TeCNNis: Predicting Tennis Shot Landings in Real-Time

A dual-CNN system for sports analytics

By Group 10

Introduction

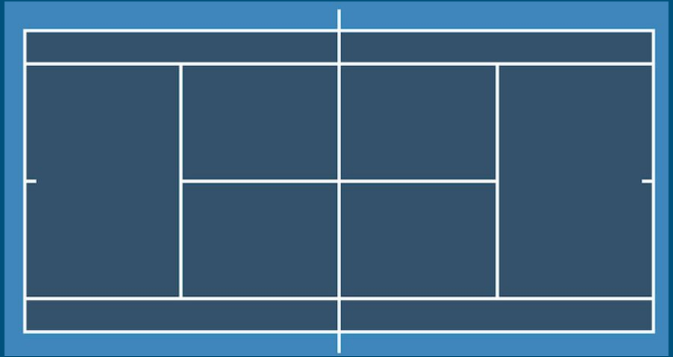
- Most modern tennis tracking systems analyse swing data **after** the shot - we want to create a model that can predict where the ball lands as the shot is being taken
- Many modern technologies analyse the swing using either video footage, or in-racket technology, or both
- The architecture of a CNN allows the model to identify many key aspects automatically for image analysis (video frames)

Research Goals

1. We want to create a CNN that can detect the exact frame of a video in which a tennis player hits the ball (this will be CNN 1)
2. We want to predict the shot's landing coordinates on the court using the frames around the time of the swing (this will be CNN 2)

Both of these will be accomplished using convolutional neural networks (CNNs) trained on real tennis footage

Our Dataset

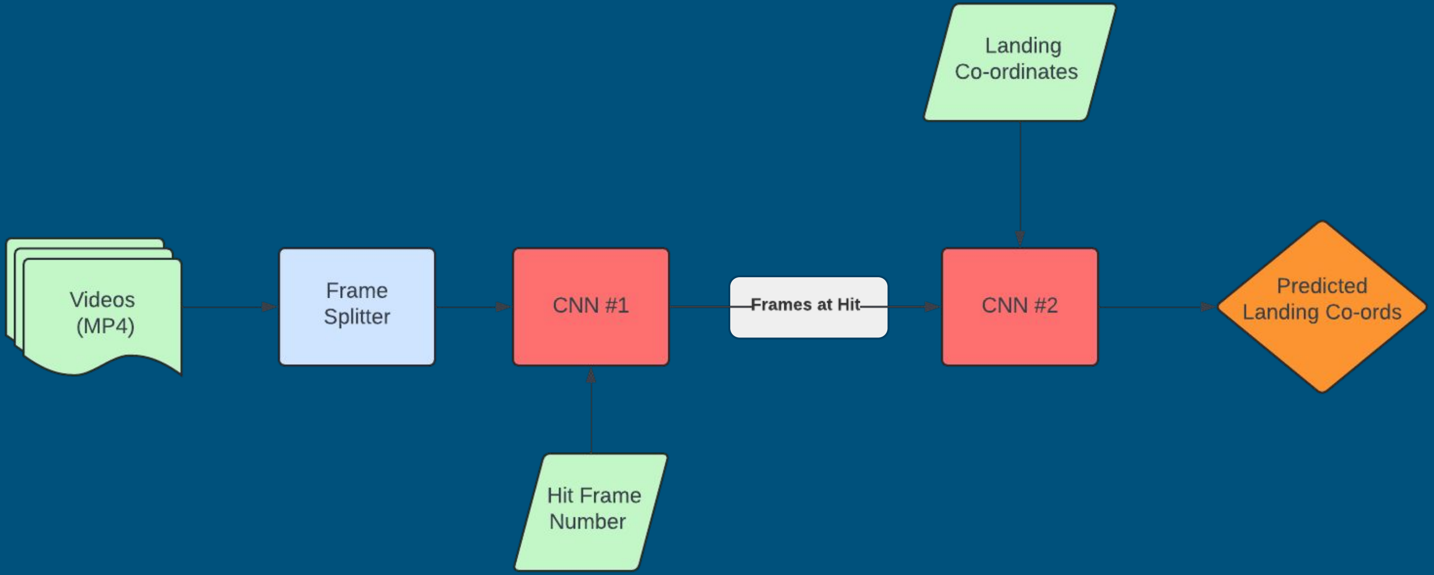


Our dataset “Tennis Shot Side-View and Top-View Data Set for Player Analysis in Tennis” contains 472 clips (side-view and top-view each are 236 clips).

It classifies into two main directories: Outdoor field (tennis clay court) and Indoor field (indoor leisure centre type courts). Each of which have two shot types: straight shot and cross-court shot. *We will only be using the top-view videos for our training.*

After the shot, each ball’s landing point coordinates was recorded. These data are recorded in CSV files. The file’s name indicates where the data is from: field categories and shot types. The coordinates are represented by distance from the closest doubles sideline and the distance from the baseline.

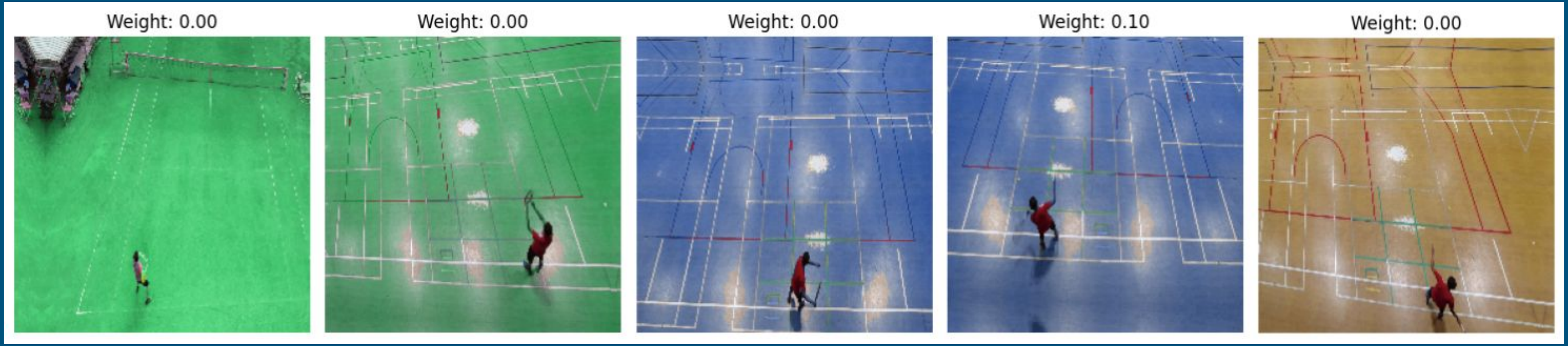
TeCNNis Pipeline



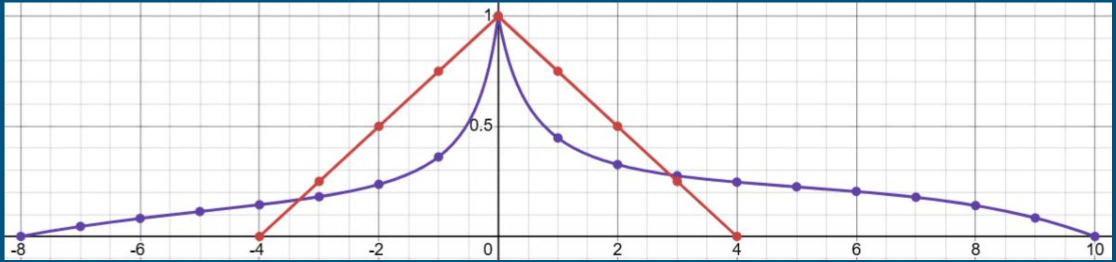
Methodology

1) **Video Preprocessing:** As the videos capturing each full swing and ball landing from the player are in MP4 format we will need to split these into the individual image frames.

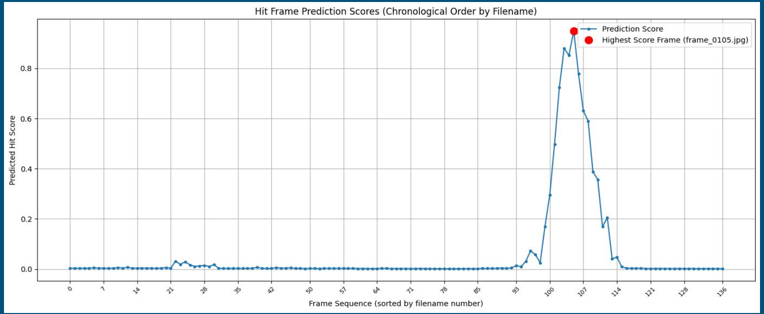
Data Augmentation: In order to increase the amount of data we could train our model on (by an order of ~100), we augmented our data in the following ways: horizontal flipping, varying the colour of the court to either red, green, or blue, and a perspective (keystone) shift.



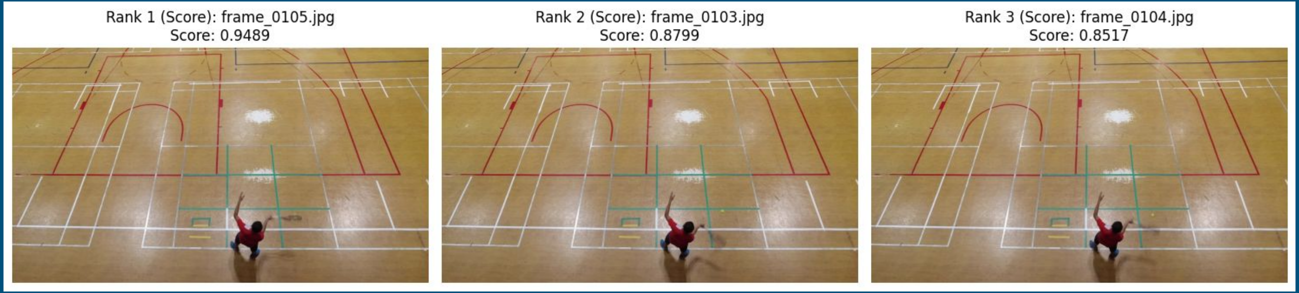
Methodology



Data Preprocessing continued: We tried a few different frame weighting systems: Linear and variable (using Bayesian optimisation).



2) **CNN #1 Hit Detection:** The sequence of individual image frames is passed into the first CNN to identify the region of frames in which the ball is hit, making this a regression and not a classification task.

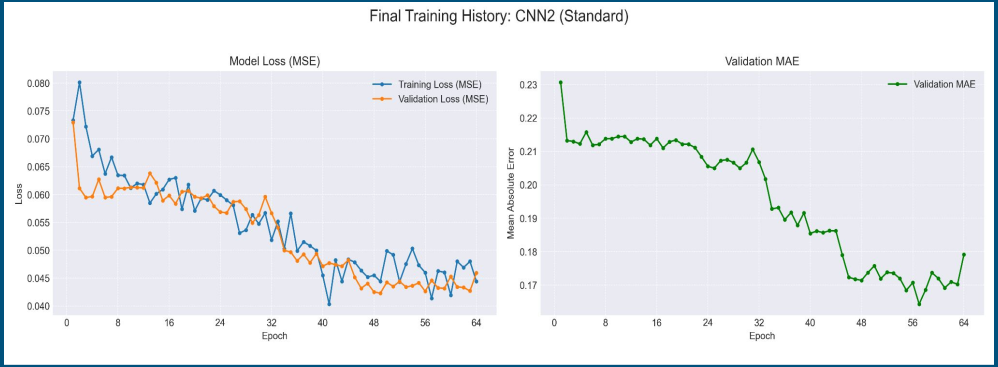


Rank 1 (Score): frame_0105.jpg
Score: 0.9489
Rank 2 (Score): frame_0103.jpg
Score: 0.8799
Rank 3 (Score): frame_0104.jpg
Score: 0.8517

Methodology

3) **CNN #2 Landing Prediction:** After receiving the predicted hit frame(s) from CNN #1, the second CNN takes the selected frame(s) as input and predicts where in the opponent's section of the court the ball will land. This CNN is trained using labeled data that includes the ball's To-Closest-Doubles-Sideline-Distance (meters) and To-Baseline-Distance (meters). These two values create a 2D coordinate that represents the landing location of the ball on the opponent's side of the court.

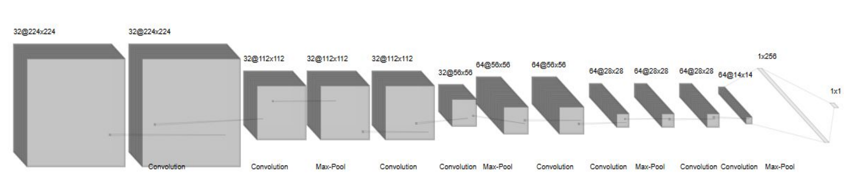
4) **Output:** The final output of this workflow is the set of predicted coordinates. For example: (dist_left, dist_right, dist_back) = (2.48m, 8.49m, 2.73m)



All training was conducted using mean square error as the loss criterion and Adam as the optimizer, and using early stopping to prevent overfitting. For training, we used a NVIDIA GPU: T4 on Colab, or local RTX. We usd PyTorch as our framework

CNN Structures

CNN 1

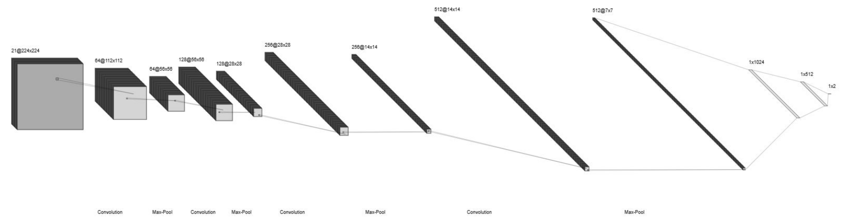


Input: Single 3-channel frame (224x224)

Structure: 4 Convolutional blocks (Conv -> BatchNorm -> ReLU -> Conv -> BatchNorm -> ReLU -> MaxPool)

Filters: [32, 32, 64, 64]

CNN 2

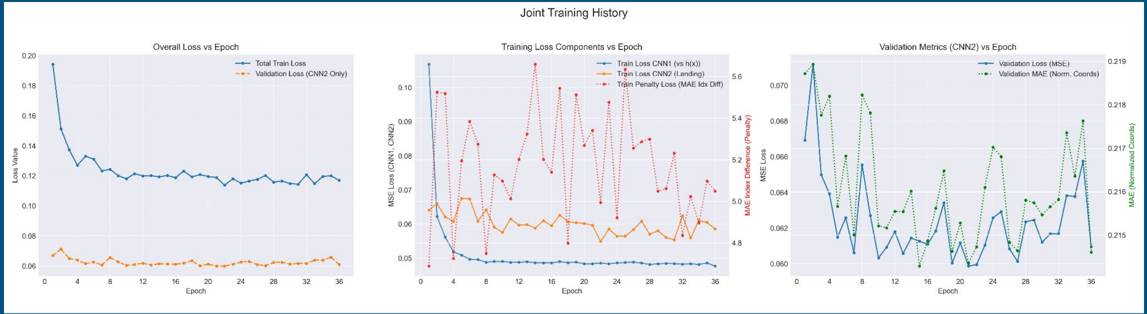


Input: Stacked sequence (e.g., 9 frames x 3 channels = 27 channels) (224x224)

Structure: 4 Convolutional stages with decreasing kernel sizes, increasing filters

Filters: [128, 256, 512, 512]

Experiments



We experimented to tune hyperparameters such as:

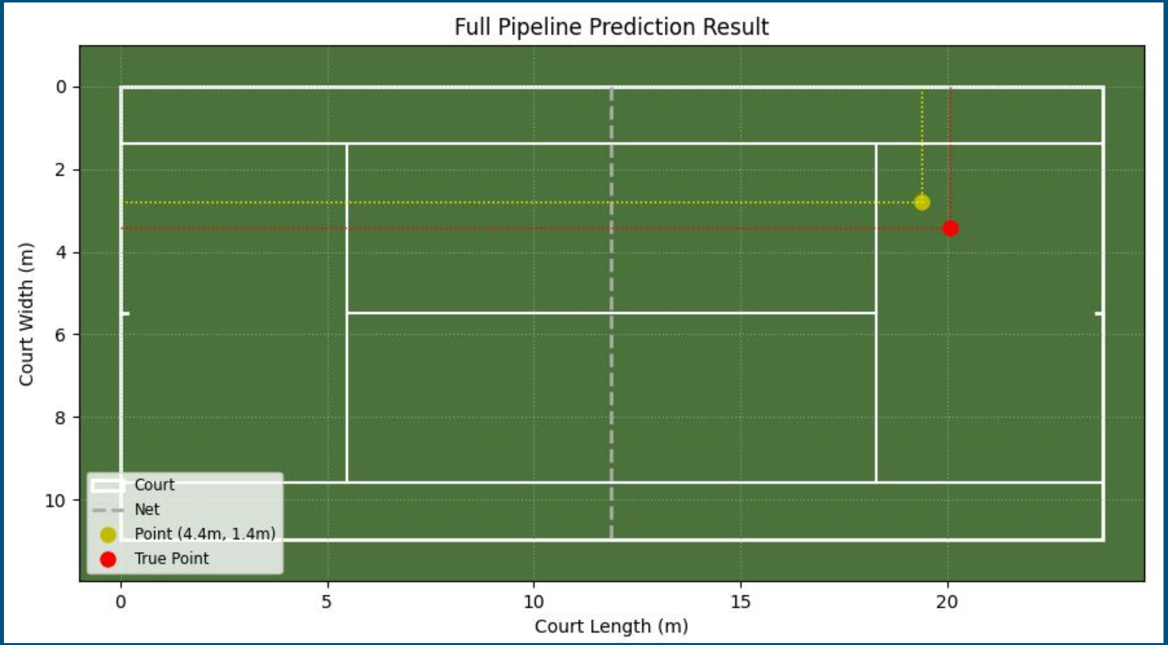
- learning rate
- batch size
- the number of frames weighted and fed into the second CNN
- the method of weighting the frames

We also experimented for individual vs. joint CNN training (shown in the image above) with joint loss equal to CNN 1 loss plus CNN 2 loss plus a timing penalty to ensure frames weren't selected to far after the hit frame

Furthermore, we compared our custom CNNs against pre-trained models (like ResNet, ViT, EfficientNet) fine-tuned for our task.

Results

Prediction Accuracy: The pipeline successfully predicts landing locations close to the true points in test videos.



Example full pipeline prediction: Yellow dot (predicted) vs Red dot (true).

Discussion / Limitations

Successes:

- Developed a functional dual-CNN pipeline for predicting tennis landings
- Achieved latency potentially suitable for near real-time use cases
- Custom-designed models proved more efficient (faster) than larger pre-trained models for this specific task

Limitations:

- Data: Trained on a dataset with limited views (top-down) and environments (2 types). Generalisation to diverse professional footage (different angles, lighting, stadiums) needs more varied data.
- Labeling: Acquiring labeled data (landing coords for pro matches) is challenging.
- Joint Training: Our end-to-end training approach needs further refinement to outperform separate training.

Future Directions:

- Acquire more diverse training data
- Optimize models further (e.g., quantisation, ONNX) to reduce latency
- Improve joint training methodology

Conclusion

We successfully designed and implemented ”TeCNNis”, a dual-CNN system to predict tennis shot landings in near real-time.

The system identifies the hit frame (CNN 1) and predicts landing coordinates (CNN 2) using video input.

Our custom architecture demonstrates competitive accuracy with significantly lower latency compared to fine-tuned standard models, making it viable for time-sensitive applications.

Potential applications include enhanced sports analytics, automated line-calling aids, or informing live betting markets.

Key challenges remain in data acquisition for broader generalisation and refining advanced training techniques.

Thank You!

Any questions?