Leveraging 3D CNNs and YOLO for Tennis Stroke Classification

Dominic Borg
Stanford University
dborg22@stanford.edu

Cameron Camp
Stanford University
dborg22@stanford.edu

Abstract

We construct an image classification model combining convolutional neural networks (CNNs) with YOLO (a computer vision model for computing object bounding boxes) in order to accurately classify tennis strokes. The aim of this project is to better enable players to study and analyze their own performance, as well as that of their opponents. Our approach was to use tennis footage as input and then use YOLO to create bounding boxes to focus on the portion of each frame containing the player. A 3D CNN was then applied in order to categorize each player's stroke during the period over which the ball is hit. Our baseline model, a vanilla 3D CNN, required an excessive amount of time to train, a problem that we aimed to alleviate by incorporating YOLO bounding boxes. We find that overall, passing the player bounding boxes obtained by YOLO into our 3D CNN improved the model's accuracy and efficiency, likely because YOLO was able to extract the most important features of the footage (the player), thereby decreasing the volume of input data while still preserving the most important, high-resolution information.

Introduction

Computer vision models can be used to allow more comprehensive analysis of athletic performance. Often, athletes want to understand, in detail, their own performance and technique as well as that of other players. They may do so as a means of gaining greater consciousness of their own playing, or as a means of better understanding the playing style of their competition. In the context of tennis, this would normally require taking footage of the player in action and manually identifying their strokes, as well as tracking the relative frequencies of each kind of stroke. For the purposes of this project, we adopt classifications for nine different strokes: top-spin forehand, return forehand, slice forehand, volley forehand, serve, top-spin backhand, return backhand, slice backhand, and volley backhand.

In order to automate the task of classifying tennis strokes given footage of players, we use a 3D CNN. This is a convolutional neural network with an extra temporal dimension that allows us to perform convolutions over multiple frames of footage. The input to our model is a segment of video, and the output to our model is a classification of which stroke the player in the video is performing. Our results demonstrate that a 3D CNN can be applied to footage of tennis players in order to classify strokes with acceptable accuracy, and that using YOLO to create bounding boxes can further speed up the process by removing unneeded features and focusing only on the relevant objects in the footage.

1.1. Results

Our results show somewhat improved accuracies when applying the first version of YOLO compared to our vanilla CNN without YOLO. Our vanilla CNN obtained a final loss of 0.82 on the test set (see Figure 1), while adding YOLO to the CNN yielded a somewhat lower loss, of around 0.79, and took substantially less time when run on the testing data. However, training the models took an extremely long time for both models, and showed lower efficiency with YOLO.

2. Related Work

We referenced several pieces of existing work that seek to use neural networks to classify sports footage. For instance, our project is a different application of the methods and objectives proposed by Xu-Hong Meng et al. In this study, the authors construct and label a dataset comprising footage of basketball players performing various techniques. In order to classify these techniques, the authors use a 3D convolutional network framework in order to track and identify "technical actions" taking place in footage of basketball players [10]. The authors initially test their approach on footage in which the players are relatively stationary, and obtain promising results while also proposing ways in which contextual information, such as the player's position on the court, can be used to improve the model's accuracy in the future. We apply a similar approach to the sport of tennis, providing a starting point that can be expanded in similar ways.

Our research also builds on that detailed in the paper "3D Convolutional Networks for Action Recognition:

Application to Sport Gesture Recognition" by Pierre-Etienne Martin et al., in which the authors adopts a similar approach to classifying moves in table tennis. This research is similar to that outlined above, although the authors also use attention modules to improve the model's efficiency and classification accuracy [6]. We also referenced "Convolutional Neural Networks for Classification of Noisy Sports Videos" by Joey Asperger and Austin Poore, a former CS 231N final project, which builds on this research by detailing the architecture and data processing pipeline for a 3D CNN meant to classify moves found in noisy sports footage [3].

We considered various methods of using bounding boxes to accelerate our model. One was using an R-CNN (regions with convolutional neural networks) model, proposed in a 2014 paper by Ross B. Girshick et al. [8]. In an R-CNN, regions of the image are proposed and then convolutional neural networks are applied in order to help optimize and classify the bounding boxes. This method significantly improved accuracy at the time of its inception, and became significantly faster in 2015 when Girshick proposed the "Fast R-CNN", which jointly classifies object proposals and refines their locations, among other improvements, thereby making the R-CNN markedly faster at test-time [7]. We also considered using a Single Shot MultiBox Detector, an alternative method proposed by Wei Liu et al. in a 2015 paper. In the paper, the authors propose predetermining a set of default bounding boxes for each feature location, then find scoresfor each and determine which boxes to use accordingly [9]. This approach offers simplicity, speed, and ease of implementation, although it performs poorly on small objects since the default bounding boxes are given large sizes.

Finally, however, we chose to implement YOLO, which we did by referencing several paper, the first of which was "You Only Look Once: Unified, Real-Time Object Detection" by Joseph Redmon et al [5]. This paper, published in 2015, proposes the first version of YOLO, standing for "You Only Look Once," referencing the fact that a single pass over the image is used to detect and bound objects, a key aspect of functionality that gives YOLO its high computational efficiency. Indeed, the authors write that their full YOLO architecture is capable of processing images in real time at 45 frames per second [5]. This architecture makes multiple predictions about bounding box locations, and uses a single convolutional network, giving the network its appealing simplicity and efficiency. Later research has expanded on and modified this approach in various ways. For instance, Joseph Redmon and Ali Farhadi, who worked on the initial YOLO paper, proposed in 2016 an improved version of YOLO called "YOLO9000", which included several modifications and augmentations to YOLO that made it work even faster with over 9,000 different classifications [4].

Most of the YOLO-related implementations after YOLO9000 have been incremental in nature. For instance, a 2020 paper by Alexey Bochkovskiy et al. proposes a fourth version of YOLO and makes several modifications to the YOLO model, including self-adversarial training and the use of CIOU loss, to help further optimize the

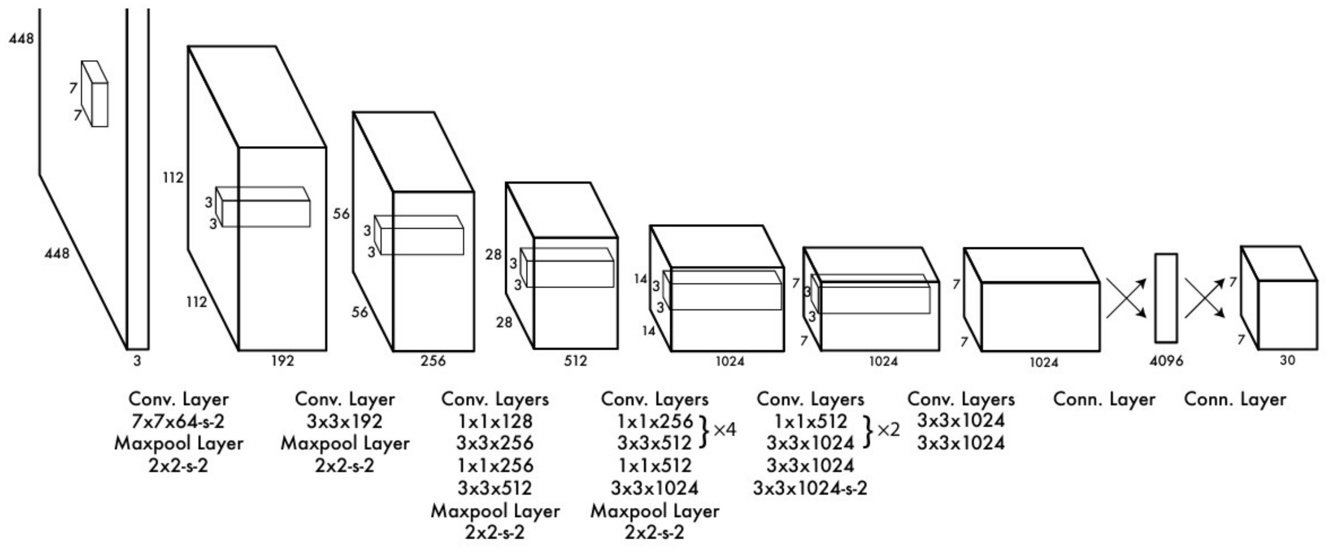

Figure 1: Proposed network architecture for YOLOv1 [5].

model [1]. The current state-of-the-art version when it comes to YOLO is “YOLOv7”, proposed in 2022 by Bochkoviskiy et al., an updated model that manages to outperform other state-of-the-art detectors in both speed and accuracy by introducing various low-cost improvements into the model bag-of-freebies style; compared to then-state-of-the-art models, YOLOv7 showed speeds that were faster by factors as high as 500% [2]. This version of YOLO is the current standard in neural-network-based object detection.

3. Data

In order to train our vanilla 3D-CNN for stroke classification, the project utilizes a dataset featuring labeled tennis strokes and the time-stamps at which they occur in the 2019 Australian Open Final. To prepare the data to train the model, the match footage is cycled through, extracting the frames at these time-stamps as well as the frames within 1 second before and after the marked frame and reducing their dimensions to 64x64 for faster convolution. The resulting processed data takes the form of sequential frame “blocks” (with dimensions H * W * T) from which the model can extract the spatial-temporal information associated with each stroke. Each frame block is assigned a true classification label as one of nine possible shot descriptions: “Topspin Forehand” (0), “Return Forehand” (1), “Slice Forehand” (2), “Volley Forehand” (3), “Serve Forehand” (4), “Top-spin Backhand” (5), “Return Backhand” (6), “Slice Forehand” (7), “Volley Forehand” (8). The frame blocks and their labels are randomly assigned to batches of size 32 via NumPy’s Dataloader capabilities.

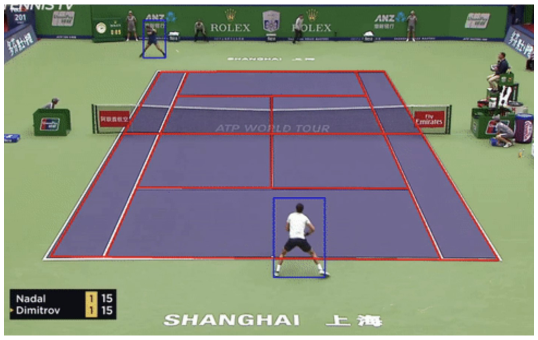

Figure 2: Labeled tennis footage with bounding boxes around the players.

We also used the dataset “Tennis ball” from user Conrad Takasi on Kaggle in order to train the model to identify and track the movements of the tennis ball in the test footage. This data was already labelled and did not require additional labelling or processing.

Finally, in order to identify the position of the player and construct a bounding box around them, we used the dataset “Human Tracking & Object Detection Dataset” from Kaggle user Training Data. The data contains 40 of various groups of people, each annotated with a set of

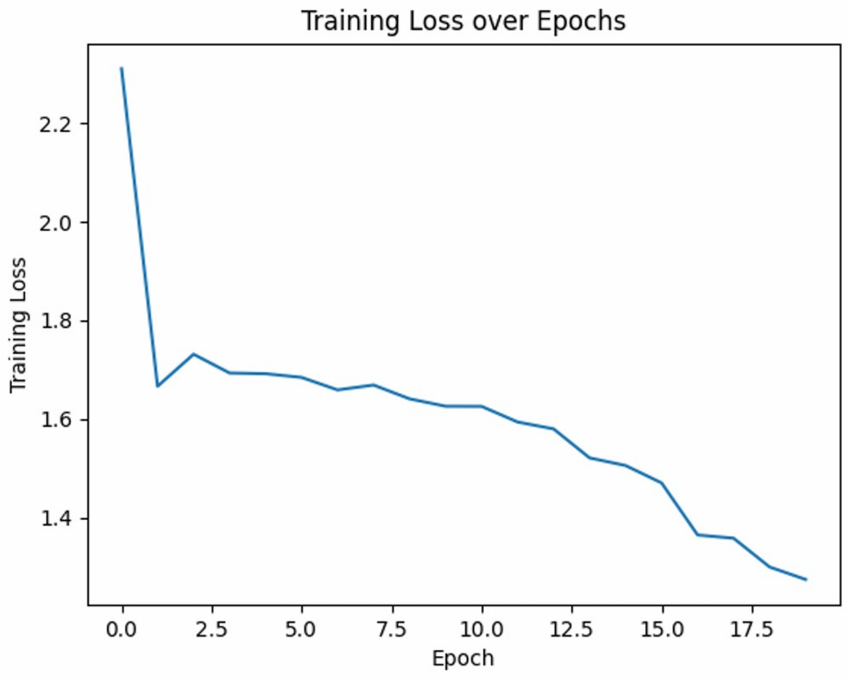

Figure 3: Training loss for vanilla CNN for tennis ball tracking.

rectangular bounding boxes that surround the people in the photo. This data was used to train our model to accurately place bounding boxes around the players in the footage using YOLO, thereby reducing data volume and computation time at test-time.

For YOLO, once we found our dataset, we imported the bounding-box annotations as .xml files, in which each box's lower-left and upper-right coordinates are specified under a child node in the xml tree. We parsed through these files in order to reconstruct the bounding-box coordinates, and used these coordinates to train our YOLO CNN.

4. Methods

Our baseline model is a 3D convolutional neural network with dimensions for height, width, and temporality. The architecture of the 3D CNN comprises three convolutional layers, each followed by a ReLU activation and max-pooling layer to progressively extract and downsample spatio-temporal features from the input video blocks. The extracted features are then flattened and passed through two fully connected layers, the first of which includes a dropout layer for regularization, reducing the risk of overfitting. The final layer outputs the class probabilities for the given input sequence, and weights are updated using Adam optimization. This approach integrates advanced deep learning techniques to effectively classify tennis strokes, leveraging the spatio-temporal dynamics captured in video sequences for comprehensive performance analysis. The baseline objective is to achieve a low average loss on the test data.

In order to improve the model's test-time speed, as well as potentially improve accuracy, we decided to add bounding boxes to our model. Of the aforementioned methods for implementing bounding-box generation, we decided to choose YOLO for our model. We chose YOLO because of its relative computational simplicity, and we opted for the first version of YOLO simply to assess whether YOLO can have an impact on the model's performance, with other proposed optimizations serving to refine the model in the future. For the CNN for our YOLO implementation, which determines bounding boxes for both the player and the ball, our architecture consists of a single convolutional neural network, the architecture of which consists of 24 convolutional layers, 4 Max Pool layers, and two fully connected layers (Figure 1). This closely follows the architecture outlined in the original paper for YOLOv1. When training this convolutional layer, we use a loss function specifically designed for detection problems. The loss function, first outlined by Redmon et al. in [5], is as follows:

$$\begin{aligned}
&\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}$$

When implementing YOLO on the training data, we had to choose hyperparameters that optimized our ability to accurately detect objects. This included running the model with different hyperparameters for the CNN portion of YOLO, including learning rate and batch size, as well as selecting values for YOLO-specific hyperparameters, notably the confidence threshold, which determines how confident we must be about whether an object is detected before we report it as a positive, and the non-maximum-suppression threshold, which determines which bounding boxes will be eliminated as redundant. For the 3D CNN, we compromised between efficiency and accuracy by choosing a batch size of 16 and a learning rate of 2e-5. We chose values similar to those used by Redmon et al., setting the confidence threshold to 0.5 and the non-maximum-suppression threshold to 0.4.

5. Experiments

5.1 Hyperparameters and Optimization

From our experiments, we chose a learning rate of 2e−5 to use with the Adam optimizer. This choice was motivated by our initial trials, for which higher rates were resulting in unstable training, and lower rates were leading to very slow convergence. We set the mini-batch size to 16 due to memory constraints and the practical consideration of achieving a balance between noise in gradient estimates and training stability, as well as training time. From graphing the model's loss against the number of epochs, the loss appeared to plateau at around 20 epochs. Because of this and for training time considerations, we chose to use 20 epochs for training. Cross-validation was performed using an 80-20 split for training and validation sets, providing a robust estimate of model performance.

5.2 Primary Metrics

Our primary metrics for evaluating the YOLOv1 model are Mean Average Precision (mAP) and Mean Squared Error (MSE). mAP is calculated as:

$$mAP = \frac{1}{|Q|}\Sigma_{q\epsilon Q}AP(q)$$

where QQ is the set of queries, and AP(q)AP(q) is the Average Precision for query qq. MSE is calculated for bounding box regression and object confidence predictions to assess the alignment and confidence of the predicted boxes against the ground truth.

5.3 Results

The YOLOv1 model for player tracking achieved a final training loss of 2.51 and a validation loss of 8.07 with corresponding mAPs of 0.45 for training and 0.42 for validation. The YOLOv1 model for tennis ball tracking achieved a final training loss of 16.45 and a validation loss of 17.23, with mAPs of 0.51 for training and 0.48 for validation.

The final training and validation accuracies for the 3D CNN used for stroke classification based on the bounded player boxes passed from the YOLOv1 player tracking model were 0.62 and 0.58 respectively. The final validation accuracy using the player boxes was marginally better than the original validation accuracy of the vanilla 3D-CNN, which achieved an accuracy of 0.58.

To represent the performance of our model visually, we utilized class visualizations and confusion matrices to further analyze the performance of our model. The visualizations for the different object classes portrayed the model's general ability to identify players and tennis balls in various match scenarios. Through the use of confusion matrices, we noticed certain areas of confusion between similar classes (different types of strokes). The model somewhat regularly confused the previous player's shot with the current player's shot, likely related to inaccuracies/miscalculations in the tennis ball's trajectory.

5.4 Discussion

The results of our experiment indicate that the 3D-CNN with bounding box inputs from the YOLOv1 models marginally outperforms the vanilla 3D-CNN. The YOLOv1 models for both player and tennis ball tracking performed suboptimally, despite our efforts to modify our code and prevent the stereotypical pitfalls of machine learning models, such as exploding gradients, for which we introduced batch normalization. The low recorded MAPs on the validation set for player tracking and for tennis ball tracking suggest that the models may struggle to generalize.

The frequent confusion between a player's stroke with the previous players due to ball trajectory inaccuracies requires further research and experimentation to improve. Improving the trajectory calculation by incorporating

temporal information or using a more sophisticated tracking algorithm may reduce these errors. But perhaps more importantly, enhancing the robustness of the model through data augmentation and incorporating more diverse training data would likely mitigate issues like overfitting and help improve generalization. Ultimately, a model with more accurate bounding boxes for tennis ball tracking would improve the accuracy of the ball location estimates and therefore the trajectory predictions.

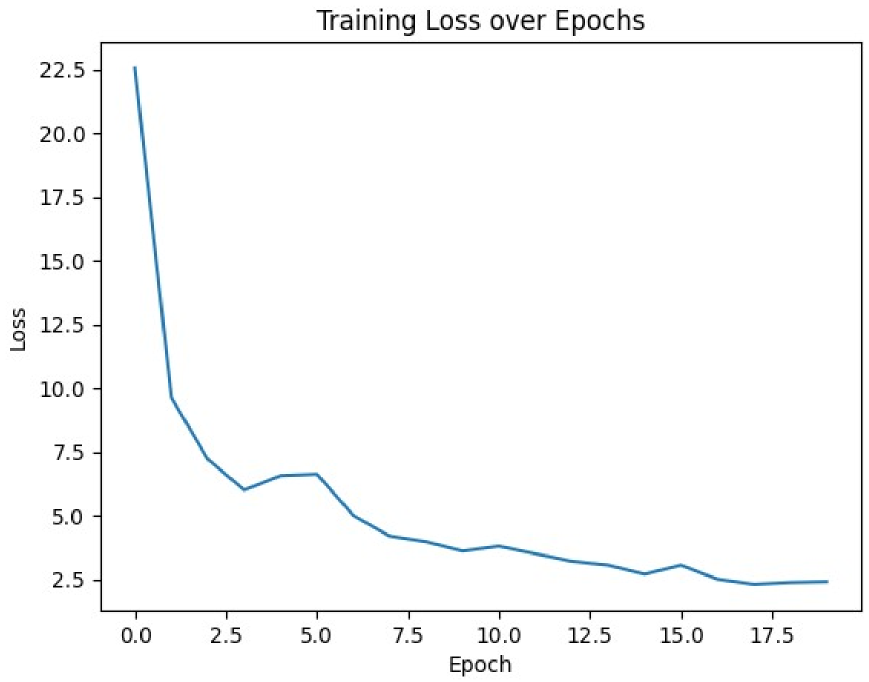

Figure 4: Training loss for YOLO for tennis ball tracking.

Additionally, to test the performance of the YOLOv1 models in different match contexts, we passed in a small set of hand-labeled images from unseen match contexts (different court colors and players). We noticed that the player bounding box model performed noticeably worse when processing these frames than those in the training/validation sets. This likely stems from overfitting to the specific visual characteristics of the frames (i.e. court surface) in the training data. When the model encountered a court with different coloring for example, its ability to accurately detect players diminished. This issue underscores the importance of further training on a diverse dataset featuring a wide array of courts and players.

6. Conclusion

Our experiments suggest that adding YOLO to a 3D CNN classifying tennis strokes can yield improvements to accuracy and speed at test-time. We implemented YOLOv1, the initial version of YOLO proposed by Joseph Redmon et al. in [5] on the subject, and obtained promising results. In the future, our model could be improved by incorporating the optimizations and updates proposed by later researchers, such as alterations to the backbone of its network or any of the small, incremental improvements proposed in each subsequent paper. To this point, there is room for more experimentation with hyperparameters.

Moreover, labelled data related to tennis footage is scarce, and the limited existing options when searching for data likely had a limiting effect on the performance of our model. Our training data was limited to footage of a single match, and future data could increase the diversity of the matches shown, as well as including matches incorporating different players, times of day, locations, and camera angles. In the future, in order to improve such models' robustness and prevent overtraining on unwanted aspects of the footage, it would likely be worthwhile to label a greater, more diverse set of data specifically tailored to these tasks. As future researchers explore avenues of improving models' performance, the might also work on gathering and labelling new data, and adding this data both to their own models and to the existing body of datasets for future researchers to use.

Finally, there exists significant potential for future research to modify the basic architecture of our classifier. For the sake of comparison, we use a vanilla 3D CNN that we then modify with YOLO, but there is room for future research to experiment with architectures, parameters, and even more significant modifications to the model. One such modification that could impact the model's performance in interesting ways would be a transformer. A transformer's built-in attention mechanisms could help the model focus on the most important parts of a player's motion, and given their known efficiency with sequential tasks, could also help the model process the video's temporal information efficiently. Overall, our results provide a simple, but promising baseline for future sport-technique classification tasks, and offer numerous opportunities for expansion, experimentation, and improvement.

References

[1] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao. YOLOv4: Optimal speed and accuracy of object detection." arXiv, 2020.

[2] C. Y. Wang, et al. YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors. arXiv, 2207(02696), 2022.

[3] J. Asperger and A. Poore. Convolutional Neural Networks for Classification of Noisy Sports Videos, 2017.

[4] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger." arXiv, 2016.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection." arXiv, 2016.

[6] P. E. Martin, J. Benois-Pineau, R. Péteri, A. Zemmari, and J. Morlier. 3D convolutional networks for action recognition: Application to sport gesture recognition. Multi-faceted Deep Learning, 2021.

[7] R. Girshick. Fast R-CNN. arXiv, 2015.

[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv, 2014.

[9] W. Liu et al. SSD: Single shot MultiBox detector. 9905: 21–37, 2016.

[10] X. H. Meng, H.-Y. Shi, and W.-H. Shang. Analysis of basketball technical movements based on human-computer interaction with deep learning," Computational Intelligence and Neuroscience, 2022(1): 4247082, 2022.