xyzv_pros.mp4 |
| Model_10_XYZ | LSTM_model_10f_xyz_ams.mp4 | LSTM_model_10f_xyz_pros.mp4 |
| Model_5_XYZV | LSTM_model_5f_xyzv_ams.mp4 | LSTM_model_5f_xyzv_pros.mp4 |
| Model_5_XYZ | LSTM_model_5f_xyz_ams.mp4 | LSTM_model_5f_xyz_ams.mp4 |
| MediaPipe Pose Classification | Reachback: MP_pose_reachback_ams.mp4<br>Powerpocket: MP_pose_powerpocket_ams.mp4<br>Followthrough: MP_pose_followthrough_ams.mp4 | Reachback: MP_pose_reachback_pros.mp4<br>Powerpocket: MP_pose_powerpocket_pros.mp4<br>Followthrough: MP_pose_followthrough_pros.mp4 |

The results are produced by evaluating ten shots from the final dataset and six shots from the initial dataset (see test videos at appendix 7). The output videos produced using the LSTM models and MediaPipe Pose Classification are analyzed and discussed in more detail in 4.2.

### 3.7.1 LSTM model probability visualization

The test output videos classified with an LSTM model visualize the history of predictions as well as the probability output from the model for each frame. New predictions are added to the history if they breach the threshold of 0.5. The history can hold up to the last eight predictions. The probability of each class is shown to the frame as floating point numbers and as colored rectangles over the predicted class. The width of the rectangle visualizes the probability (see figure 38).



```
History:  reachback powerpocket followthrough
followthrough
powerpocket
reachback
Probability followthrough:6.6817724e-23
Probability powerpocket:9.946084e-10
Probability reachback:1.0
```

Figure 38: LSTM model visualize probability

### 3.7.2 MediaPipe Pose confidence visualization

The test output videos classified with MediaPipe Pose Classification targets one class at a time (three output videos for one test video). A plot is used for visualizing the confidence history of the target class. Each frame is classified against the training dataset (3.4) using k-NN which returns the confidence. If the confidence for the target class goes above 6 the counter is incremented. Once the confidence drops below 4 the counter can increment again for the targeted class.

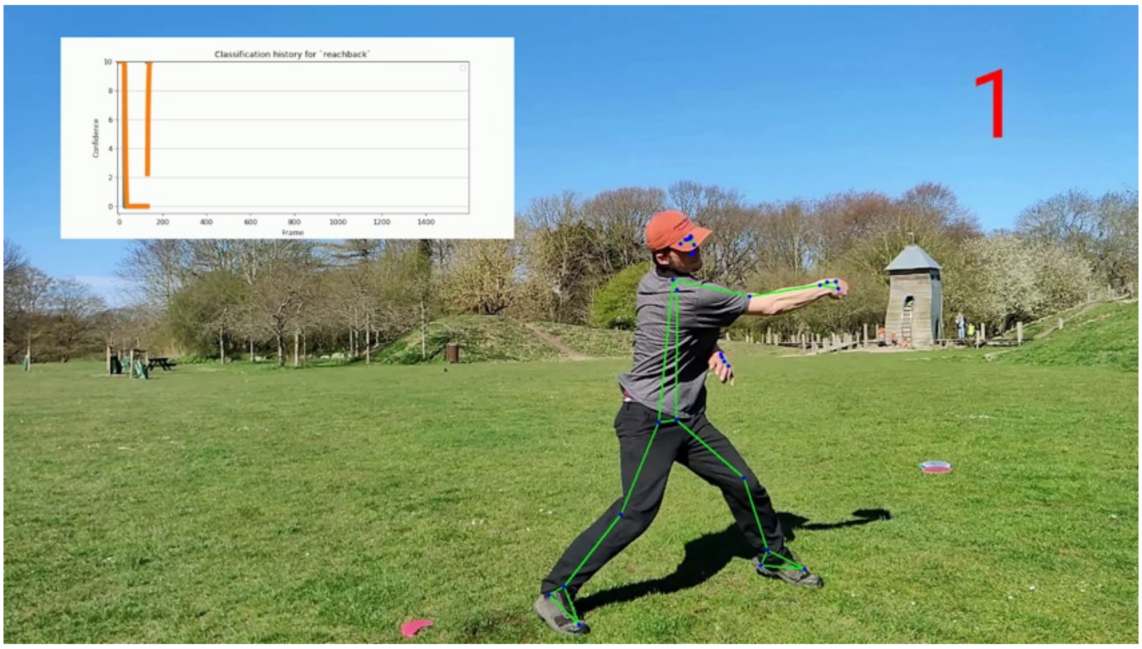


Figure 39: MediaPipe Pose Classification visualize probability

# 4 Analysis & Discussion

## 4.1 Initial Dataset & Final Dataset

As described in Data & Results (3) the difference between the initial dataset and the final dataset was the number of frames per second that was outputted for each video. The initial dataset only has a quarter of the number of frames per second that the final dataset has. At the beginning of the project, the initial dataset was used during data processing to look for clustering for classes defining the disc golf shot. In that process, it was decided that only having 4-7 frames for classifying the powerpocket would be too little to efficiently train an LSTM network. The first dataset was also captured using varying angles and distances that were not notated. This results in the positional data from pose landmarks having variations since the person throwing the shot would be at more varying distances and angles. Therefore the final dataset was produced to avoid the challenges of having too few frames and varying angles and distances. The tradeoff however was that it resulted in moving away from a dataset with professional disc golf players to amateur disc golf players. With amateurs, there is more variation between each shot compared to a professional. And since the player isn't as consistent as a professional the form varies more due to inconsistency and differences in skill. Figure 40 shows that there are clearer similarities between the reachback for professionals than there are for amateurs. By looking at where the chest is facing and the position and height of the hand reaching back (see appendix 9 for comparison between powerpocket and followthrough).

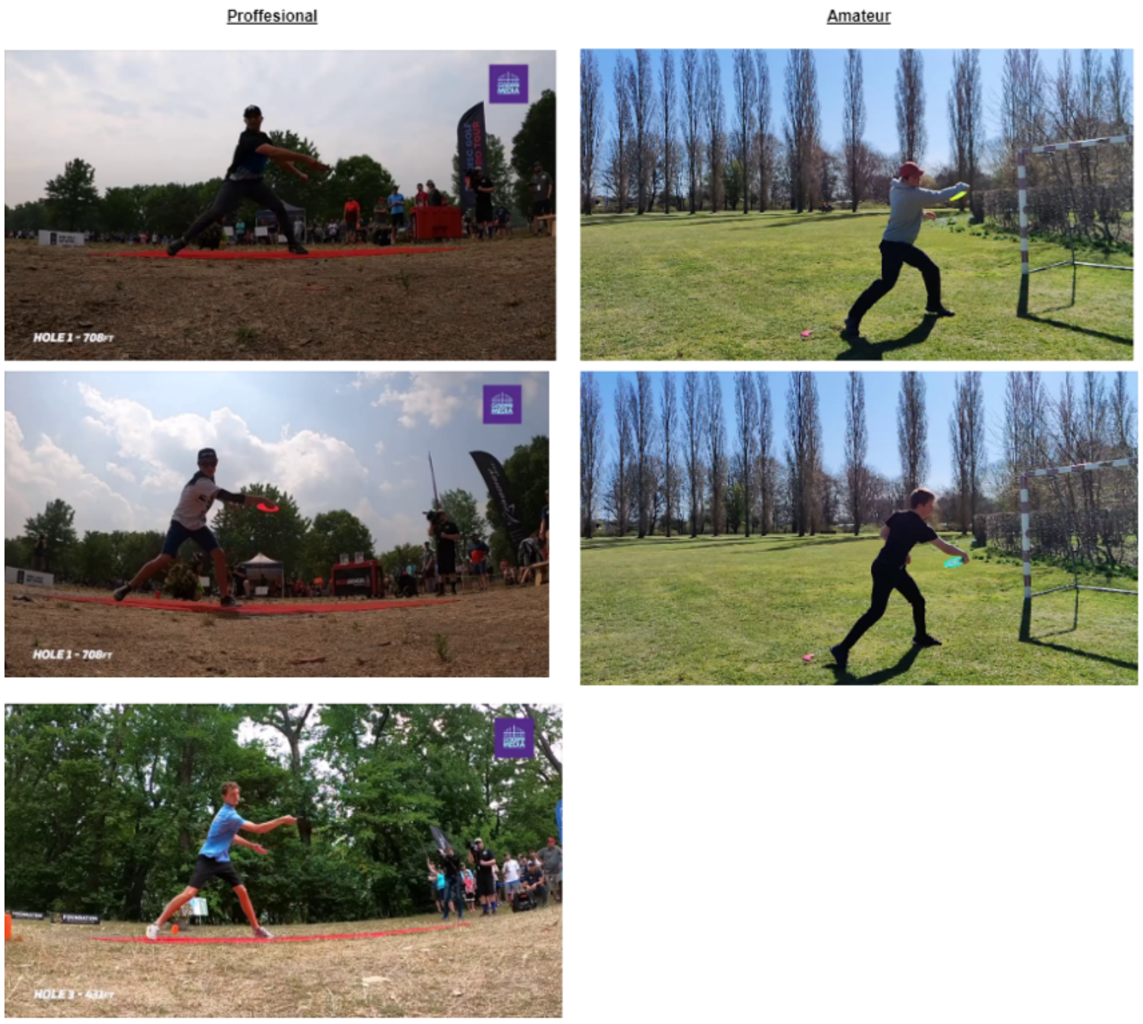


Figure 40: Comparison between reachback of pros and amateurs

When training a neural network the data has a huge impact on whether the network is any good. It is said if you give your network “garbage-in” you will get “garbage-out”. Many amateur sports athletes also coin the phrase ”I’m so garbage” when underperforming and not executing their throws correctly. In the initial dataset (with pro athletes) there is a much higher skill in the athletes performing the throws which means the quality of the throws is good or even great. However, for the final dataset (with two amateurs) there are some shots that are good and some that are okay. So it would be ideal to have a dataset of top-tier professionals recorded under controlled environments (with noted distance between the camera and so on) and with the proper equipment. However, the similarities that exist between the amateurs and the professionals are enough for the LSTM models and the MediaPipe Pose Classification to work on both the initial and the final dataset (as shown by classified videos at appendix 7). Meaning that the LSTM models and MediaPipe Pose Classification solution, which are built using shots from amateurs, also classify shots from professionals well. This could either mean that the amateurs execute the disc golf form very well or the small differences in style don't impact the accuracy of the models very much. However, if the classification was able to analyze more and smaller variables in the form it should have a higher impact. This raises the question what benefits a more complex classifier could provide information about when analyzing the disc golf throw.

## 4.2 Analysis of processed video results

The analysis is based on the results from the different LSTM models and MediaPipe Pose Classification (3.7) when processing shots from the initial and final dataset. Ten shots from the final dataset (amateurs) are compiled into "test_shots_ams.mp4" (see appendix 7 for video) which is used for testing the different LSTM models and MediaPipe Pose Classification on. The LSTM models and MediaPipe Pose Classification is also tested on six shots from the initial dataset (professionals) which are compiled into "test_shot_pros.mp4" (see appendix 7 for video). The shots by the professionals have not been used for training data for the LSTM models or MediaPipe Pose Classification solution. However, the shots from the amateurs are from the final dataset that has been used for training the LSTM models and as training data for the k-NN algorithm that is utilized by the MediaPipe Pose Classification solution.

### 4.2.1 LSTM models

The results of the different LSTM models (3.6) show that the predictions are over 99 percent accurate on test data from the train_test_split method. When visualizing the predictions by processing the recorded shots from 3.7 the LSTM models perform well at dividing the shot into the classes reachback, powerpocket, and followthrough as defined in 2.1. The predictions are more clear in the results produced from the final dataset where every class is correctly added to the history. However, the LSTM models also predict fairly accurately for the initial dataset. The shots from the initial dataset are played at 60 fps (as described in 3.1.1) which shortens the number of frames that define the classes. Since the powerpocket and reachback are represented in fewer frames the LSTM models that predict the class from a sequence of five frames perform better than the models that take a sequence of ten frames. The difference between "LSTM_model_5f_xyzv_pros.mp4" and "LSTM_model_10f_xyzv_pros.mp4" (see videos at appendix 7) is that the reachback is not correctly added to the history of predictions when using the model that takes a sequence of ten frames (see figure 41).

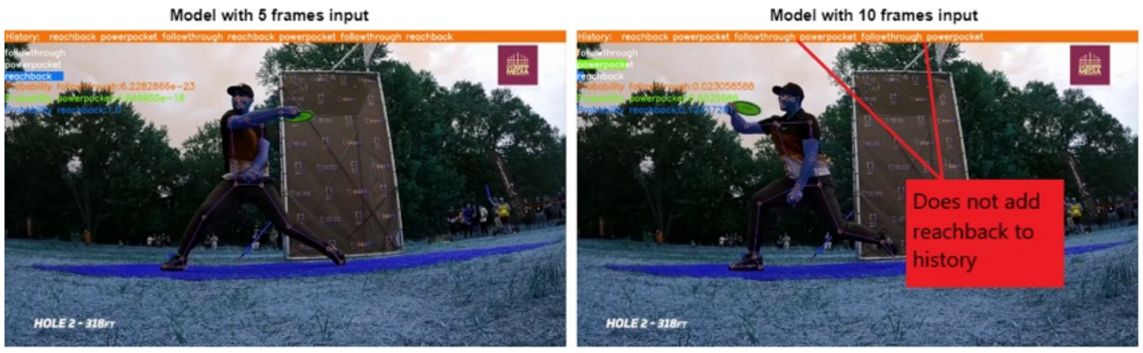

Model with 5 frames input
Model with 10 frames input
History: reachback powerpocket followthrough reachback powerpocket followthrough reachback
History: reachback powerpocket followthrough powerpocket followthrough powerpocket
Does not add reachback to history
HOLE 2 - 318FT
HOLE 2 - 318FT

Figure 41: Comparison between history of "LSTM_model_5f_xyzv_pros.mp4" and "LSTM_model_10f_xyzv_pros.mp4"

Depending on the LSTM model the predictions take an input shape of either ten or five frames. When processing the test videos the model runs predictions on the last sequence of ten or five frames (depending on the model). The train_test_videos/final_dataset (see appendix 8 for video files) consists of the 55 shots divided to each class (reachback, powerpocket, and followthrough). The folder holds 165 .mp4 files where the name of the class is specified in the filename. The sequences of frames is written to "shots_train_v4.csv" (as described in 3.2.2). However, when running predictions on sequences from full shots ("test_shot_ams.mp4" and "test_shot_pros.mp4") the case arrives where the sequence contains frames that should be classified as different classes (see figure 42).

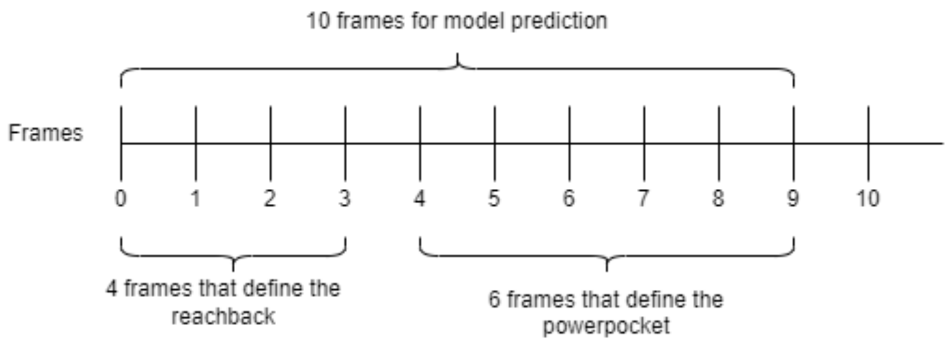
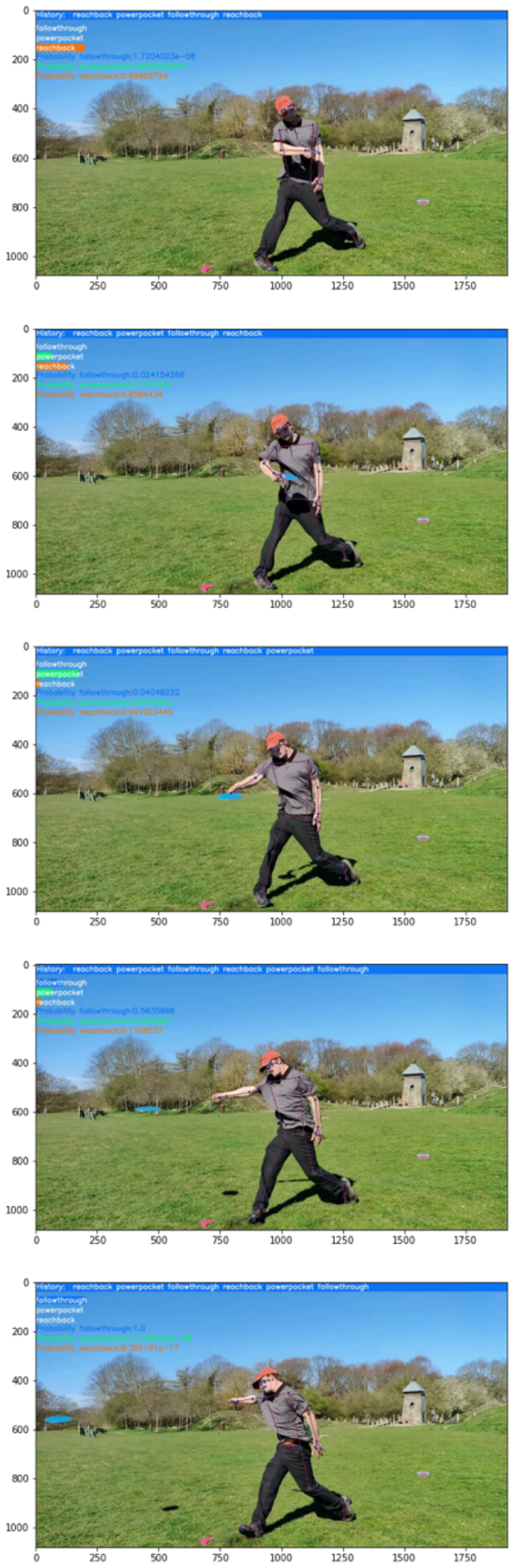

10 frames for model prediction
Frames
0
1
2
3
4
5
6
7
8
9
10
4 frames that define the reachback
6 frames that define the powerpocket

Figure 42: Sequence containing frames of different classes

The LSTM model is not trained with inputs where the sequence defined in figure 42 occurs because of how the final dataset is processed (3.2.2). So the models are trained with sequences where all frames in a sequence point to one class. When analyzing the processed video results (from appendix 7) it shows that the correct class prediction is visualized around ten or five frames late (see figure 43).

Figure 43: Shows that prediction is delayed for first LSTM model (3.6.1)



```
History:  reachback powerpocket followthrough reachback
followthrough
powerpocket
reachback
Probability followthrough:1.7204023e-08
Probability powerpocket:0.0009020284
Probability reachback:0.99909794
```

```
History:  reachback powerpocket followthrough reachback
followthrough
powerpocket
reachback
Probability followthrough:0.024154266
Probability powerpocket:0.3174024
Probability reachback:0.6584434
```

```
History:  reachback powerpocket followthrough reachback powerpocket
followthrough
powerpocket
reachback
Probability followthrough:0.04048222
Probability powerpocket:0.8675943
Probability reachback:0.09192344
```

```
History:  reachback powerpocket followthrough reachback powerpocket followthrough
followthrough
powerpocket
reachback
Probability followthrough:0.5635966
Probability powerpocket:0.3255997
Probability reachback:0.1108037
```

```
History:  reachback powerpocket followthrough reachback powerpocket followthrough
followthrough
powerpocket
reachback
Probability followthrough:1.0
Probability powerpocket:1.1298333e-08
Probability reachback:8.355181e-17
```

### 4.2.2 MediaPipe Pose Classification

The results from processing "test_shots_ams.mp4" from the final dataset using MediaPipe Pose Classification varies depending on the target class. When targeting the reachback the classification and repetition counter works as intended for all 10 shots. The confidence spikes to 10 when the body is in the reachback form and drops when leaving that part of the form. In a few of the frames that should define the followthrough the confidence for reachback falsely rises to around two (see figure 44 or appendix 7 "MP_pose_reachback_ams.mp4" for more detail).

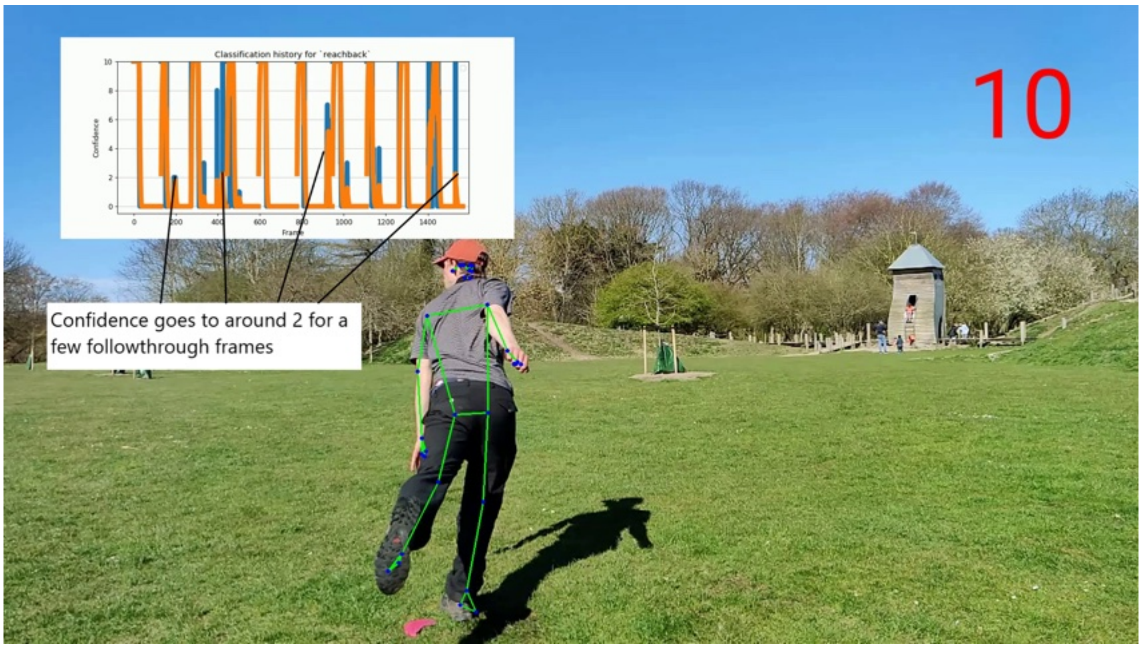

Classification history for `reachback`
Confidence
Frame
10
Confidence goes to around 2 for a few followthrough frames

Figure 44: Reachback history: for MediaPipe Pose Classification (see 7)

When targeting the powerpocket MediaPipe Pose Classification predicts very well the start of the powerpocket as well as when exiting the powerpocket. However for the second shot in "test_shots_ams.mp4" the Pose Classification wrongly predicts a couple of frames from the followthrough as powerpocket (see figure 45).

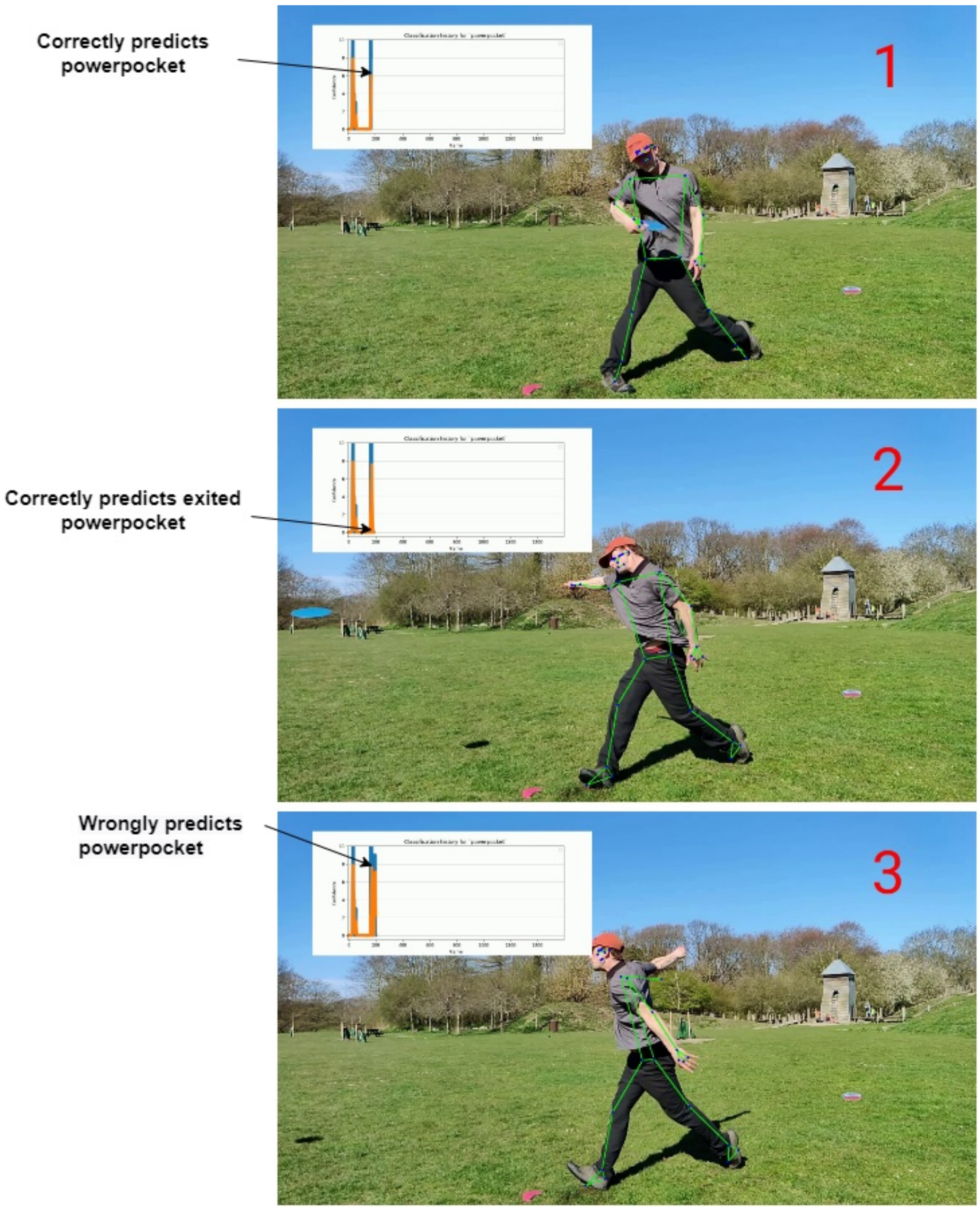

Correctly predicts powerpocket
1
Correctly predicts exited powerpocket
2
Wrongly predicts powerpocket
3

Figure 45: Powerpocket misprediction

When targeting the followthrough MediaPipe Pose Classification struggles more with classifying frames that are a part of the followthrough as followthrough. The confidence drops and rises for some of the shots while in the followthrough motion. The classifier does well at correctly classifying the start and end of the followthrough. However, the frames in the middle of the followthrough are where it struggles for some of the shots in "test_shots_ams.mp4". Since the classification struggles with the frames in the middle the confidence drops and rises causing the counter to incorrectly count 14 followthroughs (see figure 46) when there should only be 9 (only 9, since the 10th followthrough never is exited because the video ends).

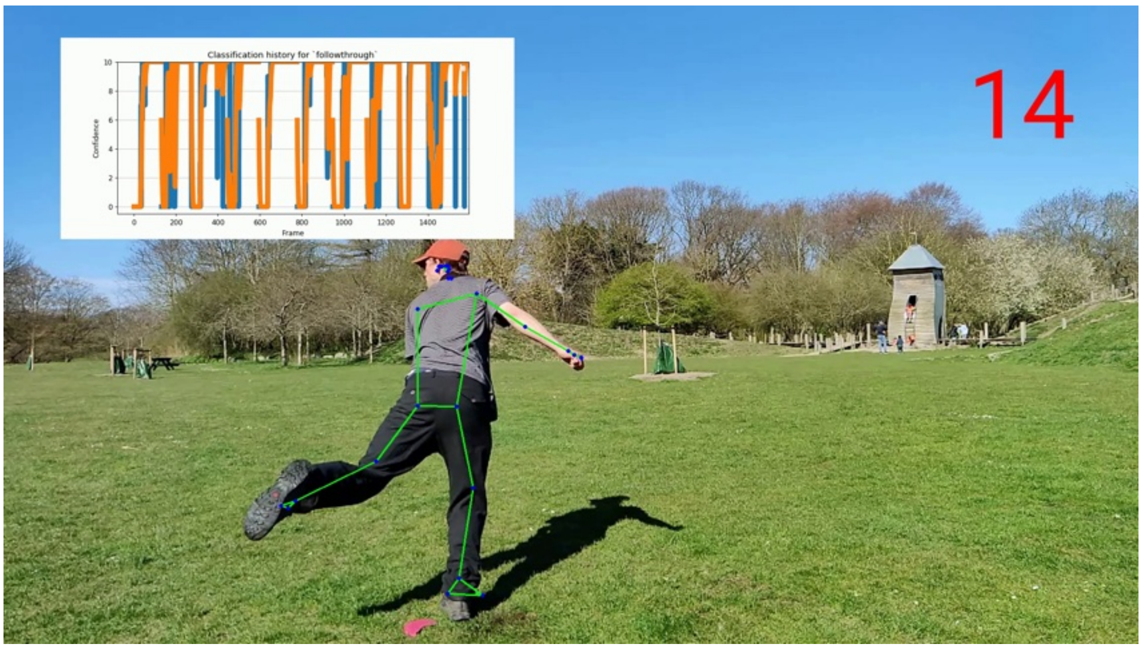


Figure 46: Followthrough history for MediaPipe Pose Classification

When processing the six shots in "test_shots_pros.mp4" the Pose Classification actually works better for classifying the followthrough (see "MP_pose_followthrough_pros.mp4" and "MP_pose_followthrough_ams.mp4 under appendix 7) compared to classifying the followthrough on "test_shots_ams.mp4". Despite the k-NN classifier using training data that is collected from processing amateur shots and not professionals. For "test_shots_pros.mp4" the confidence graph depicts confidently the hole followthrough for all shots except a bit of the first shot in the history. The counter also increments correctly as it shows five at the end of the video. The test video has six shots however the counter does not increment before the targeted class is exited. Since the video stops while in the last followthrough the counter stays at five instead of incrementing to six (see figure 47).

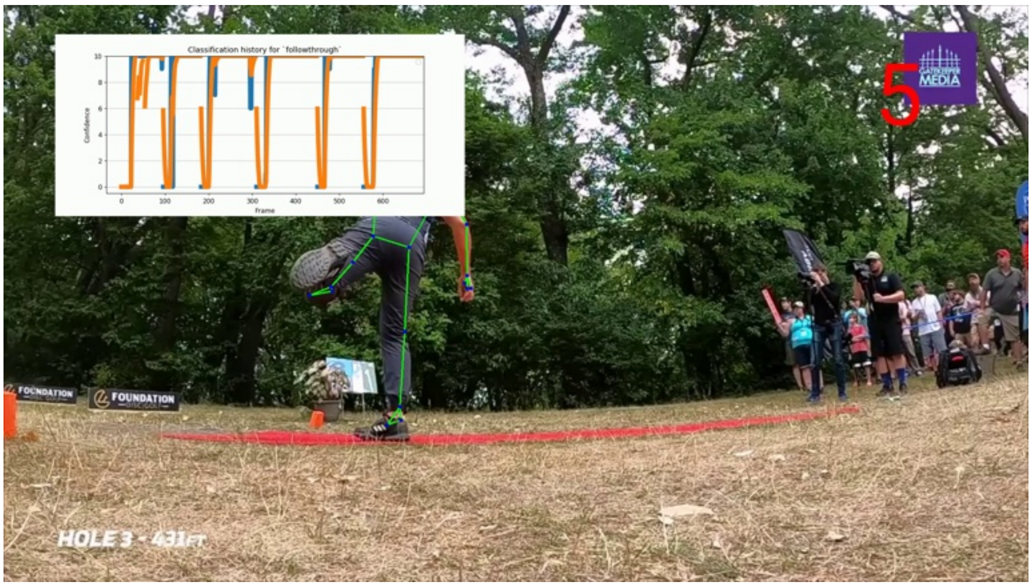


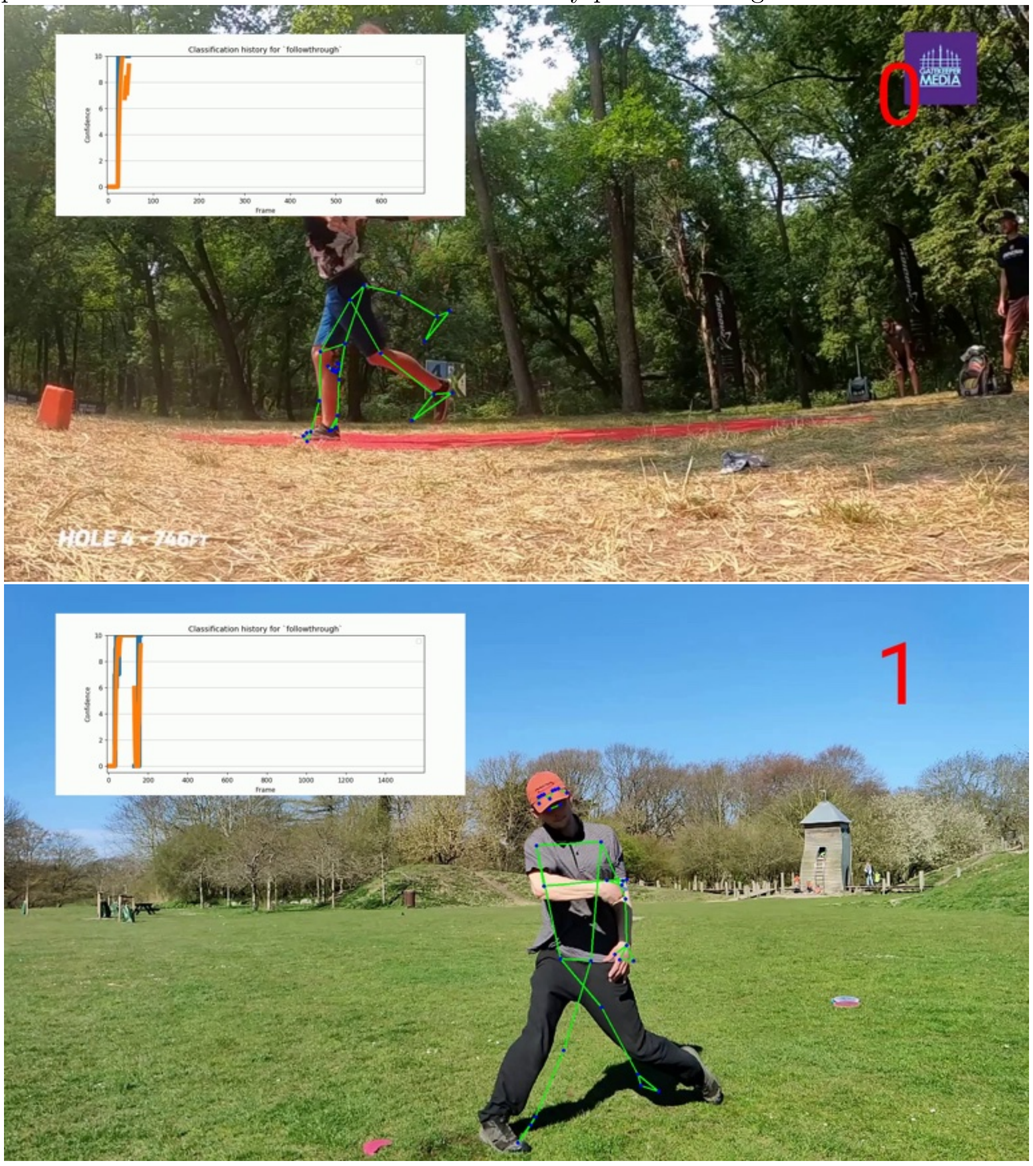
Figure 47: Followthrough history for MediaPipe Pose Classification

On the first of the six shots in ”test_shots_pros.mp4” the confidence drops and rises. This is due to MediaPipe Pose predicting the pose landmarks incorrectly (see figure 48). The pose landmark predictions are not 100 percent accurate and that impacts the Pose Classification.

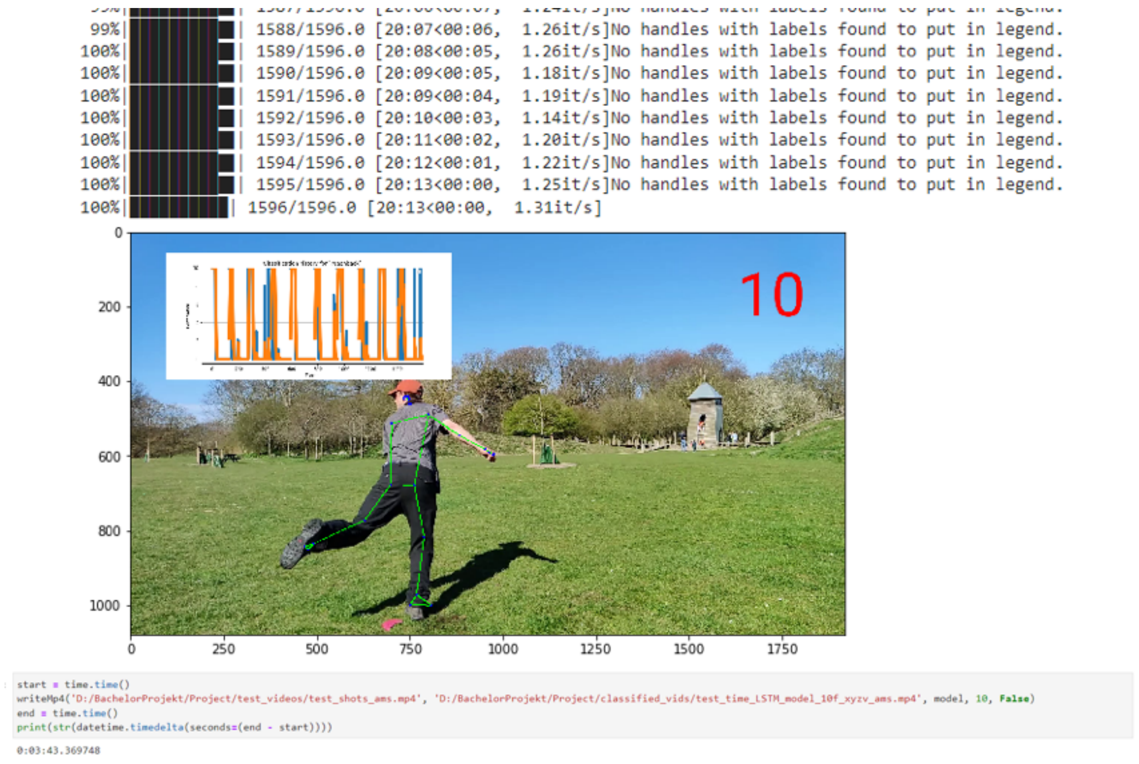
Figure 48: Pose landmarks inaccurate

1st picture: Pose landmarks drawn does not read upper body and misreads the lower body

2nd picture: Pose landmarks does not match the body pose as the legs are not crossed

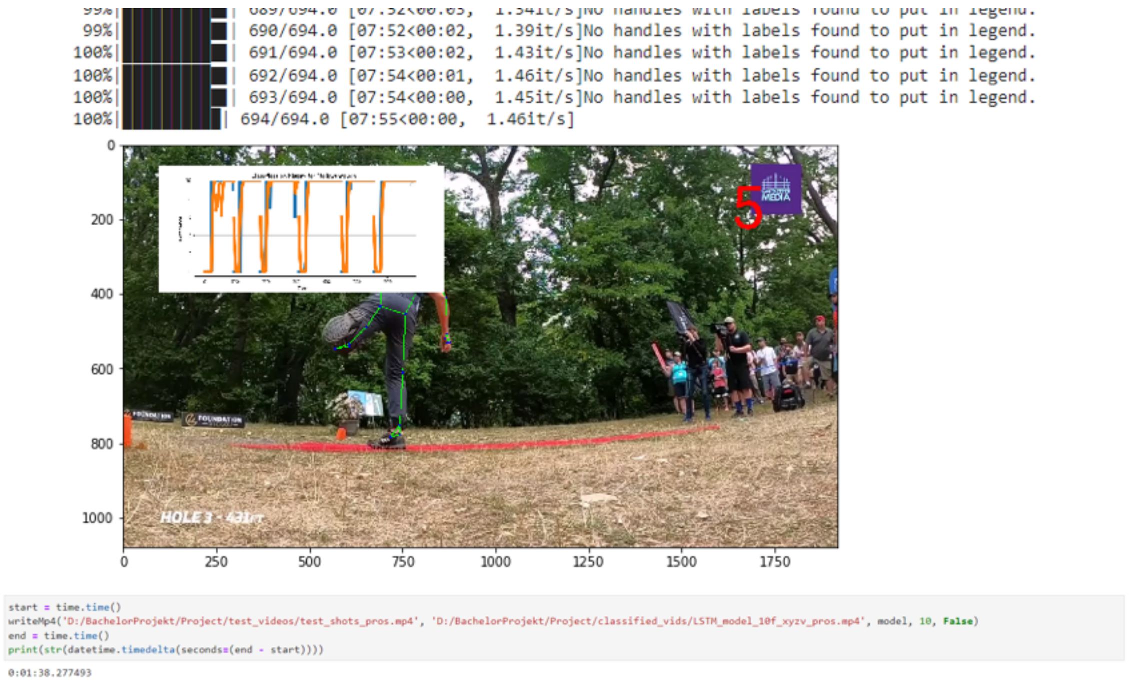

Classification history for `followthrough`
Confidence
Frame
0
GATEKEEPER MEDIA
HOLE 4 - 746FT
Classification history for `followthrough`
Confidence
Frame
1

The time which it takes to run the classifiers on "test_shots_ams.mp4" and "test_shots_pros.mp4" depends on the classifier used. When using the first LSTM model (3.6.1) the test videos are processed in 3'43" (for "test_shots_ams.mp4") and 1'38" (for "test_shots_pros.mp4"). The solution using MediaPipe Pose Classification processed the same two videos in 20'13" (for "test_shots_ams.mp4") and 7'55" (for "test_shots_pros.mp4") (see figure 49 and 50). That shows that the classifiers using a trained LSTM model performs over five times faster when processing the test videos compared to the classifier using MediaPipe Pose Classification.


```
 99%|██████████▉| 1588/1596.0 [20:07<00:06,  1.26it/s]No handles with labels found to put in legend.
100%|██████████▉| 1589/1596.0 [20:08<00:05,  1.26it/s]No handles with labels found to put in legend.
100%|██████████▉| 1590/1596.0 [20:09<00:05,  1.18it/s]No handles with labels found to put in legend.
100%|██████████▉| 1591/1596.0 [20:09<00:04,  1.19it/s]No handles with labels found to put in legend.
100%|██████████▉| 1592/1596.0 [20:10<00:03,  1.14it/s]No handles with labels found to put in legend.
100%|██████████▉| 1593/1596.0 [20:11<00:02,  1.20it/s]No handles with labels found to put in legend.
100%|██████████▉| 1594/1596.0 [20:12<00:01,  1.22it/s]No handles with labels found to put in legend.
100%|██████████▉| 1595/1596.0 [20:13<00:00,  1.25it/s]No handles with labels found to put in legend.
100%|██████████| 1596/1596.0 [20:13<00:00,  1.31it/s]
```

```
start = time.time()
writeMp4('D:/BachelorProjekt/Project/test_videos/test_shots_ams.mp4', 'D:/BachelorProjekt/Project/classified_vids/test_time_LSTM_model_10f_xyzv_ams.mp4', model, 10, False)
end = time.time()
print(str(datetime.timedelta(seconds=(end - start))))
```
```
0:03:43.369748
```

Figure 49: Performance of LSTM model and MediaPipe Pose Classification on "test_shots_ams.mp4"



```
 99%|█████████▉| 689/694.0 [07:52<00:03,  1.34it/s]No handles with labels found to put in legend.
 99%|█████████▉| 690/694.0 [07:52<00:02,  1.39it/s]No handles with labels found to put in legend.
100%|█████████▉| 691/694.0 [07:53<00:02,  1.43it/s]No handles with labels found to put in legend.
100%|█████████▉| 692/694.0 [07:54<00:01,  1.46it/s]No handles with labels found to put in legend.
100%|█████████▉| 693/694.0 [07:54<00:00,  1.45it/s]No handles with labels found to put in legend.
100%|██████████| 694/694.0 [07:55<00:00,  1.46it/s]
```

```
start = time.time()
writeMp4('D:/BachelorProjekt/Project/test_videos/test_shots_pros.mp4', 'D:/BachelorProjekt/Project/classified_vids/LSTM_model_10f_xyzv_pros.mp4', model, 10, False)
end = time.time()
print(str(datetime.timedelta(seconds=(end - start))))
```

```
0:01:38.277493
```

Figure 50: Performance of LSTM model and MediaPipe Pose Classification on "test_shots_pros.mp4"

## 4.3 MediaPipe Pose Accuracy

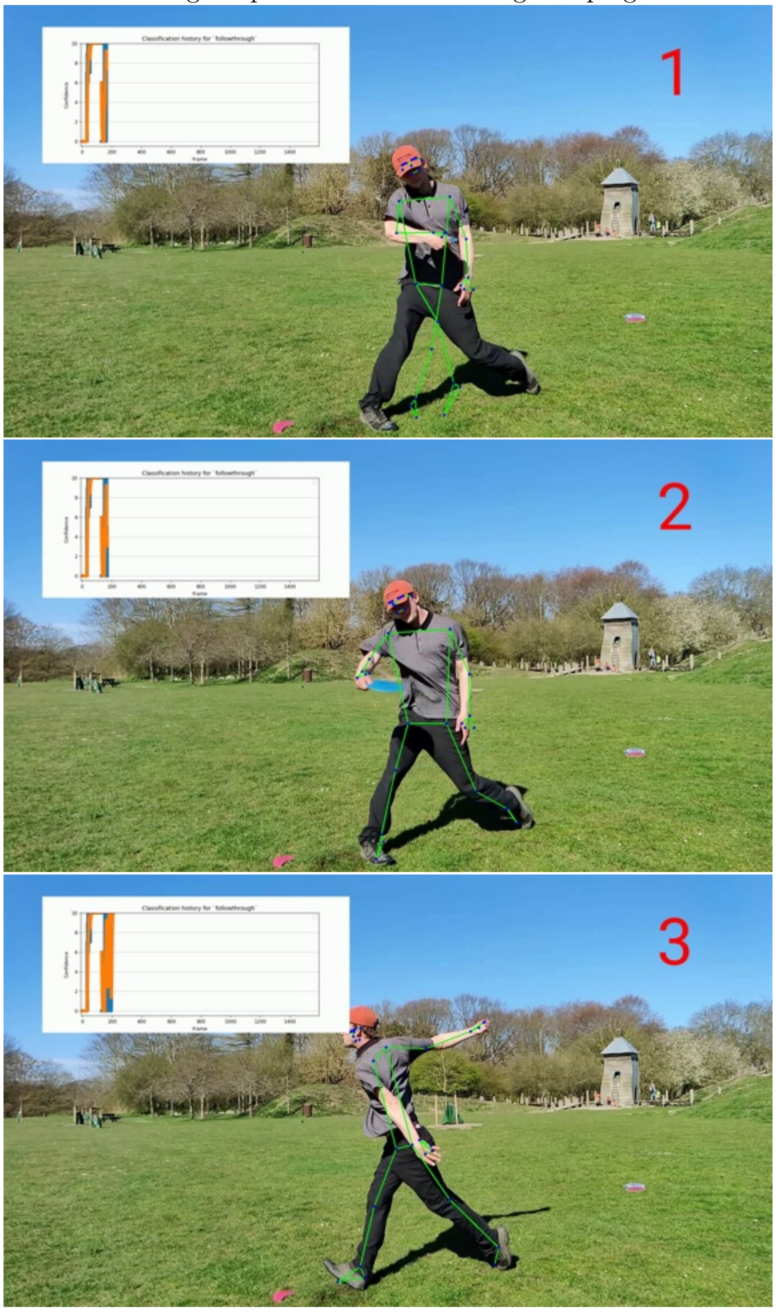
MediaPipe Pose estimation is not 100 percent accurate when predicting pose landmarks. That affects the dataset, the LSTM models, and MediaPipe Pose Classification. The documentation defines the pose estimation quality as around 95 percent accurate on three validation datasets (Yoga, Dance, and HIIT)[4]. Since throwing a disc is a similar motion to body poses represented in the validation datasets (mostly HIIT dataset) the accuracy of the body poses derived from the initial and final dataset can be assumed to be around 95 percent. Therefore the training data contains entries where pose landmarks do not match the body position entirely. That impacts the LSTM models and the MediaPipe Pose Classification since the training data has not been filtered to remove or correct pose landmarks that are wrongly predicted. For the LSTM models, the impact is less than for the classifier using MediaPipe Pose Classification because the LSTM uses temporal information and predicts on inputs of more than one frame. So the weight of MediaPipe Pose accuracy is less than when predicting from one frame. MediaPipe Pose Classification predicts each frame individually using k-NN algorithm. If the pose landmarks read from the frame are somewhat inaccurate this impacts the prediction when performing k-NN since the pose landmarks read could be nearer to the wrong classes. When that happens the confidence spikes up and down causing prediction jittering (see figure 51).

Figure 51: Prediction jittering due to wrong pose landmarks.

1st picture: Pose landmarks show legs crossed which is not the case. Confidence is up.

2nd picture: Correct pose landmarks. Confidence goes down again causing increment.

3rd picture: Actual followthrough is predicted. Confidence goes up again incrementing counter



Since there are entries of inaccurate pose landmarks in the training data this can also impact the prediction of correct pose landmarks. However, since k-NN is invoked twice with different distance metrics the impact is minimal since there should be a nearer neighbor in the training dataset. Therefore, when pose predictions are accurate the MediaPipe Pose Classification also predicts more accurately. When pose predictions are wrong the k-nearest neighbor is a result of a wrong body pose.

## 4.4 MediaPipe Pose Classification and LSTM network

The LSTM model is trained and tested with overall 13077 rows of data. The 13077 rows of data however only represent every frame from 55 shots. A neural network trained on 55 shots is not a lot and the model could be improved upon if fed with more data. The distribution of the classes that are represented in the dataset is unbalanced due to the nature of the disc golf throw. The reachback and powerpocket occur in fewer frames than the followthrough resulting in the training set being somewhat unbalanced (as shown in figure 16). Therefore the results are unoptimized for the unbalanced classes and perform better when predicting followthroughs as shown by the results of the LSTM networks (3.6).

The MediaPipe Pose Classification and LSTM network both have different advantages and disadvantages when it comes to performance and training. One of the differences between the two is that MediaPipe Pose Classification uses k-NN for classification which requires zero training time. This results in slower performance at evaluation time since it must evaluate each frame against the training set with k-NN. On the other hand, the training data for the LSTM network is obsolete once the network is trained and is no longer needed to make new predictions. This results in better performance which is useful for evaluating shots in a shorter amount of time. However, the k-NN classifier performs very well on smaller datasets of a few hundred samples per class for each terminal state [5]. The training data used for the k-NN classifier has 1793 samples for the reachback, 705 samples for the powerpocket, and 12064 samples for the followthrough (as shown in 3.4). However, the three classes that define the backhand disc golf throw are defined as a sequence of frames and not as much a terminal state. Therefore the training data also consists of more samples since each class has the whole sequence of frames that define the class. The reachback and powerpocket is a much shorter motion than the followthrough and that is why there are fewer samples for those classes.

## 4.5 Quality of form and throw

When analyzing disc golf form using the solutions provided by this study it is hard to provide information or score about the quality of the throw. The solutions provide the tool to classify the reachback, powerpocket, and followthrough, which shows that body pose detection can prove useful for analyzing and reviewing disc golf form. However, it would be possible to calculate the angles between different pose landmarks to check if the form for each class fulfilled the ideal form as described in 2.1 and shown in figures 1, 3, and 5. Using the classifiers it would be possible to select specific frames from each class and compute information that would score the quality of the form from defined parameters. That could prove useful for training tips and comments for improving form technique. Many players try to replicate their form to match that of a specific professional player. Models could be trained and analyzed such that they enforced form based on a single player. It is important to distinguish between the form and the shot quality. Being able to score the quality of disc golf form is not the same as predicting if the shot is good since factors like disc selection, wind, or release angle also impact the quality of the shot. A good shot is often measured by its end result whereas good form is distinguished from the shot itself.

# 5 Conclusion

The technique that is required for performing a backhand disc golf throw is defined by many factors and the form required for executing that technique can be divided into different classes. Those classes make up the motion of performing a backhand disc golf throw and can be analyzed using ML solutions to retrieve the positional data that represents the body poses. When processing the initial dataset the study showed that the reachback and powerpocket were represented in only a few frames because of frames per second. The final dataset resolved the problem since it was shot with more frames per second. The data extracted from the final dataset showed that the distribution of the different classes was uneven which affected the training of a neural network since the train and test dataset had an overrepresentation of followthrough entries. The overrepresentation occurred due to the followthrough consisting of a larger sequence of frames compared to the reachback and powerpocket. This study establishes that the time series representing different phases of disc golf form theory showed clustering when analyzed using dimensionality reduction with Principal Component Analysis. The phases in the backhand disc golf throw can be classified using either an LSTM network or a k-NN solution from MediaPipe Pose Classification because of the significant clustering for each class. Between the different LSTM models, there was no noticeable difference in accuracy when shrinking the window from 10 to 5 frames or only predicting with x, y, and z values from pose landmarks. For all the LSTM models the accuracy on the test data was above 99 percent where it performed best on followthroughs since it was overrepresented in the train and test data. By testing the models on shots from the initial and final dataset it was established that using an LSTM model with a window of 5 frames performed better on shots from the initial dataset because of the number of frames per second. The study also shows a MediaPipe Pose Classification solution using the same training data as for the LSTM models. By comparing the processed test videos using different LSTM models and MediaPipe Pose Classification the study analyzed and discussed the differences between the two solutions. The study concludes that the LSTM models perform faster predictions and are more accurate because they use temporal information for prediction. However, MediaPipe Pose Classification performs well on smaller amounts of data and requires no training since it uses k-NN for classification. The study showed that accuracy for pose landmarks had more impact on the MediaPipe Pose Classification solution which resulted in prediction jittering in test videos from the initial and final dataset. It can be concluded that analyzing the form technique of a disc golf throw using ML solutions can classify a throw into the classes reachback, powerpocket, and followthrough. The disc golf throw has many aspects and factors which define the quality of the throw and quality of the form. The LSTM models and MediaPipe Pose Classification can provide insight on the body poses that define the form. Disc golf form consists of classes that when executed properly is what allow a player to improve and stay consistent. Therefore analysis of disc golf form is essential and there are many more complex factors that could be analyzed further using ML solutions.

# 6 Reflections

When performing a backhand disc golf shot the form can be reviewed and analyzed in great detail. It is shown that collecting data from the body pose is great for classifying and dividing the shot classes that define different parts of the form. However, there are many small details that impact the quality of the shot which are more complex to analyze from the data available from the body pose. This raises the question if automated form critique can work as a training tool that gives a player an advantage when training form technique and what are the risks of relying on big data when used for training purposes. The risks involved with using big data is the model is solely based on the data set. Many players enforce the same techniques in their form that are represented by the body poses in the data set. One could argue that the model reinforces a specific style of throwing form such that form in general would never evolve into something completely different. The form that is defined today as ideal (based on the best professionals) is reinforced as a byproduct of using big data of that form. However, when analyzing performance in sports the best professional players are often used as a benchmark that defines great execution, and therefore they are the most interesting to analyze. There are many factors that impact the quality of throwing a good disc golf shot (2.1.4) and therefore a lot of potential for further analysis and data collection. Whereas this project solely focuses on reviewing and analyzing the form there is also great potential in analyzing the flight and outcome of a disc golf throw. Combining the two with a large data set of professionals could greatly increase the complexity and ability to analyze the performance of the throw and classify a shot as good or bad. Or even having the ability to provide tips and tricks for amateurs and other players who are trying to improve their skill. The sport of disc golf is experiencing rapid growth in recent years and there is a lot of potential for utilizing ML solutions for analyzing and improving the experience of disc golf as a whole.

# 7 Appendix A

Appendix_B.zip has to folders. One folder holds all the output videos produced with the different LSTM models and MediaPipe Pose Classification solution. The other folder has the two input videos that were used test the models on. Unzip Appendix_B.zip to get the following:

## Folder structure

```
+---classified_videos
|       LSTM_model_10f_xyzv_ams.mp4
|       LSTM_model_10f_xyzv_pros.mp4
|       LSTM_model_10f_xyz_ams.mp4
|       LSTM_model_10f_xyz_pros.mp4
|       LSTM_model_5f_xyzv_ams.mp4
|       LSTM_model_5f_xyzv_ams_v2.mp4
|       LSTM_model_5f_xyzv_pros.mp4
|       LSTM_model_5f_xyzv_pros_v2.mp4
|       LSTM_model_5f_xyz_ams.mp4
|       LSTM_model_5f_xyz_pros.mp4
|       MP_pose_followthrough_ams.mp4
|       MP_pose_followthrough_pros.mp4
|       MP_pose_powerpocket_ams.mp4
|       MP_pose_powerpocket_pros.mp4
|       MP_pose_reachback_ams.mp4
|       MP_pose_reachback_pros.mp4
|
\---unclassified_videos
        test_shots_ams.mp4
        test_shots_pros.mp4
```

# 8 Appendix B

Appendix_C.zip has two main folders. One folder has the csv files used at the end of this study. The other holds all the videos which the csv files was created using. Unzip Appendix_C.zip to get the following:

## Folder structure

```
 +---csv_files
|   +---early_data
|   |       eagle_shot2.csv
|   |       eagle_shot2_followthrough.csv
|   |       eagle_shot2_powerpocket.csv
|   |       eagle_shot2_reachback.csv
|   |       eagle_shot3.csv
|   |       eagle_shot3_followthrough.csv
|   |       eagle_shot3_powerpocket.csv
|   |       eagle_shot3_reachback.csv
|   |       heimberg_shot1.csv
|   |       heimberg_shot1_followthrough.csv
|   |       heimberg_shot1_powerpocket.csv
|   |       heimberg_shot1_reachback.csv
|   |       heimberg_shot3.csv
|   |       heimberg_shot3_followthrough.csv
|   |       heimberg_shot3_powerpocket.csv
|   |       heimberg_shot3_reachback.csv
|   |       mcbeth_shot1.csv
|   |       mcbeth_shot1_followthrough.csv
|   |       mcbeth_shot1_powerpocket.csv
|   |       mcbeth_shot1_reachback.csv
|   |
|   +---mediapipe_pose_classification_data
|   |       followthrough.csv
|   |       powerpocket.csv
|   |       reachback.csv
|   |
|   +---pca_png
|   |       pca3D_5_shots_window(3,1).png
|   |       pca_5_shots.png
|   |       pca_5_shots_window(4,1).png
|   |
|   \---train_test_data
|           shots_train_v4.csv
|
\---train_test_videos
    +---final_dataset
    |       VID_20220419_100422_followthrough.mp4
    |       VID_20220419_100422_powerpocket.mp4
    |       VID_20220419_100422_reachback.mp4
    |       VID_20220419_100439_followthrough.mp4
    |       VID_20220419_100439_powerpocket.mp4
    |       VID_20220419_100439_reachback.mp4
    |       VID_20220419_100449_followthrough.mp4
    |       VID_20220419_100449_powerpocket.mp4
    |       VID_20220419_100449_reachback.mp4
    |       VID_20220419_100506_followthrough.mp4
    |       VID_20220419_100506_powerpocket.mp4
    |       VID_20220419_100506_reachback.mp4
    |       VID_20220419_100518_followthrough.mp4
    |       VID_20220419_100518_powerpocket.mp4
    |       VID_20220419_100518_reachback.mp4
    |       VID_20220419_100552_followthrough.mp4
    |       VID_20220419_100552_powerpocket.mp4
    |       VID_20220419_100552_reachback.mp4
    |       VID_20220419_100600_followthrough.mp4
    |       VID_20220419_100600_powerpocket.mp4
    |       VID_20220419_100600_reachback.mp4
    |       VID_20220419_100609_followthrough.mp4
    |       VID_20220419_100609_powerpocket.mp4
    |          VID_20220419_100609_reachback.mp4
    |          VID_20220419_100618_followthrough.mp4
    |          VID_20220419_100618_powerpocket.mp4
    |          VID_20220419_100618_reachback.mp4
    |          VID_20220419_100626_followthrough.mp4
    |          VID_20220419_100626_powerpocket.mp4
    |          VID_20220419_100626_reachback.mp4
    |          VID_20220419_100731_followthrough.mp4
    |          VID_20220419_100731_powerpocket.mp4
    |          VID_20220419_100731_reachback.mp4
    |          VID_20220419_100745_followthrough.mp4
    |          VID_20220419_100745_powerpocket.mp4
    |          VID_20220419_100745_reachback.mp4
    |          VID_20220419_100801_followthrough.mp4
    |          VID_20220419_100801_powerpocket.mp4
    |          VID_20220419_100801_reachback.mp4
    |          VID_20220419_100817_followthrough.mp4
    |          VID_20220419_100817_powerpocket.mp4
    |          VID_20220419_100817_reachback.mp4
    |          VID_20220419_100833_followthrough.mp4
    |          VID_20220419_100833_powerpocket.mp4
    |          VID_20220419_100833_reachback.mp4
    |          VID_20220419_100901_followthrough.mp4
    |          VID_20220419_100901_powerpocket.mp4
    |          VID_20220419_100901_reachback.mp4
    |          VID_20220419_100910_followthrough.mp4
    |          VID_20220419_100910_powerpocket.mp4
    |          VID_20220419_100910_reachback.mp4
    |          VID_20220419_100919_followthrough.mp4
    |          VID_20220419_100919_powerpocket.mp4
    |          VID_20220419_100919_reachback.mp4
    |          VID_20220419_100928_followthrough.mp4
    |          VID_20220419_100928_powerpocket.mp4
    |          VID_20220419_100928_reachback.mp4
    |          VID_20220419_100936_followthrough.mp4
    |          VID_20220419_100936_powerpocket.mp4
    |          VID_20220419_100936_reachback.mp4
    |          VID_20220419_101046_followthrough.mp4
    |          VID_20220419_101046_powerpocket.mp4
    |          VID_20220419_101046_reachback.mp4
    |          VID_20220419_101058_followthrough.mp4
    |          VID_20220419_101058_powerpocket.mp4
    |          VID_20220419_101058_reachback.mp4
    |          VID_20220419_101116_followthrough.mp4
    |          VID_20220419_101116_powerpocket.mp4
    |          VID_20220419_101116_reachback.mp4
    |          VID_20220419_101132_followthrough.mp4
    |          VID_20220419_101132_powerpocket.mp4
    |          VID_20220419_101132_reachback.mp4
    |          VID_20220419_101149_followthrough.mp4
    |          VID_20220419_101149_powerpocket.mp4
    |          VID_20220419_101149_reachback.mp4
    |          VID_20220419_101215_followthrough.mp4
    |          VID_20220419_101215_powerpocket.mp4
    |          VID_20220419_101215_reachback.mp4
    |          VID_20220419_101224_followthrough.mp4
    |          VID_20220419_101224_powerpocket.mp4
    |          VID_20220419_101224_reachback.mp4
    |          VID_20220419_101233_followthrough.mp4
    |          VID_20220419_101233_powerpocket.mp4
    |          VID_20220419_101233_reachback.mp4
    |          VID_20220419_101244_followthrough.mp4
    |          VID_20220419_101244_powerpocket.mp4
    |          VID_20220419_101244_reachback.mp4
    |          VID_20220419_101254_followthrough.mp4
    |          VID_20220419_101254_powerpocket.mp4
    |          VID_20220419_101254_reachback.mp4
    |          VID_20220419_101412_followthrough.mp4
    |          VID_20220419_101412_powerpocket.mp4
    |          VID_20220419_101412_reachback.mp4
    |          VID_20220419_101429_followthrough.mp4
    |          VID_20220419_101429_powerpocket.mp4
    |          VID_20220419_101429_reachback.mp4
    |          VID_20220419_101458_followthrough.mp4
    |       VID_20220419_101458_powerpocket.mp4
    |       VID_20220419_101458_reachback.mp4
    |       VID_20220419_101513_followthrough.mp4
    |       VID_20220419_101513_powerpocket.mp4
    |       VID_20220419_101513_reachback.mp4
    |       VID_20220419_101531_followthrough.mp4
    |       VID_20220419_101531_powerpocket.mp4
    |       VID_20220419_101531_reachback.mp4
    |       VID_20220419_102214_followthrough.mp4
    |       VID_20220419_102214_powerpocket.mp4
    |       VID_20220419_102214_reachback.mp4
    |       VID_20220419_102229_followthrough.mp4
    |       VID_20220419_102229_powerpocket.mp4
    |       VID_20220419_102229_reachback.mp4
    |       VID_20220419_102240_followthrough.mp4
    |       VID_20220419_102240_powerpocket.mp4
    |       VID_20220419_102240_reachback.mp4
    |       VID_20220419_102311_followthrough.mp4
    |       VID_20220419_102311_powerpocket.mp4
    |       VID_20220419_102311_reachback.mp4
    |       VID_20220419_102323_followthrough.mp4
    |       VID_20220419_102323_powerpocket.mp4
    |       VID_20220419_102323_reachback.mp4
    |       VID_20220419_102344_followthrough.mp4
    |       VID_20220419_102344_powerpocket.mp4
    |       VID_20220419_102344_reachback.mp4
    |       VID_20220419_102354_followthrough.mp4
    |       VID_20220419_102354_powerpocket.mp4
    |       VID_20220419_102354_reachback.mp4
    |       VID_20220419_102403_followthrough.mp4
    |       VID_20220419_102403_powerpocket.mp4
    |       VID_20220419_102403_reachback.mp4
    |       VID_20220419_102413_followthrough.mp4
    |       VID_20220419_102413_powerpocket.mp4
    |       VID_20220419_102413_reachback.mp4
    |       VID_20220419_102424_followthrough.mp4
    |       VID_20220419_102424_powerpocket.mp4
    |       VID_20220419_102424_reachback.mp4
    |       VID_20220419_102448_followthrough.mp4
    |       VID_20220419_102448_powerpocket.mp4
    |       VID_20220419_102448_reachback.mp4
    |       VID_20220419_102506_followthrough.mp4
    |       VID_20220419_102506_powerpocket.mp4
    |       VID_20220419_102506_reachback.mp4
    |       VID_20220419_102527_followthrough.mp4
    |       VID_20220419_102527_powerpocket.mp4
    |       VID_20220419_102527_reachback.mp4
    |       VID_20220419_102542_followthrough.mp4
    |       VID_20220419_102542_powerpocket.mp4
    |       VID_20220419_102542_reachback.mp4
    |       VID_20220419_102559_followthrough.mp4
    |       VID_20220419_102559_powerpocket.mp4
    |       VID_20220419_102559_reachback.mp4
    |       VID_20220419_102619_followthrough.mp4
    |       VID_20220419_102619_powerpocket.mp4
    |       VID_20220419_102619_reachback.mp4
    |       VID_20220419_102632_followthrough.mp4
    |       VID_20220419_102632_powerpocket.mp4
    |       VID_20220419_102632_reachback.mp4
    |       VID_20220419_102644_followthrough.mp4
    |       VID_20220419_102644_powerpocket.mp4
    |       VID_20220419_102644_reachback.mp4
    |       VID_20220419_102654_followthrough.mp4
    |       VID_20220419_102654_powerpocket.mp4
    |       VID_20220419_102654_reachback.mp4
    |       VID_20220419_102704_followthrough.mp4
    |       VID_20220419_102704_powerpocket.mp4
    |       VID_20220419_102704_reachback.mp4
    |
    \---initial_dataset
            eagle_shot1_followthrough.mp4
            eagle_shot1_powerpocket.mp4
            eagle_shot1_reachback.mp4
            eagle_shot2_followthrough.mp4
```

```
eagle_shot2_powerpocket.mp4
eagle_shot2_reachback.mp4
eagle_shot3_followthrough.mp4
eagle_shot3_powerpocket.mp4
eagle_shot3_reachback.mp4
heimberg_shot1_followthrough.mp4
heimberg_shot1_powerpocket.mp4
heimberg_shot1_reachback.mp4
heimberg_shot3_followthrough.mp4
heimberg_shot3_powerpocket.mp4
heimberg_shot3_reachback.mp4
mcbeth_shot1_followthrough.mp4
mcbeth_shot1_powerpocket.mp4
mcbeth_shot1_reachback.mp4
```

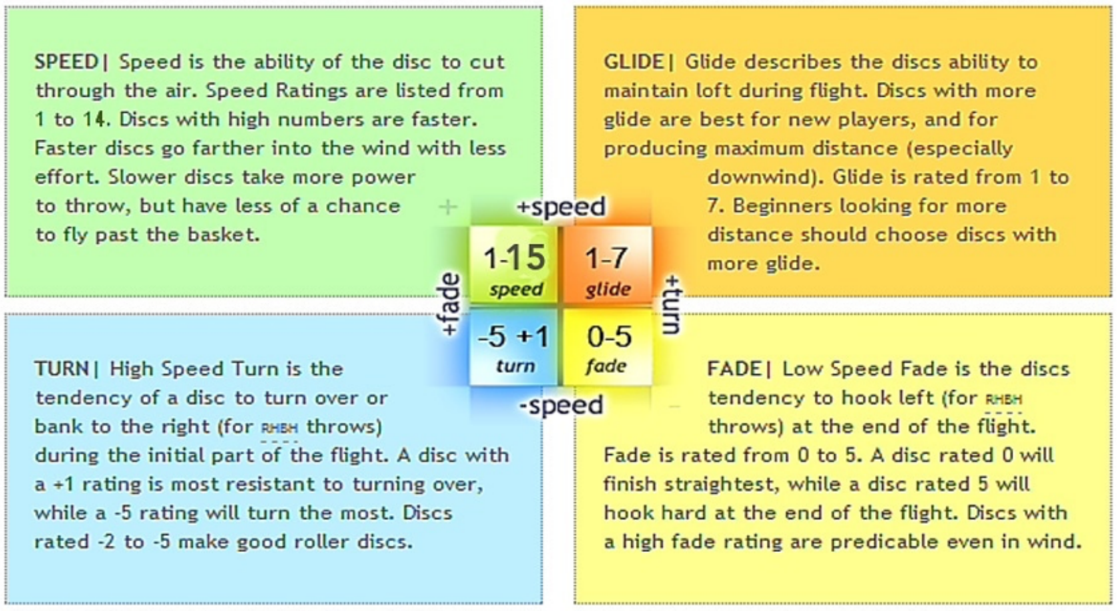
# 9 Appendix C

## Code for project

1. Data processing and LSTM models in Jupyter Lab on github: https://github.itu.dk/lakj/Bachelor-Project

2. MediaPipe Pose Classification in Google Colab: https://colab.research.google.com/drive/1JRyjoKwbtaNJHV8ra-ERfd2j3U0Obas7?usp=sharing

## Flightnumbers explained


SPEED| Speed is the ability of the disc to cut through the air. Speed Ratings are listed from 1 to 14. Discs with high numbers are faster. Faster discs go farther into the wind with less effort. Slower discs take more power to throw, but have less of a chance to fly past the basket.

GLIDE| Glide describes the discs ability to maintain loft during flight. Discs with more glide are best for new players, and for producing maximum distance (especially downwind). Glide is rated from 1 to 7. Beginners looking for more distance should choose discs with more glide.

TURN| High Speed Turn is the tendency of a disc to turn over or bank to the right (for RHBH throws) during the initial part of the flight. A disc with a +1 rating is most resistant to turning over, while a -5 rating will turn the most. Discs rated -2 to -5 make good roller discs.

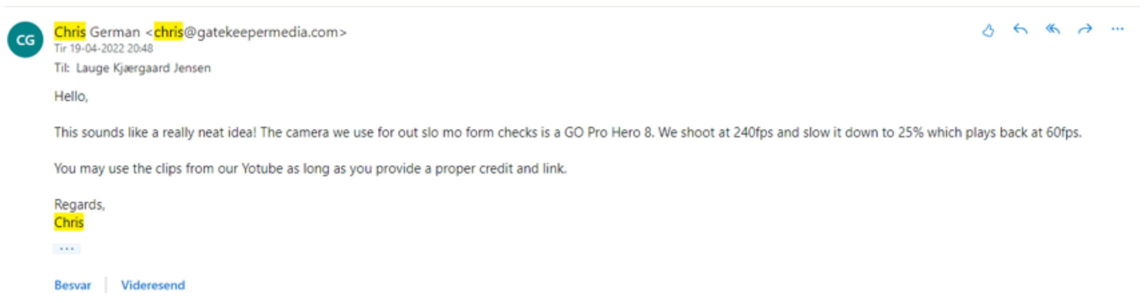
FADE| Low Speed Fade is the discs tendency to hook left (for RHBH throws) at the end of the flight. Fade is rated from 0 to 5. A disc rated 0 will finish straightest, while a disc rated 5 will hook hard at the end of the flight. Discs with a high fade rating are predicable even in wind.

Figure 52: Flightnumbers explained

## Mail from GatekeeperMedia

Thanks to GatekeeperMedia for allowing me to use following videos:

- https://www.youtube.com/watch?v=m8E3kCqtKzU&t=95s
- https://www.youtube.com/watch?v=adznE_7UUEA&t=559s
- https://www.youtube.com/watch?v=EahjLxGn42s
- https://www.youtube.com/watch?v=zwQtFSOGXaE&t=85s
- https://www.youtube.com/watch?v=zeKHPH1_wLg&t=1s
- https://www.youtube.com/watch?v=fC9W4Eux_6g&t=98s



Chris German <chris@gatekeepermedia.com>
Tir 19-04-2022 20:48
Til: Lauge Kjærgaard Jensen

Hello,

This sounds like a really neat idea! The camera we use for out slo mo form checks is a GO Pro Hero 8. We shoot at 240fps and slow it down to 25% which plays back at 60fps.

You may use the clips from our Yotube as long as you provide a proper credit and link.

Regards,
Chris

Figure 53: Initial dataset recorded with Go Pro Hero 8

## Similarities between form of professional and amateur

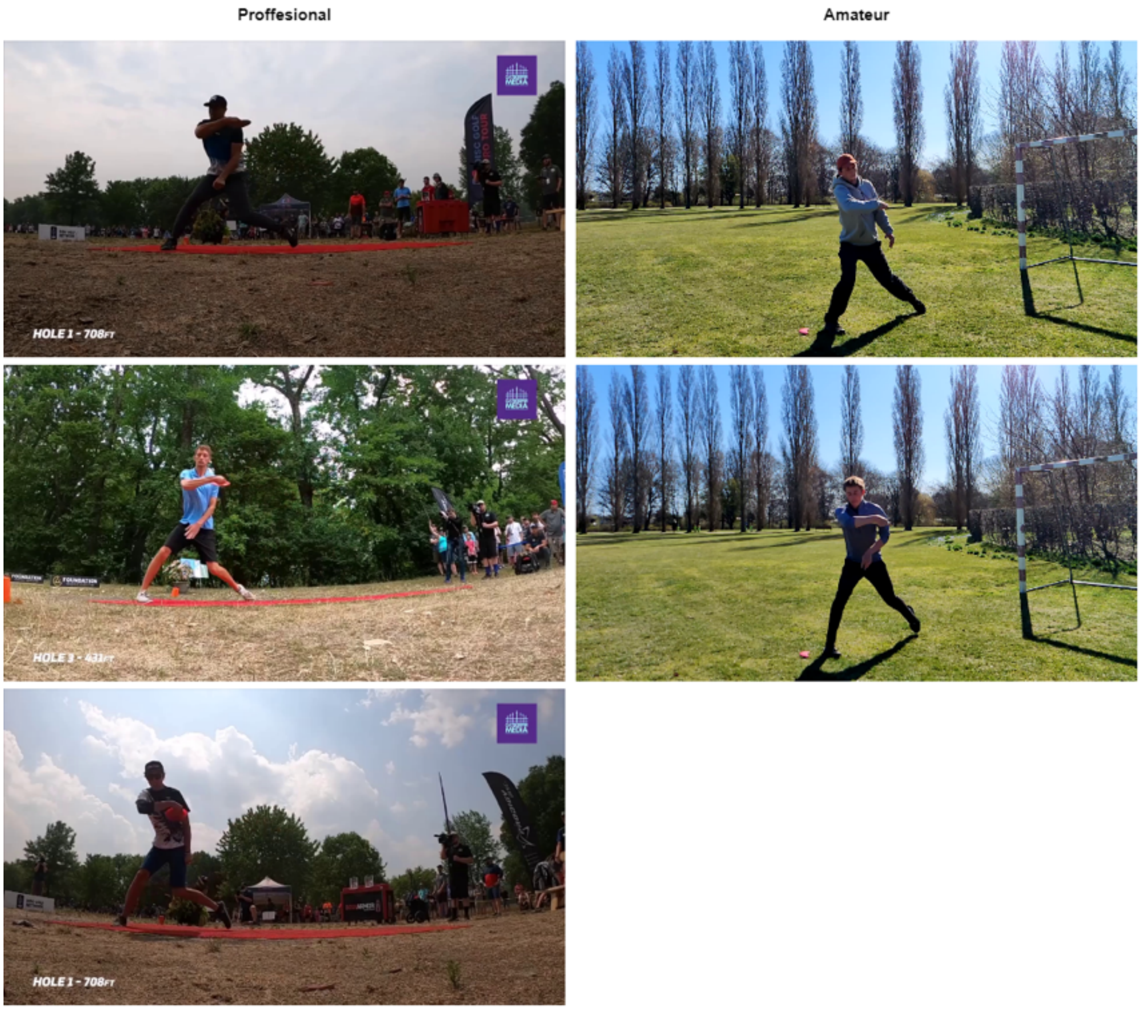


Figure 54: Comparison between powerpocket of pros and amateurs

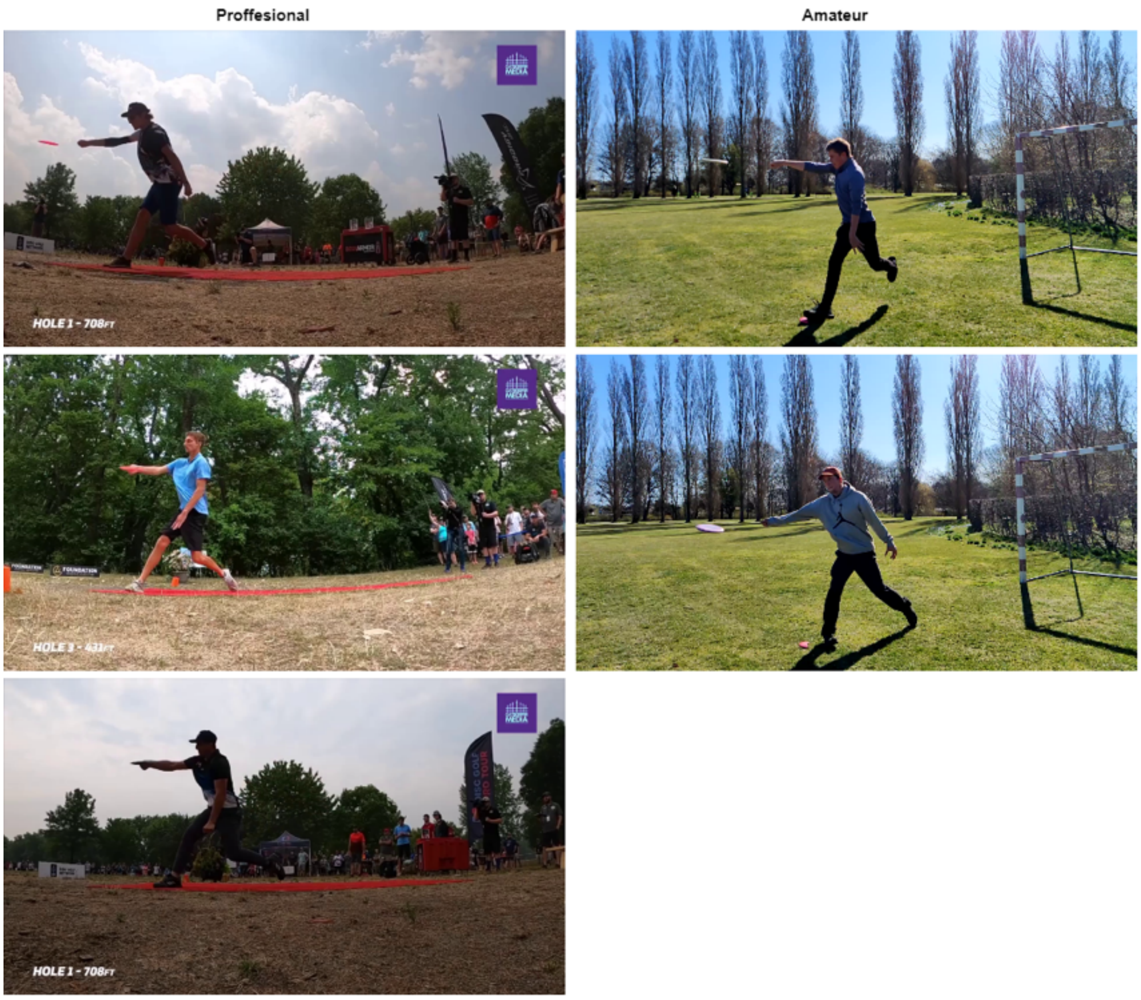


Figure 55: Comparison between followthrough of pros and amateurs

# Bibliography

[1] Christopher Olah. *Understanding LSTM Networks*. 2015. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/. (accessed: 16.05.2022).

[2] Ezra Aderhold. *Power Pocket Explained!! (For Distance) - Disc Golf*. URL: https://www.youtube.com/watch?v=uQIzQcY05nE. (accessed: 16.05.2022).

[3] Professional Disc Golf Association. *United States Tour Rankings*. URL: https://www.pdga.com/united-states-tour-ranking. (accessed: 16.05.2022).

[4] Google. *MediaPipe Pose*. URL: https://google.github.io/mediapipe/solutions/pose.html. (accessed: 16.05.2022).

[5] Google. *MediaPipe Pose Classification*. URL: https://google.github.io/mediapipe/solutions/pose_classification.html. (accessed: 16.05.2022).

[6] Google. *Tuesday Tips: Breaking Down The Backhand*. URL: https://discgolf.ultiworld.com/2017/03/28/tuesday-tips-breaking-backhand/. (accessed: 16.05.2022).

[7] *How Big Data Analytics is Changing Sports*. URL: https://medium.com/analytics-vidhya/how-big-data-analytics-is-changing-sports-40dc72c26fc8. (accessed: 16.05.2022).

[8] *sklearn.model$_s$election.train$_t$est$_s$plit*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html. (accessed: 16.05.2022).

[9] *TensorFlow API*. URL: https://www.tensorflow.org/api_docs/python/tf/keras/. (accessed: 16.05.2022).