Final Report for Computer Systems Lab SmartRally: An AI Approach to Coaching Badminton Strategy with Mobile App

İpek Sayar, Jacob Dipasupil

May 28, 2025

Gabor Period 5, Yilmaz Period 3

Table of Contents

Abstract

This project presents an AI-powered model coaching system that helps badminton players improve their skills by analyzing gameplay videos with computer vision and deep learning techniques. The system focuses on three main aspects of badminton performance: the player's actions, the shuttlecock's movement path, and the player's positions on court. It gives personalized feedback and recommends the best action to take in each situation, which is called the Next Best Move (NBM). The neural network is trained to predict the NBM accuracy reached an accuracy of about 70%.

To track where the shuttlecock is hit and where it lands, the system uses a tool called TrackNet. Player movements and locations are calculated using a YOLOv11 OpenPose model. The results are displayed in a video that includes color coded visual feedback showing the player's shot, direction they are aiming at, and color coded captions that mark important information.

The project also includes a mobile app developed with Kivy, which allows the user to upload their videos, view a series of videos with the generated feedback, and store the generated analysis for future reference. Overall, this system provides a low cost and accessible way for badminton players to receive training and improve their performance using AI.

1 Introduction

Badminton is not a very popular sport in the US. The US has never won an Olympic medal of any kind in badminton. The majority of badminton clubs in the US are in California, and there is only one club in Virginia, located in Ashburn, with dedicated badminton facilities. In

southern California, the price of badminton coaching can range from $50-100 per hour, which makes it very hard to get regular practice and coaching for lower income families. Finding and securing transportation to a practice site, and paying expensive coaching fees are major obstacles for lower income athletes trying to get into the sport. We developed an AI-powered app for badminton that leverages advanced computer vision and machine learning techniques to enhance players' skills by analyzing their stroke techniques and offering personalized feedback. Unlike traditional coaching, which is limited by the availability and cost of personal trainers, this AI system enables players to receive continuous, precise feedback through video recordings, significantly reducing the need for frequent in-person coaching sessions and making high-quality training more accessible.

2 Background

A group of three students at MIT-WPU in Pune, India published a paper on their research developing an AI which takes gyroscopic data from a ‘smart racket’ and badminton videos to perform sport analysis [3]. The authors also focus on considerations for the player’s physique, diet, and body type to perform their sport analysis. Their application to badminton centers largely around analyzing the posture, racket handling, angle of the stroke, hip and leg movement, and other aspects of badminton form. The authors do not explicitly state an accuracy they achieved, but they mention that a key issue in their research was the need for cleaner data to improve accuracy and efficiency.

Additionally, the software tools that we’re using have also considerably improved over time. The MIT-WPU team, for example, used YOLOv3 for object detection; we adopted YOLOv11—a more recent version. Newer versions of YOLO indeed have different

enhancements, including accuracy improvements, faster processing times, and better usage of computational resources. For example, YOLOv11 has enhanced capabilities for occlusion handling and multi-scale detection, which means it is even more reliable in identifying players and shuttlecocks under different conditions. This further enables more accurate and smooth analysis for improving players.

2.1 VideoBadminton

The VideoBadminton dataset—compiled by five students from Auburn University and National Central University in Taiwan—is derived from high-quality badminton footage, and provides clean footage of badminton strokes [1]. The dataset consists of 7,822 clips of 18 categories of badminton strokes, and includes detailed annotations of player locations throughout the match.

Figure 1. Directory Structure of the VideoBadminton Dataset

(a) Class of Clear

(b) Class of Cross Courtflight

(c) Class of Smash

Figure 2. The key frame samples from various badminton classes in VideoBadminton

There do exist other video-based datasets, such as the Badminton Olympic data-set—comprised of 10 YouTube videos of hour long badminton matches annotated with the player positions, instances where players score points, and stroke labels—however, we use the VideoBadminton dataset not only for its capabilities for training pose recognition models, but also to remedy the problem the MIT-WPU team ran into with clean data.

2.2 ShuttleSet

ShuttleSet is "the largest publicly available badminton singles dataset with annotated stroke-level record" [4]. Containing 104 sets, 3,685 rallies, and 36,492 strokes from top-ranking badminton singles matches, ShuttleSet holds abundant and essential information regarding both the sequences of strokes and the positioning of the shuttlecock throughout a match. Like VideoBadminton, ShuttleSet has 18 distinct classes for shot type. Unlike VideoBadminton,

however, the class labels were in Chinese and translated into English. Some of the names of the strokes were the same; however, some were different, and we did our best to match them back to the labels for VideoBadminton.

VideoBadminton Labels		**ShuttleSet Labels**
Short serve	→	Short service
Long serve	→	Long service
Lift	→	Lob
Defensive clear	→	Defensive return lob
Clear	→	Clear
Defensive drive	→	Defensive return drive
Rear court flat drive	→	Back-court drive
Flat shot	→	Driven flight
Cross Court Flight	→	Cross-court net shot
Push shot	→	Push
Drop shot	→	Drop
Transitional slice	→	Passive drop
Cut	→	Net shot
Rush shot	→	Rush
Block	→	Return net
Tap smash	→	Wrist smash
Smash	→	Smash
Short flat shot	→	Drive

Figure 3. VideoBadminton to ShuttleSet Conversion Map

We use ShuttleSet in our project for our model to learn how to predict where and how to hit the shuttlecock throughout the match to produce the most consistent results. We preprocessed the ShuttleSet dataset to keep only five key features: the type of shot, the hit area (ranging from 0 to 15; see Fig. 12), landing area (0-15; see Fig. 12), player location area (0-8; see Fig. 9), and opponent location area (0-8; see Fig. 9) to use as input for the Next Best Move algorithm.. ShuttleSet is used for our model to learn the ins and outs of badminton strategy, whereas VideoBadminton handles training the pre-existing models to extract the essential features of the videos throughout the match that we use to feed into the Next Best Move algorithm.

Badminton technique, whether it be how you hold the racket, move around the court, or strike the shuttlecock, is an essential aspect of badminton to focus on no matter what skill level you are. However, equally important is the strategy and game sense you employ to undermine your opponent and score against them. Knowing where to hit the shuttle and the type of stroke to use to hit the shuttle are two important points to keep in mind when playing badminton, yet are often a black box for newer players. There are a variety of tutorials online about proper badminton technique, but without insight from a professional, coach or otherwise, new players can often hit a wall when it comes to actually playing against an opponent you are unfamiliar with. To truly improve gameplay, it's not enough to just be able to react quickly and return the shuttlecock. Players need to actually think strategically about where and how to hit the shuttlecock in order to maximize their chance of scoring against their opponent. For these reasons, we developed an AI-powered badminton coaching system that takes a video input to analyze in-game strategy and provide feedback in order to help build up game sense and guide players through learning general badminton strategy.

3 Applications

This AI-powered badminton coaching system has several important applications in both individual and professional training. For individual players, the system provides a cost effective way to receive feedback without relying on in person coaching. By uploading videos of their gameplay to the app, players can review segmented clips in the video that include feedback on the Next Best Move that should be played in the sequence. In low resource communities, where access to professional coaching is limited or unaffordable, this system offers a useful alternative. The mobile app is designed to be functional on standard devices, making it suitable for users without high end hardware. Additionally, the app stores past analyses and video segments locally, reducing the need for constant internet access. This increases the system's usability in rural or even in international areas where connectivity could be a problem.

For competitive players, the system can function as a post match analysis tool that can provide deeper analysis into gameplay. After a tournament, athletes can upload match footage and receive a breakdown of all the moves that they played. This can help players understand their strengths and find their weaknesses. Additionally, they could analyze their opponents' patterns and can prepare for more effective strategies for future matches. Coaches could also use this system to track the progress of their players over time. Instead of relying solely on subjective assessments during live sessions, coaches can refer to this objective data that is being gathered. This will help them give further feedback on what the app is already doing and focus on the nuanced details.

This app is meant to guide players through a continuous learning loop. Receiving feedback from the app is not the final part of their improvement, it's actually the very beginning. After a user receives specific feedback, such as "You should have dropped the shuttlecock to

section 2 instead of smashing it to section 12," the goal is to take that advice back onto the court. Players are encouraged to recreate those scenarios during their practices, apply the recommended strategies, and record the new gameplay footage. This uploaded video can be fed back into the system to evaluate whether the adjustments led to more successful outcomes or not. Over time, these feedback loops will lead to deeper understanding and more effective decision making. This concept is similar to how new drivers learn, although they may initially understand the rules in theory, real ability only develops from repeated exposure to situations such as learning how to merge into fast moving lanes or how to watch out for the car in front of them suddenly breaking. In the same way, this system helps players recognize patterns in their games. Learning when to play defensively and when to attack comes from real experiences. Eventually, users begin to internalize good choices and build their "game sense" that allows them to automatically know the right move, no longer needing to ask the app for advice.

Beyond coaching, this system supports research and development in sports technology and artificial intelligence. It can be used to generate structured data from user's videos which could itself create a dataset for models that could be developed in the future. Additionally, the system can potentially be integrated with external sensors such as smart rackets or other types of wearable devices to create an even deeper and more precise system. This idea could also be used with other types of sports such as Tennis or Volleyball with the appropriate dataset.

4 Methods

4.1 Player Tracking

Each video that is input into the app is split into multiple frames at around 10-15 frames per second. Each frame is then passed through the player tracking algorithm to extract the information required for the Next Best Move algorithm, player_location, opponent_location.

Court Detection

The first step after a video is input into the app is identifying the exact boundaries of the badminton court. The process begins by applying the Canny Edge Detection algorithm (Fig. 4). This method finds the areas of the image where there is a sudden change in pixel's brightness or color, essentially marking the outlines of all the objects in an image. Then, the Hough Line Transform algorithm is used (Fig. 4), which scans the detected edges and finds the straight lines. In the context of a badminton video, these straight lines typically correspond to the boundaries of the court. From the set of detected lines, we isolate the topmost, bottommost, leftmost, and rightmost lines that form the outermost boundaries of the court. By calculating where these lines intersect, we obtain the four corner points of the court (Fig. 5).

```
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

_, binary_image = cv2.threshold(gray_image, 190, 255, cv2.THRESH_BINARY)

edges = cv2.Canny(binary_image, 100, 200)
lines = cv2.HoughLinesP(edges, 1, np.pi / 90, 90, None, 10, 250)
```

Figure 4. Code Snippet: Canny Edge Detection Followed by Hough Lines

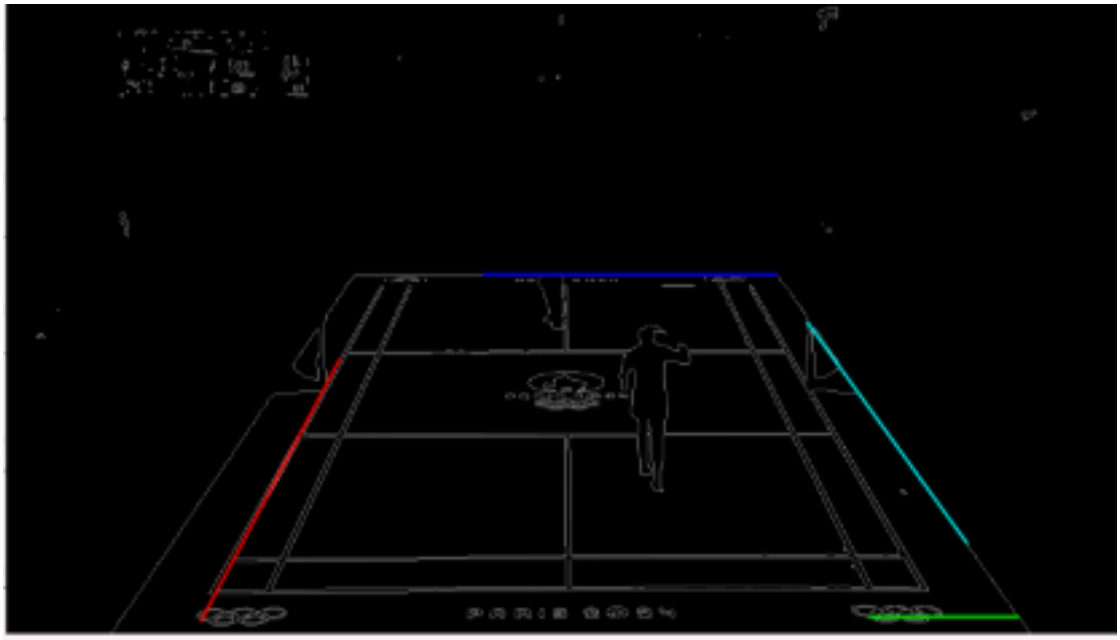

Figure 5. Hough Line Transform

Human Detection and Pose Estimation

Once the court is identified, the next step is detecting the people within the frame. This is done by using YOLOv11, a real time object detection model known for its speed and accuracy. YOLOv11 is capable of identifying all the people in the frame, which may include not only the players but also referees, coaches, and spectators. YOLOv11 has a pose detection model called OpenPose which detects (14) key points including joints and limbs (Fig. 6).

```
model = YOLO('yolo11m-pose.pt')
results = model(source=filename, show=False, conf=0.3, save=True)
keypoints = []

for index, result in enumerate(results):
    frame = []
    for i, pose in enumerate(result.keypoints):
        player = []
        for index, keypoint in enumerate(p:=(pose.xy[0])):
            x, y = keypoint[0].item(), keypoint[1].item()

            if is_in(equations,(x,y)):
                player += [(x,y)]
```

Figure 6. Code Snippet: Player Keypoint Detection using OpenPose and OpenCV

Filtering Non Players

Having the key points of every single visible person is not needed for our purpose, as we are specifically interested in identifying the two players. To filter out non players, we use the court boundary that was determined earlier and slightly extend the bounding box upward to accommodate players who might be jumping during gameplay. We check whether each detected person falls within this expanded region and filter out the ones who do not (Fig. 7).

Figure 7. OpenPose Keypoint Detection

Player Localization

After the players are successfully identified, we need to determine their location to send to the next best move algorithm. To do this, the court is first divided into a simple 3x3 grid, splitting it into 9 sections. Each player's location is determined by calculating the midpoint between their last two key points which are their left and right foots and mapping the result to one of the nine sections (Fig. 8). The sections are indexed from 0 to 8 in a left to right, top to

bottom order with 0 representing the top left and 8 representing the bottom right when viewing the court from a bird's eye perspective (Fig. 9). For every frame, the algorithm stores the section index for each player under the variables player_location and opponent_location which is sent as inputs to the Next Best Move algorithm.

Figure 8. Player Location Calculation

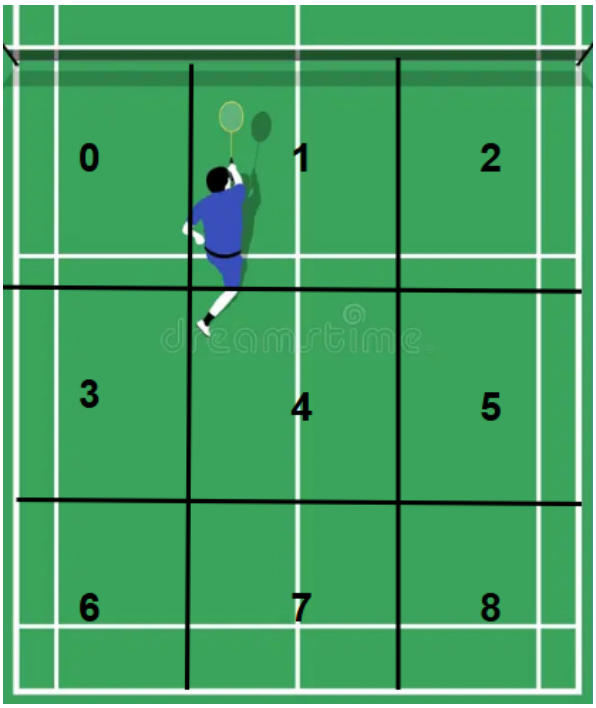

Figure 9. Court Sectioning for Player Locations

4.2 TrackNet

TrackNet is a deep learning network for high-speed small object tracking [5]. Taking multiple consecutive frames as input, Tracknet learns object and trajectory tracking to elevate positioning capabilities. TrackNet takes consecutive frames as input in order to analyze the movement of small objects across the video. Using that information, TrackNet predicts the location of the small object, in this case a shuttlecock. The structure of TrackNet consists of VGG16, a sixteen layer convolutional neural network, and DeconvNet which is a deconvolutional neural network (Fig. 10).

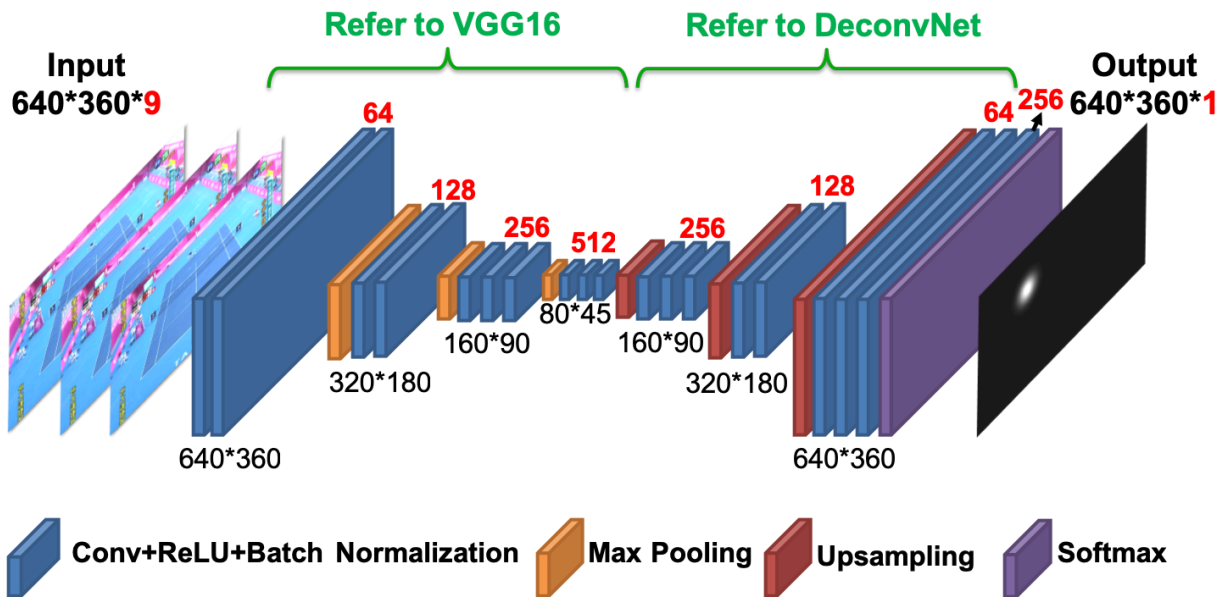

Figure 10. TrackNet Structure

We trained a TrackNet model on the VideoBadminton dataset for shuttlecock tracking purposes. TrackNet takes a video input, in this case a badminton match, and produces a gaussian heatmap centered on the model's predicted location of the shuttlecock at each frame (Fig. 11).

Using this TrackNet model, we extract information about the trajectory and location of the shuttlecock as a key datapoint for any given video input and assign it a section between 0-15 (Fig. 12) for the Next Best Move Algorithm.

Figure 11. Shuttlecock Tracking using TrackNet

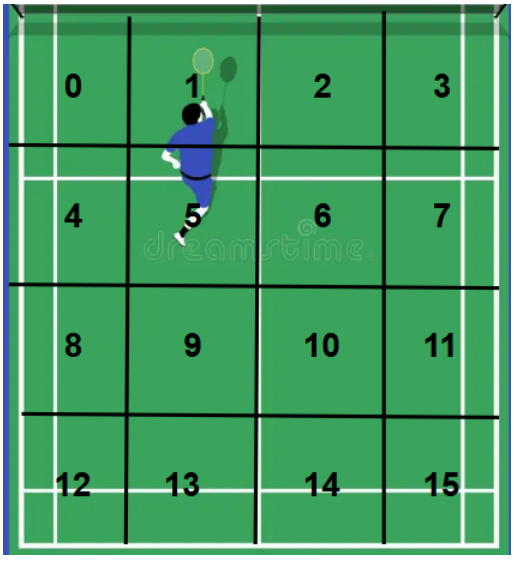

Figure 12. Court Sectioning for Shuttlecock

4.3 SlowFast

The SlowFast model, developed by researchers at Facebook AI, is a dual-pathway neural network architecture designed for video action recognition [6]. Its central innovation lies in processing video at two distinct temporal resolutions, allowing it to effectively capture both slow and fast dynamics in motion.

The Slow pathway operates at a low frame rate, focusing on spatial semantics and contextual features that evolve gradually over time. This enables the model to interpret high-level elements in a video, such as object identity and scene layout. In contrast, the Fast pathway processes a higher frame rate with fewer spatial channels, emphasizing motion and fine-grained temporal patterns. This stream captures rapid movements that might be missed by the slower pathway alone (Fig. 13).

By fusing features from both pathways, the SlowFast model achieves a balanced representation of both spatial and temporal information. This architecture has shown significant performance improvements on standard video recognition benchmarks and has contributed to advancements in areas such as human action detection and gesture recognition.

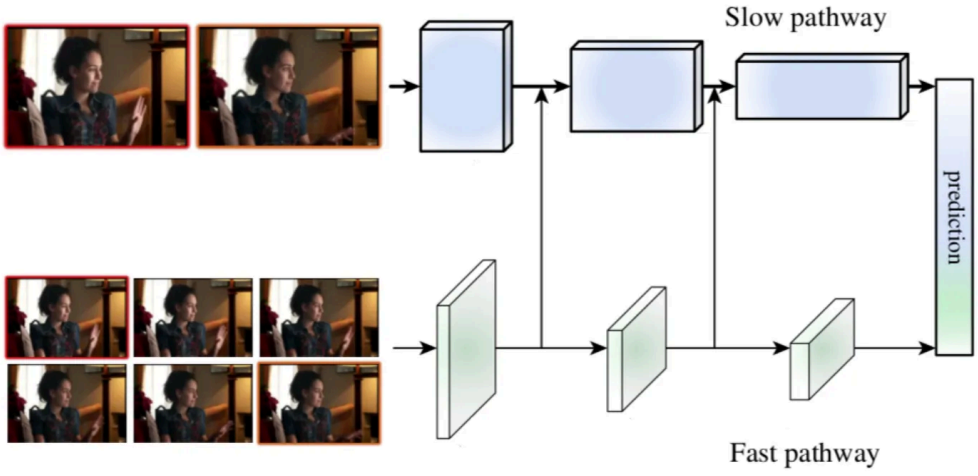

Figure 13. Illustrative Example of the SlowFast Network

For our purposes, we trained a SlowFast model on the VideoBadminton dataset to identify each stroke the user does throughout the match. SlowFast takes a video, in this case the user's badminton match, and identifies the stroke the player is using. Our algorithm temporally segments inputted videos by each stroke, and pairs that with the information extracted from TrackNet and YOLOv11 to use as an input for the Next Best Move algorithm (Fig. 14).

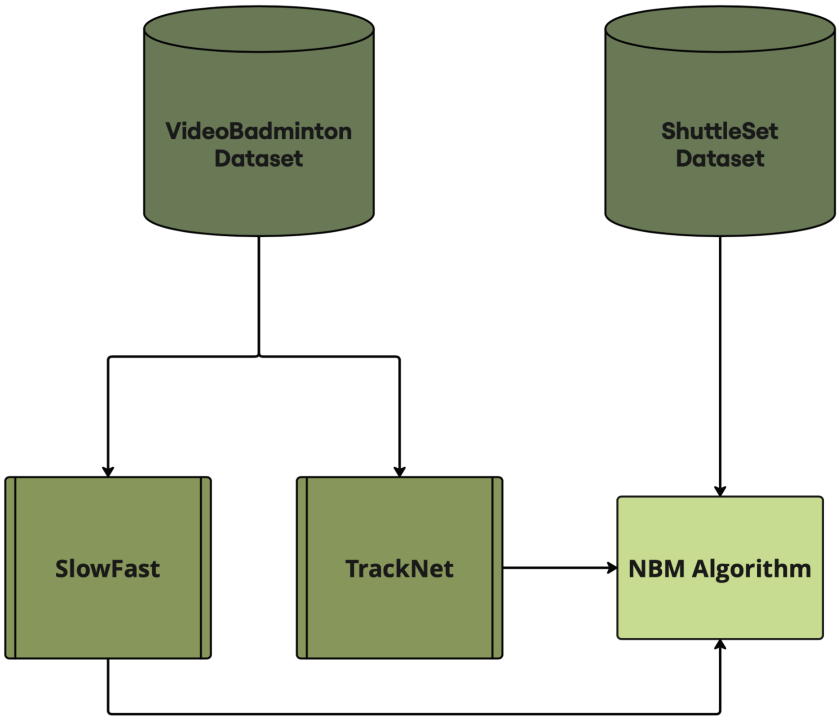

Figure 14. Flowchart of Training

4.4 Next Best Move Algorithm

To decide what move a badminton player should make next, we built a feedforward neural network using Pytorch [5]. This is a type of artificial intelligence model that learns patterns from data and uses them to make decisions. The model begins with an input layer, which

takes in a set of five pieces of information. These include the player's current location on the court, the opponents current location on the court, the section of the court where the shuttlecock was hit, the expecting landing section of the shuttlecock, and the badminton action that the player just performed (Fig. 15). These five inputs are packed into a 5x1 vector, representing a full snapshot of what is happening in the game at one moment. This vector is passed through hidden layers, which are the parts of the model where most of the learning takes place.

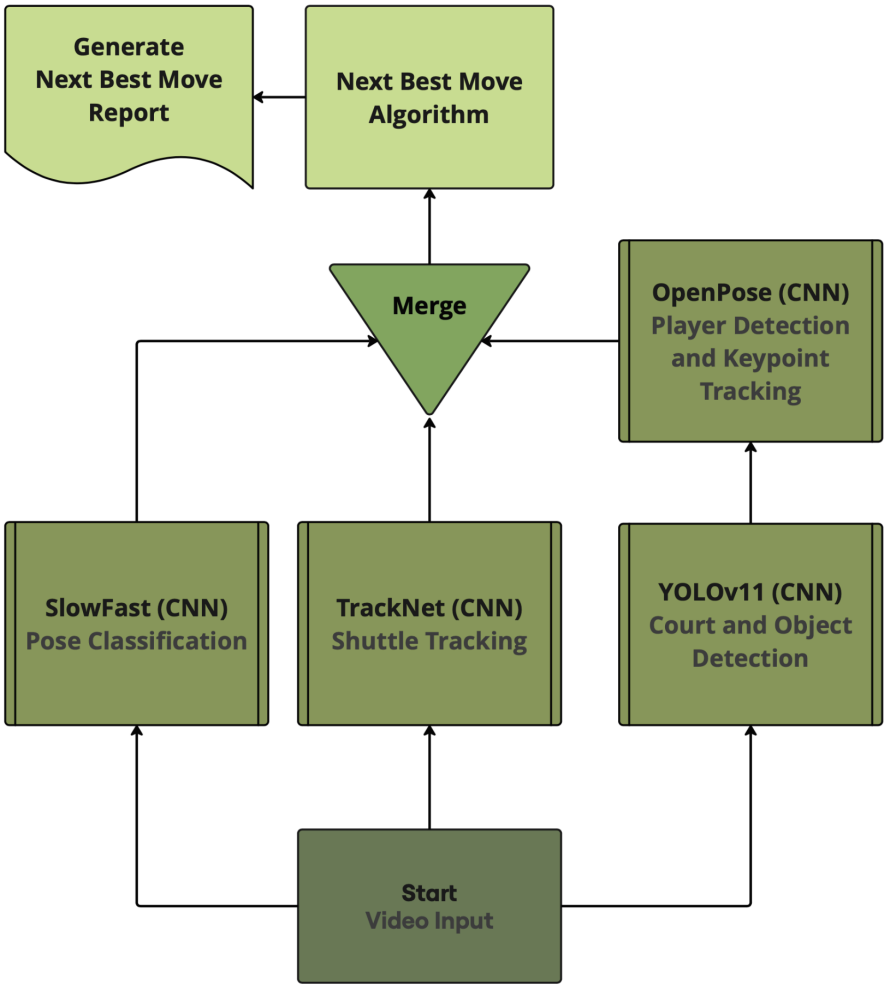

Figure 15. Flowchart of Model

Between each hidden layer, we use a ReLU activation function (Fig. 16). ReLU helps the network learn faster by solving a problem known as the vanishing gradient. That problem happens when a model learns too slowly or stops improving even though the errors are still high. ReLU keeps the signals strong as they pass through the layers, which makes the model more stable and easier to train.

```
nn.Sequential(
# Input: [shot, hit_area (0-15), land_area (0-15), player_location_area(0-8), opponent_location_ar
    nn.Linear(5, 32),
    nn.ReLU(),           # Prevent vanishing gradient problem
    nn.Linear(32, 64),   # Hidden layer 1
    nn.ReLU(),
    nn.Linear(64, 64),   # Hidden layer 2
    nn.ReLU(),
    nn.Linear(64, 32),   # Hidden layer 3
    nn.ReLU(),
    nn.Linear(32, 2)     # Output Layer: [next best move, next best location]
)
```

Figure 16. Code Snippet: NBM Algorithm Neural Network Model

We chose to use exactly three hidden layers based on experiments and testing. When we tried models with only one or two hidden layers, they did not learn enough. They were too simple and missed many more important patterns in the game. This phenomenon is called underfitting. On the other hand, when we added more than three layers, the model started to overfit, meaning it memorized the training data rather than learning how to generalize new solutions. It is similar to a student who memorizes homework answers but doesn't actually learn how to do the problem so they end up doing poorly on a test with slightly different questions. Three layers turned out to be the best balance, the network was deep enough to find important patterns but not so deep that the model became too specific.

Each hidden layer has a different job. The first layer learns simple patterns like where the players are standing and what shots they are using. The second layer starts to understand the interactions between the players and their shots. For example, it teaches that the opponent is

standing near the net and the player hits a high lift, the opponent can smash it back easily. The third layer uses all of this knowledge to make smart decisions. It connects positions and interaction to the most useful next move. For example, it may figure out that hiding a clear to the backcourt is the safest option because it gives the player time to recover.

After passing through all three hidden layers, the model reaches the output layer. This layer produces a 2x1 vector. The first value is the move that the player should make next and the second value is the location on the court where the player should aim the shuttlecock. These two values together form the model's recommendation.

4.5 Mobile App

To build and test the app, we design a simple interface with two main options: "Next Best Move" and "Archives" using Kivy. When the app is opened, the user sees a logo and these two buttons (Fig. 17). Pressing "Next Best Move" prompts the user to upload a video of their badminton gameplay. Once uploaded, the app sends the video to the process above that eventually leads to the Next Best Move algorithm, which then processes the footage and generates a series of videos, each showing a clip of a single shot made by the player.

Figure 17. SmartRally Start Screen

For each clip, the app displays visual feedback: a red dot shows where the player actually aimed and a green dot shows where they should have aimed. A bounding box also appears around the player's body which is green if their action matches the recommended shot, and red if it does not. Each video also has a caption that explains the snapshot (Fig. 18). If there is a mistake, it will say: "IMPROVEMENT: The best possible shot is *NBM output one,* but you played *current played* instead" and "IMPROVEMENT: The best possible direction is section *NBM output two,* but you played section *current aim* instead." If the players makes the correct move, the caption will say: "CORRECT: Best possible shot is *current played*" and

"CORRECT: Best possible direction is section *current aim* of the opponent's court."

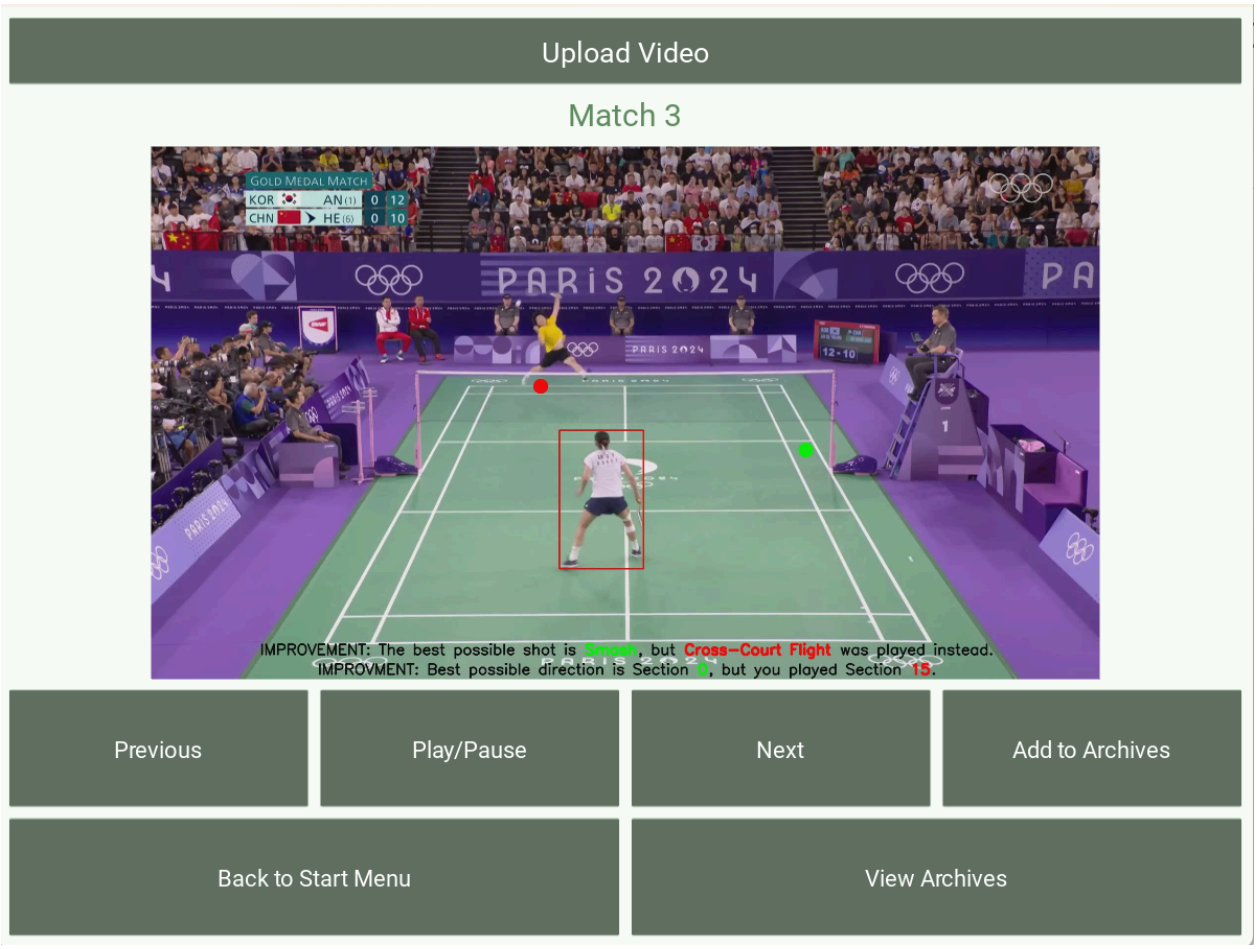

Figure 18. SmartRally's Next Best Move Feature

There are also buttons that allow the user to move back and forth between the clips, reply to them, and study the feedback as many times as they want (Fig. 18). When they're done for the time being, they can choose to save the generated video by giving it a name (Fig. 19). Saved videos go to the "Archives," which stores them locally on the device. Pressing the "Archives"

button lets users view any of their saved videos. From there they can choose a video to watch again or delete a video if they no longer need it (Fig. 20).

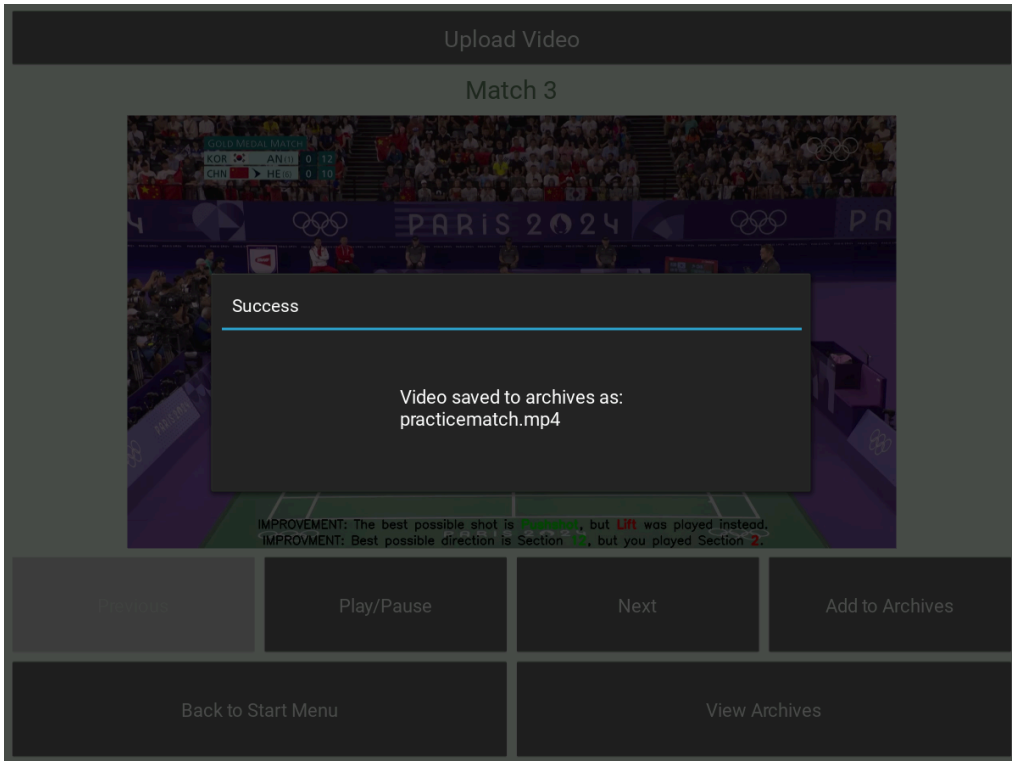

Figure 19. SmartRally's Saving to Archives Feature

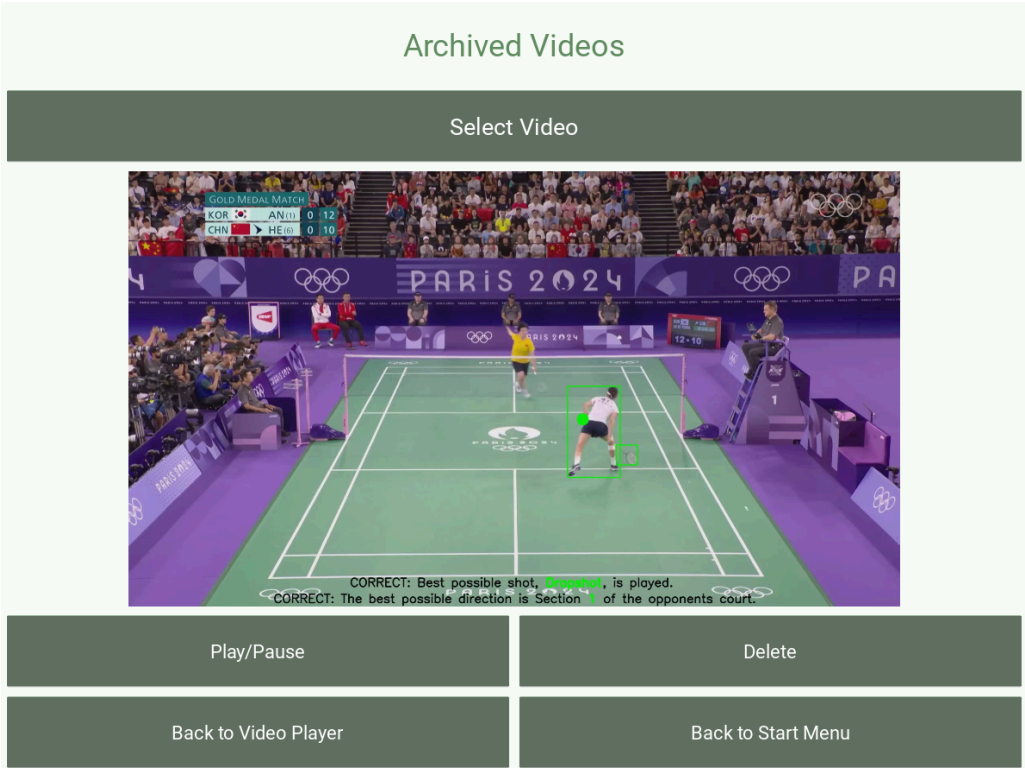

Figure 20. SmartRally's Archives Feature

5 Results

On the VideoBadminton dataset, SlowFast achieved a top-1 accuracy of around 72%. In 737 rallies and 7,298 strokes for 27 of the top-ranking singles players in matches from 2018 to 2021, the Next Best Move Algorithm achieved an accuracy of approximately 70%. Given that there are hundreds of different ways to play any given match, and 18 different strokes to choose from at any given time, being able to predict the stroke that promises a consistently good chance of scoring against your opponent 7/10 times is great. The SmartRally advice is not absolute, and there are a variety of other factors SmartRally does not consider that are important to predicting how a match will go such as overall fatigue, individual player strengths and weaknesses, technique, body type, posture, and more. However, SmartRally gives the stroke with the most consistent results. With the use of SmartRally, users can gain an intuitive understanding of the

types of strokes to use depending on the shuttle trajectory and player positions. Of course, to elevate skill, players should seek proper training by a professional, however SmartRally serves as an accessible resource for those who want to learn, but don't have the resources to start.

6 Limitations

The most glaring limitation to SmartRally is accuracy of our SlowFast model. With 72% accuracy on SlowFast, that means that our input is fundamentally not accurate more than a third of the time. An inaccurate input will lead to, obviously, an inaccurate output not only giving the user advice that is not useful, but potentially leading to confusion or misunderstanding in core badminton strategy. A more minor problem has to do with our TrackNet model. We trained a TrackNet model on the VideoBadminton dataset, however the VideoBadminton dataset did not have data on the location of the shuttlecock at every given frame. Because of this, we used an AI annotating tool on RoboFlow, which had only around a 70% accuracy. Resultantly, our TrackNet model struggled to accurately predict the shuttlecock trajectory, or be unable to detect it at all at certain points throughout a rally. This only became an apparent problem when the shuttlecock was near the net, however, and did not have a large effect on the annotation the input video to our algorithm since we only took the location of the shuttlecock when it was hit by the player, their opponent, or when it landed on the ground—which TrackNet excelled at detecting.

7 Conclusion

SmartRally was created to make badminton training smarter and more accessible. Instead of needing a coach by your side, players can now receive meaningful feedback just by uploading a video. The app watches and understands the game, then responds with insights that help the

user grow. SmartRally is designed to act as a virtual coach that is fast, consistent, and always available. The goal is to bridge the gap between causal players and professional level analysis. Many athletes, especially those in low income communities, don't have access to constant coaching or expensive training programs. SmartRally helps close that gap by offering advice based on real patterns seen in high level gameplay. It encourages learning by showing players what usually works best, not just what happened in one moment. Looking ahead, SmartRally has the potential to go even further. In the future, it could help correct shots or track a player;s progress over time. But even now, it already gives users a strong foundation for improvement.

8 Future Work

SmartRally is not absolute, and only gives advice based on generalized and consistent results which we extracted from the ShuttleNet dataset. For future improvements, we suggest first addressing the issues faced in our limitations: enhancing the SlowFast model to produce more accurate predictions of badminton strokes, and finding better annotating tools—or manually annotating every frame in the VideoBadminton dataset—in order to produce a more accurate TrackNet model. Although ambitious, applying the same concepts of the Next Best Move algorithm to a pair of AR smart glasses would be an incredible achievement in sports analysis. Instead of having to play through an entire match and review SmartRally’s advice post-match, you could instead get real time advice and feedback while in the middle of a match. Players could learn on their feet, reinforcing the coaching advice and ingraining it not only to their memory, but their body as well.

Another future improvement could be applying evolutionary learning, or LSTMs to the model, allowing it to adapt to a specific opponent. Every player has a different playstyle and may react differently in different situations which can affect the applicability of SmartRally’s advice

as it only gives advice based on the playstyles of many different players. A model that could learn the strengths and weaknesses of an opponent and predict which stroke would be most effective against them specifically would be a powerful tool for players to elevate their game sense and learn how to adapt to different situations.

Future works should also consider more points of data in their analysis. Our NBM algorithm only takes in five data points about the stroke and locations of players and shuttlecock. Accounting for other factors such as racket type, fatigue, and technique could give potentially more informative results and better predictions.

References

[1] "VideoBadminton: A Video Dataset for Badminton Action Recognition," *Arxiv.org*, 2024. https://arxiv.org/html/2403.12385v1#S3 (accessed May 28, 2025).

[2] Lê Công Luận and Trần Minh Triết, "Analyze video action with a deep learning model based on space-time and context information," *Analyze video action with a deep learning model based on space-time and context information*, Dec. 2023, doi: https://doi.org/10.1145/3628797.3628956.

[3] D. Toshniwal, V. Karad, A. Patil, and N. Vachhani, "AI coach for badminton."

[4] W.-Y. Wang, Y.-C. Huang, T.-U. Ik, and W.-C. Peng, "ShuttleSet: A Human-Annotated Stroke-Level Singles Dataset for Badminton Tactical Analysis," *arXiv.org*, 2023. https://arxiv.org/abs/2306.04948

[5] yastrebksv, "GitHub - yastrebksv/TrackNet: Unofficial PyTorch implementation of TrackNet," *GitHub*, 2022. https://github.com/yastrebksv/TrackNet

[6] facebookresearch, “GitHub - facebookresearch/SlowFast: PySlowFast: video understanding codebase from FAIR for reproducing state-of-the-art video models.,” *GitHub*, 2019. https://github.com/facebookresearch/SlowFast (accessed May 28, 2025).

[7] “PyTorch,” *www.pytorch.org*. https://pytorch.org

[8] Chang-Chia-Chi, “GitHub Chang-Chia-Chi/TrackNet-Badminton-Tracking-tensorflow2: TrackNet for badminton tracking using tensorflow2,” *GitHub*, 2020. https://github.com/Chang-Chia-Chi/TrackNet-Badminton-Tracking-tensorflow2?tab=readme-ov-file (accessed May 28, 2025).

[9] Rani Horev, “SlowFast Explained: Dual-mode CNN for Video Understanding,” *Medium*, Dec. 27, 2018. https://medium.com/data-science/slowfast-explained-dual-mode-cnn-for-video-understanding-8bf639960256 (accessed May 28, 2025).

[10] Rawpixelimages, “Aerial view of a badminton court,” *Dreamstime*, Jul. 18, 2018. https://www.dreamstime.com/aerial-view-badminton-court-image121537103

[11] “Miro: the collaborative whiteboard platform for distributed teams,” *https://miro.com/*. https://miro.com/app

[12] https://github.com/JacobDipz/SmartRally.git