A Tennis Shot Classification Pipeline using Pose Estimation

Kevin Bretz[1]

Leiden University, Leiden , NL

https://www.universiteitleiden.nl/en/education/study-programmes/master/computer-science

Abstract. In this paper I will present a straightforward pipeline that can be used to train and evaluate a computer vision model to classify tennis shots recorded for example during a TV broadcast. I make use of a 2D pose estimation model to greatly compress the networks input data while preserving critical spatial and temporal relationships.

I start by stating the relevance of this paper from a performance and coaching perspective, followed by explaining how the data is gathered, extracted and preprocessed, as well as highlighting certain design choices. Following this, I elaborate on the exact training process, how the training data is split, and what the model architecture looks like. Finally, I assess the training results, discuss their implications as well as an exemplary use case, and finish by making suggestions for future work.

Each step of the presented pipeline, from data collection and manual annotation to preprocessing and model architecture, has potential to be greatly expanded on and optimized. However, due to this project being conducted by a single person and under limited time, the focus is not on flawless optimization and performance maximization, but rather on overall proof of concept. With this in mind, if promising results can be achieved using limited data and minimal optimization, it stands to reason that working with pose estimation models instead of raw video input could yield great benefits in the entire field of sports action classification.

All of my produced code is in the submission and the entire pipeline can be followed along step by step. Instructions are in the README.md file.

Keywords: Action classification · Sports Data Science · Pose Estimation

1 Client Analysis

Automatic action classification systems are invaluable to various parties within a specific sport. Broadcasters and media companies gain value through such models thanks to them enabling real-time content indexing, leading to more accurate

highlight generation and on-screen analytics, and ideally less manual labor and human error.
With further advancements, such systems could also be used by coaches and performance analysts to gain nuanced insights on a players technique, tactics and patterns through precise motion analysis, which can be used to identify a players strengths and weaknesses and adjust training accordingly. In the same manner, they can be used to analyze opponents and come up with specific game plans.
Such a system could even be beneficial for sports tech companies and startups to develop commercial, non-professional products for training, fan engagement, and virtual coaching.

To meet the demands of these clients, first a base functionality of performing action classification on video footage must be achieved. This will be the goal of this project. All in all, an automatic shot classification system for tennis videos would deliver a key foundation that can be expanded and built upon by various entities within the tennis world.

2 The Data

In the following section I will go over which data I decided to use for training my model, as well as how it was preprocessed and labeled.

2.1 Acquisition

Over the past years there has been a steep increase in the amount of official, professional level tennis matches that have become widely available for free on the internet. Indeed, all of the 4 grand slams have their own YouTube channels with a huge range of tennis broadcasts from the past decades freely available. It is from these recordings that I have hand picked 4 different tennis matches to manually extract and label shots from. The 4 matches I will be using for training, validating and testing my model are:

Table 1. Selected video data

Tournament	Year	Player 1	Player 2	Citation
US Open	2024	Jannik Sinner	Taylor Fritz	[1]
Australian Open	2025	Novak Djokovic	Carlos Alcaraz	[2]
Australian Open	2020	Novak Djokovic	Roger Federer	[3]
Australian Open	2017	Roger Federer	Rafael Nadal	[4]

These matches were chosen for the following reasons:

1. **Video Quality**: All of the recordings are fairly recent, with the oldest one being roughly 9 years old. This means that the video quality and resolution

is adequate to be able to identify players on the far side of the court. In older recordings with lower resolutions our human detection model struggles to perform well.
2. **Video Length**: The matches are long enough to gather a wide variety of shots.
3. **Court Surface and Colors**: Our human detection model was under performing and frequently not recognizing players on clay and grass courts, so we decided to omit these surfaces from the dataset.
4. **Player Variety**: We chose matches including players with distinct styles and differences so that our resulting classification model stays as generalized as possible. For example, we included a match of Roger Federer vs Rafael Nadal at the Australian Open in 2017. Federer has a notorious one-handed backhand, which, although not employed as commonly as a two-handed backhand, is still a legitimate backhand stroke and should therefore be recognized by the model. Nadal on the other hand is left handed, another variation that must be correctly generalized by our model.

This step leaves us with full recordings of tennis matches.

2.2 Shot Extraction

Taking an entire broadcast video and audio recording as input, I have developed a script that automatically extracts clips of individual shots from the full recording. It detects audio spikes above a certain threshold and assumes they are the sound of the ball being hit by a racket, then extracts a clip starting 0.8 seconds before impact and ending 1 second after. Clips are therefore 1.8 seconds long. The video frame rate is set to 25fps, so every clip contains a sequence of 45 frames.
This is a very basic automatic shot extraction algorithm that is prone to extracting false shots, or clips that are not of tennis shots. However, due to there being an inevitable manual labeling phase, the basic shot extraction algorithm will suffice and all false shots will be manually deleted later.
This step leaves us with short clips of a player making a shot, but with the entire broadcast in frame, so the shot is only happening in a small part of the frame.

2.3 Manual Labeling

Once all broadcast recordings have been automatically segmented into a collection of short clips depicting single tennis shots (and plenty of false shots), the process of manually labeling them can begin. I have also written a script to attempt to make this process as efficient as possible.
First, a clip is presented to the viewer. They then get to decide whether to keep or delete the clip. If the clip is verified, YOLOv8 [8] person detection is run on it and the viewer is presented with all detected bounding boxes and must click on which one is the player taking a shot. The selected player is then cropped out and tracked so they are always in the center of the frame, and the cropped clip

is once again presented to the viewer. Finally, the viewer gives a keyboard input to determine which shot is being played. There are a total of 8 different labels that can be assigned to a shot. Afterwards, in a separate shot_annotations.json file, all clips that are successfully labeled and therefore legitimate tennis shots get saved in combination with their respective labels. All other clips are deleted. Following this, the next novel clip is presented to the viewer, and this process is repeated until all available clips have been exhausted.
This step leaves us with short clips where the player making the shot fills the frame / is cropped out, as well as a shot_annotations.json file that maps all filenames to a label.

2.4 Pose Extraction

For the crucial step of turning the labeled clips into a sequence of pose data, I make use of the Sports2D repository from Github [5]. It provides a straightforward and simple implementation of various pose estimation models. For this project, we worked with the standard model, which is RTMPose (in the HALPE_26 configuration) [7], providing us with a total of 26 distinct keypoints. I have written a batch script for executing the pose extraction on all labeled clips. For every labeled clip, a text file is created containing a sequence of 45 sets of coordinates of estimated keypoints, one for each frame.
In theory, even if the pose estimation isn't perfect and produces errors, as long as these behave consistently over all data a model should still be able to distinguish between different shots. For example, when the player is facing away from the camera, their hands and arms may not be visible, so the pose estimation may be inaccurate. However, as long as the pose estimation consistently produces the same errors in a forehand sequence and different errors in a backhand sequence, they remain distinguishable.
After this step we are left with text files containing sequences of pose data. The pose data is still raw, meaning the coordinates are based on pixel values and are not normalized.

2.5 Preprocessing Pose Data

This is the final step before training can begin, and I have provided a python script which performs these processes. Currently, our pose data consists of absolute pixel coordinates. We need to preprocess our pose data uniformly so that our data becomes directly comparable and our model can effectively be trained. The following steps are performed individually for each pose in a sequence of poses.
First, we redefine the center of our coordinate system as being exactly where the hip joint / keypoint is. Effectively, the hip joint coordinates are subtracted from all keypoints. Now, the new hip joint coordinates are (0, 0).
We also need to evenly scale our pose data. To do so, we calculate the distance between the hip and the neck joint, and divide it by 50. We then divide every

keypoint coordinate by this value, leaving us with a hip to neck distance of exactly 50 for every pose, with the rest of the body scaled accordingly.

Additionally, we also employ a data augmentation technique to account for left-handedness. If trained on non-augmented data, the model may learn to classify shots based on which side of the body movement is happening on, as the majority of people are right handed and therefore play a forehand on their right side and a backhand on their left side. To counteract this and force the model to look more closely for finer details that can define a shot, we produce a mirrored copy of every single pose sequence by multiplying all x coordinates by -1.
After this step we are finally left with a normalized sequence of keypoint coordinates which can be used to train a classification model.

3 Creating the model

To solve the task of classifying a sequence of pose coordinates into different tennis shots, I employ a neural network to train a classification model. The model is trained on and takes an input of a sequence of 45 pose coordinate sets, which were produced in the previous section. It is trained in an identical manner as a computer vision classification model, but with pose information as input data. In the following section I will explain the process of training an action classification model using the data we previously obtained.

3.1 Architecture and Hyperparameters

According to [6], a Bidirectional LSTM is especially well suited for a sequence classification task. The sequence length is directly linked to the amount of frames per extracted shot clip, equaling a total of 45 poses per sequence. I achieved the most promising results using following hyperparameters:

- Batch Size = 8
- Epochs = 10
- Learning Rate = 1e-5

3.2 Data subdivision

After completing the previously defined data acquisition and preprocessing pipeline for the above mentioned 4 tennis recordings, I am left with a total of 3158 labeled clips. Of these, a total of 3007 clips belong to the labels "Serve" (815), "Forehand" (1192), and "Backhand" (1000). Indeed, these are the most commonly played shots in tennis. This leaves us with a total of 151 clips that belong to one of the other 5 labels, which is not enough data to train on. Ideally, there should be a similar amount of data for each classification. For the other labels to be effectively classified, I would first need to produce much more training data. I will therefore only train a model to classify the 3 labels for which enough

training data was gathered.
From the 3007 pose sequences, 100 are removed from the training data at the start of training to be strictly used for testing post training. During training, a subset of 80% of the remaining data is used for training, and 20% for validation. This is performed for a fixed amount of epochs, after which the training and validation data gets reshuffled and a new run starts, continuing with the previously obtained model. Training only ends once a run is achieved in which every single epoch returns a validation accuracy > 0.995.
To confirm that the model is not overfitting to its training data we evaluate it on the test pool of 100 shot sequences.

4 Results

The results of 2 sample training instances can be found in the submitted *Example 1* and *Example 2* directories. In the *Example 1/incorrect_ predictions.txt* file one can see that during testing of this particular model using precisely 100 entirely novel pose sequences, it only misclassifies 2 of them, achieving an overall test accuracy of 98%. Our model is therefore proven to work given appropriate test data and achieve an impressive accuracy. This can be verified using the provided *validate.py* file, instructions are located in the README.md.
1 visualizes the training progression for the provided *Example 1*. One can observe that the model makes rapid leaps in performance during early stages of training, already averaging a precision over 97% after just 10 episodes. Further training leads to lowered variance and greater consistency in the model's average performance, until it eventually reaches our predefined end condition after 61 episodes.
The second example training scores a validation accuracy of 97%.

4.1 Example Use Case

In this example, I have developed an application which is intended for professional tennis players and coaches to hopefully give them a competitive edge over their opponents. The player would select their opponent and a given time frame, and then the system would automatically analyze all video footage of the opponent within the given time frame. The model would then gather a shot count for the opponents opponent in every single match. This information could be strategically useful, as it will reveal wether the opponent has a preference to which side they hit the ball, which can be positionally accommodated for, and makes them potentially more predictable.
In the example visualization seen in 2 one can see that the opponent's opponents predominantly played backhands at a rate which is statistically far beyond being a coincidence, indicating a possible preference to playing to people's backhands. For the majority of the population who is right-handed, this would be there left side. It would therefore be tactically clever to make sure to always position yourself slightly further to the left if you were to play against this opponent, or

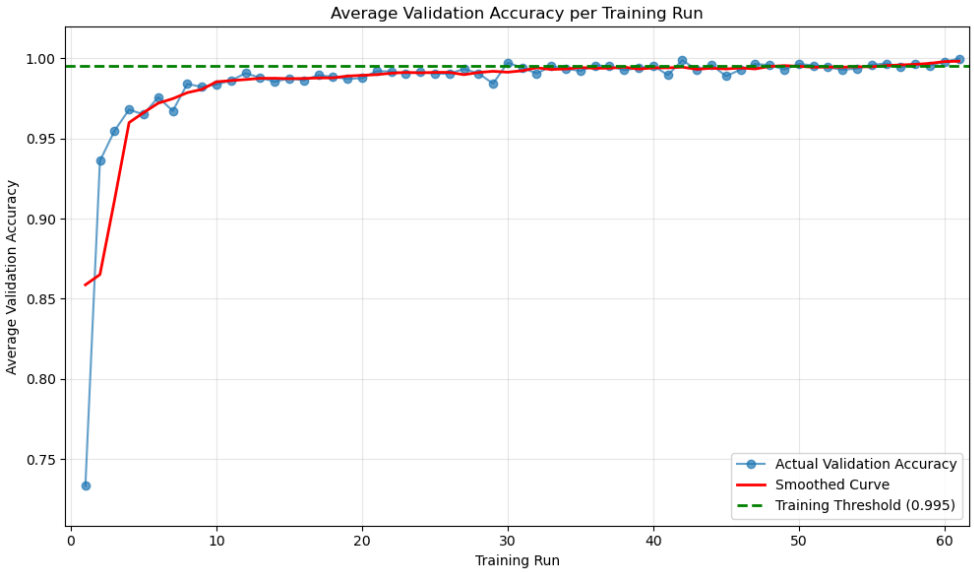

Fig. 1. Average validation accuracy per run

to at least be aware of this tendency and to try to play into it.

This example is kept very rudimentary as it respects the limitations of the model produced in this project, which can only reliably distinguish between a limited number of labels. A theoretical improved model could, for example, also determine wether the shot was successful or not, providing even more potentially statistically significant insight about an opponents strengths and weaknesses to a competing party, and opening the door to more complex strategic positioning and ball placement.

5 Discussion

The results of this project as a proof of concept are very promising. Supervised learning with pixel data requires a notoriously large amount of data and time consuming manual labeling. If it is feasible to reduce the models input dimensionality to only a very limited amount of numerical coordinates instead of entire RGB matrices through pose estimation models, preserving and condensing all information that is crucial for solving the task while discarding everything else, exponential reduction in required training data, time and complexity is to be expected, so this approach deserves to be further explored.

In this section, I will highlight the optimization steps that were skipped throughout the pipeline in favor of time and proof of concept, and finish by making a proposition for future work.

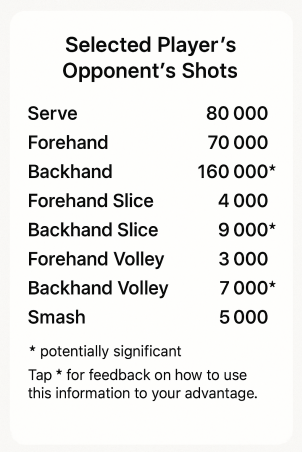

Fig. 2. Example of potential analysis feedback

5.1 Flaws

Not a lot of optimization went into each individual step of the pipeline, as the focus was primarily on seeing if such a pipeline could be realized using free open source and readily available software and libraries and the limited amount of time at my disposal. Here is a list of all the individual elements of the pipeline that should be expanded upon and optimized, given extra time, or possibly in future work:

1. **More Diverse Data**: For such a model to work consistently it should be trained on a much more diverse dataset. It should include data from players with widely varying playing styles and also from different court surfaces, as surfaces can influence the bodies movement, especially footwork, during a shot. For example, players tend to slide a lot more on clay and grass courts, and there are nuanced differences in the racket swing based on court surface as well, as the balls speed and bounce behavior varies.

2. **Label verification**: As the task of labeling each pose sequence must be completed manually, it is prone to human errors. Indeed, when verifying that the validated label is actually what is seen in the video, I occasionally come across incorrectly labeled data. There should therefore be a method implemented for verifying the given labels, wether that be automatically, potentially by a separate model, or, once again, completely manually, or possibly a combination of the two.

3. **Hyperparameter optimization**: Although the proposed training pipeline leads to a functioning model, not much can be inferred about the effect of it's chosen hyperparameters. Ideally, a hyperparameter ablation study should be

performed to determine the ideal combination of values.

4. **Improved Preprocessing**: The chosen preprocessing steps were defined very briefly to see if any results at all could be achieved, with the intention of reformulating them once training progress could be assessed. However, training immediately went well, which could indicate that our preprocessing steps were chosen immaculately, or more likely means that the entire approach and pipeline is quite robust and well thought out, so that even with sub optimal preprocessing and hyperparameters, I am still able to achieve promising results. On a less optimistic note, it could also mean that our test data is too similar to our training data. In a performance-oriented variation of this work, it would be highly beneficial to experiment with numerous preprocessing options and combinations.

5. **Unbiased Testing**: Our entire data set is shuffled into one large pool of sequential pose data, and for every training, 100 data objects are isolated from the rest and only used for validation post-training. This works to prove that the model is successfully learning to classify these specific players, on those specific courts, on that specific day, but leaves me guessing how it would perform on a completely different set of data, provided it is obtained using the same pipeline as the training data. Gathering and labeling the data for this project already took up the bulk of my time for this project, so I leave this for future attempts.

6. **Pose Estimation and Calculation**: A more accurate, error-free pose estimation model should be chosen. Although even with errors the model can still learn to correctly classify a shot, the model learns to recognize the errors as indicators of a classification, which will cause the model to fail on error-free data (overfitting to the errors).

   Although 2D pose estimation models work well to build a model for television broadcasts (as demonstrated in this paper), they only function correctly when input footage has been recorded with very specific camera orientation and positioning. Training a model with this approach to function from every imaginable angle and distance would take a lot of time, necessitating an infeasible amount of carefully hand picked training data, as well as precise tuning.
   It is much more efficient to work with 3 dimensional poses, provided they accurately depict the whole body throughout the entire duration of the sequence. Given appropriate normalization and preprocessing, input data could be captured from all possible perspectives.

5.2 3D Poses and future work

Originally I had intended to work with 3D pose data to hopefully eliminate the dependency on camera perspective from action classification tasks. This could

have large implications for the entire field of human action classification. However, I struggled to get satisfactory results with the 3D pose estimation models I was able to get running.
Alternatively, it may be feasible to reliably augment 2D data into it's 3D counterpart, either algorithmically or through a dedicated "2D to 3D" pose estimation model. This proved to be beyond the scope of this project, therefore I leave it here as a proposition for future work.

References

1. US Open Tennis Championships, "Jannik Sinner vs. Taylor Fritz Full Match | 2024 US Open Final" *YouTube*, uploaded by US Open Tennis Championships, Sep. 10, 2024. [Online]. Available: https://www.youtube.com/watch?v=Ce3dRYHWIBI
2. Australian Open TV, "Novak Djokovic v Carlos Alcaraz Full Match | Australian Open 2025 Quarterfinal" *YouTube*, uploaded by Australian Open TV, Feb. 4, 2025. [Online]. Available: https://www.youtube.com/watch?v=ATI9B7ZLof8
3. Australian Open TV, "Novak Djokovic v Roger Federer Full Match | Australian Open 2020 Semifinal" *YouTube*, uploaded by Australian Open TV, Aug. 13, 2023. [Online]. Available: https://www.youtube.com/watch?v=i29zruv8mKc
4. Australian Open TV, "Roger Federer v Rafael Nadal Full Match | Australian Open 2017 Final" *YouTube*, uploaded by Australian Open TV, May 27, 2022. [Online]. Available: https://www.youtube.com/watch?v=KTCDxjJvs2U
5. David Pagnon, "Sports2D: 2D pose visualization and annotation tool for sports" *GitHub*, 2023. [Online]. Available: https://github.com/davidpagnon/Sports2D
6. M. Schuster and K. K. Paliwal, "Bidirectional Recurrent Neural Networks" *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997. [Online]. Available: https://doi.org/10.1109/72.650093
7. Tao Jiang, Peng Lu, Li Zhang, Ningsheng Ma, Rui Han, Chengqi Lyu, Yining Li, Kai Chen, and MMPose Contributors. *RTMPose: Real-Time Multi-Person Pose Estimation based on MMPose*. GitHub repository: https://github.com/open-mmlab/mmpose/tree/main/projects/rtmpose, 2023.
8. Glenn Jocher, Ayush Chaurasia, and Jing Qiu. *Ultralytics YOLOv8*. Version 8.0.0, Ultralytics, 2023. GitHub, AGPL-3.0. https://github.com/ultralytics/ultralytics