Tennis Ball Detection in Live Professional Tennis Matches: A Comparative Study of YOLOv8 and Mask R-CNN for Precision

Joshua Glaspey
College of Engineering and Computer Science
University of Central Florida
Orlando, FL, USA
joshua.glaspey@ucf.edu

***Abstract*—Object detection is a core domain within machine learning. It encounters unique challenges when applied to sports, particularly when it is applied to swing motions – particularly with swiftly moving, small area objects. This research focuses on the specific task of tennis ball detection within frames of professional matches. Tennis balls, with their diminutive size and speeds exceeding 100 miles per hour, demand precise localization. The methodology involves collecting training and testing data, including over 2000 combined images of generic tennis ball images and frames extracted from professional points sourced by the Association of Tennis Professionals (ATP). Two separate solutions will be studied: Mask R-CNN and YOLOv8. Both of these models aim to address the challenges posed by the distinctive characteristics of tennis balls. While the study pushes questions to be further addressed, its significance lies in contributing insights to the effectiveness of these different models towards high velocity object tracking.**

Keywords—tennis ball, object detection, CNN, Mask R-CNN, YOLOv8, precision, computer vision.

I. Introduction

In the recent years, the intersection of computer vision and sports analytics has produced considerable advancements, transforming the way we perceive and engage with live sporting events. One intriguing application within this domain is the detection of sports objects, a subset of object detection tasks in machine learning. This research emphasizes the importance of being able to maintain consistent tracking of an object with a high velocity and low area across live video feed.

Object detection, a popular feat of computer vision, has witnessed extensive exploration and innovation. Particularly, sports ball detection introduces unique challenges, as it demands the identification of objects identified by high velocities and small sizes. This paper covers the investigation of the complex object detection problem during live professional match play. Tennis balls, with a surface area of approximately 19.63 square inches, travel across the court at speeds varying to over 100 miles per hour. The intrinsic difficulty of this task lies not only in the size and speed of the tennis balls but also in the dynamic nature of live tennis matches.

The primary objective of this research is to leverage current state-of-the-art deep neural network algorithms to develop robust solutions for tennis ball detection within image frames. Successful implementation holds immense potential for advancing sports technology, providing tools that can contribute to a deeper understanding of real-time tennis court line calling systems.

The training and evaluation of the proposed solutions rely on carefully curated datasets, composed of both training and test data. The collection process is extensive, involving gathering images of tennis balls and splitting tennis videos into collections of frames. To assess the performance of the developed models, the research adopts a binary classification approach, categorizing predictions into "yes" or "no" evaluations. Evaluation metrics such as precision and mean average precision (mAP) provide a comprehensive analysis of the models' effectiveness in capturing tennis ball instances.

This paper presents two distinct solutions for tennis ball detection: Solution 1 uses the Mask R-CNN architecture, while Solution 2 uses the YOLOv8 architecture. Each solution undergoes a meticulous development process, including data manipulation to enhance the model's training. The results are subsequently analyzed using the metrics listed prior.

Overall, this research draws inspiration from existing studies, incorporating methods and strategies from recent works on tennis ball recognition using deep learning algorithms. By comparing the performance of Mask R-CNN and YOLOv8 CNN, this study aims to contribute valuable findings to the evolving landscape of object detection within the realm of professional sports.

Personally, I am interested in researching this topic due to my attachment to the sport. I have been an active tennis player for over 13 years as of this report's creation, and practice it actively. I also keep up with the professional scene, which inspired me to ask questions about the design of professional tennis' automatic systems, including line calling. There are many cameras placed around the court to capture all the angles of every tennis shot, including finely tracking its exact location when landing. Being able to actively detect tennis balls is industrially applied, and performing this research can provide an insight to the methodology employed for this task.

II. Related Work

A. *Color and Contour Classification of Tennis Balls*

The tennis ball detection landscape encompasses a variety of methodologies with one notable approach relying on color-based recognition and contour analysis. [1] Employing a recognition algorithm grounded in the Hue, Saturation, and

Value (HSV) color space, a robotic system can be applied to segment pixels exhibiting a distinct color characteristic. The subsequent application of a Hough circle transformation [2] accentuates the round contour of the objects within the image. This dual strategy involves both color and contour features to detect tennis balls during optimal camera conditions, which is relevant to that presented during live feed by the ATP. However, this approach presents challenges with scenarios involving background colors resembling tennis balls or potential misclassifications of irrelevant items in the background.

B. TrackNet

TrackNet [3] leverages an extensive dataset sourced from the broadcast video of the tennis men's singles final at the 2017 Summer Universiade, focusing on the development and evaluation of the TrackNet framework for tennis ball detection and tracking. While this study primarily centers on tennis, its methodology draws inspiration from Archana's algorithm [4], which is an image processing technique. Archana's approach serves as a benchmark for comparison, emphasizing the shift towards deep learning networks for improved object detection. The dataset itself comprises 20,844 frames with labeled attributes. Additionally, evaluation is extended to the challenging domain of badminton, demonstrating the network's adaptability to other domains. This research project sheds light onto high velocity object tracking, which is explored throughout this research paper.

C. Region Proposal Network

Convolutional neural networks (CNNs) form the foundational architecture for many computer vision tasks, including object detection. These neural networks are specifically designed to process and analyze visual data by leveraging convolutional layers. Convolutional operations involve the application of filters or kernels to input images, extracting features hierarchically. These features are learned through the network's training process, enabling the model to recognize patterns such as edges, textures, and more complex visual elements. The use of pooling layers helps reduce spatial dimensions, focusing on essential information. CNNs have proven highly effective in image-related tasks, providing the groundwork for subsequent advancements in object detection methodologies like Faster R-CNN.

The Faster R-CNN [5] architecture introduced Region Proposal Networks (RPN) to enhance the speed of detection frame generation, a critical advantage over traditional methods. RPN consists of two key components: anchor generation and bounding box regression. Anchors, representing candidate boxes at each sliding window position, are classified and refined through Softmax and bounding box regression, respectively. The proposal layer synthesizes positive anchors and their corresponding regression offsets, producing recommendations while eliminating unsuitable proposals. The RPN structure significantly improves the efficiency of generating detection frames compared to methods like selective search employed by traditional CNN models.

D. Mask R-CNN

Mask R-CNN [6] is an extension of the Faster R-CNN framework, designed specifically for instance segmentation tasks. It maintains the two-stage process of Faster R-CNN, featuring a Region Proposal Network (RPN) for candidate object boxes and a second stage for classification and bounding-box regression. What sets Mask R-CNN apart is the introduction of a third branch dedicated to generating binary masks for each Region of Interest (RoI), providing detailed spatial information for object instances.

During training, Mask R-CNN employs a multi-task loss function for each RoI, combining classification loss, bounding-box loss, and mask loss. The mask branch produces pixel-wise predictions, ensuring pixel-to-pixel correspondence and employing the RoIAlign layer to accurately align features within RoIs. This layer uses bilinear interpolation to address quantization issues and preserve spatial layout fidelity.

The flexibility of Mask R-CNN is highlighted through its compatibility with various backbone architectures, such as ResNet, ResNeXt, and Feature Pyramid Network (FPN). The framework achieves state-of-the-art performance in instance segmentation, outperforming other models on datasets like COCO. Mask R-CNN will be employed as the first solution for the tennis ball detection task, utilizing a dataset registered through COCO.

E. YOLOv8

The YOLO (You Only Look Once) series [7], starting with YOLOv1 in 2015, has seen continuous evolution, with YOLOv8 emerging as the latest advancement. YOLOv1 introduced a groundbreaking one-pass regression approach for object detection, while subsequent versions like YOLOv5 refined the architecture by incorporating anchor boxes. YOLOv8 builds upon this legacy, featuring a novel anchor-free detection mechanism and leveraging a diverse training dataset for improved performance on a wide range of images.

Selected for its assumed state-of-the-art status, YOLOv8 demonstrates superior metrics, including higher mean average precision (mAP) and refined post-processing techniques like Soft-NMS. The model's training process involves meticulous steps, such as transfer learning with pre-trained COCO weights, model size optimization, and hyperparameter tuning. Despite a slightly reduced detection speed compared to YOLOv5, YOLOv8 maintains real-time processing capabilities on modern GPUs.

Comprehensive evaluations, including challenging scenarios like detecting small objects and handling camouflage, highlight the model's adaptability. The refined model, achieved through transfer learning on a real-world dataset, showcases YOLOv8's robustness and effectiveness in various practical applications. In this project, YOLOv8 will be employed as the second solution to live tennis ball detection, relying on its robustness and ease of deployment to facilitate real-time object detection.

F. Roboflow

Roboflow [8], a comprehensive platform for computer vision, facilitates the entire lifecycle of building and deploying models, particularly specializing in object detection and labeling. Used by over 250,000 engineers, it offers a range of tools for creating datasets, training models, and deploying them into production environments. With a focus on streamlining workflows, Roboflow enables users to manage visual data, annotate images and videos, use foundation models, and deploy models both in the cloud and at the edge.

The platform supports integration into various pipelines with open APIs, SDKs, and developer tools, making it adaptable to different applications. Users praise Roboflow for its user-friendly interface and powerful features, emphasizing its impact on improving labeling experiences and streamlining the computer vision model development process. The usage of Roboflow for this project can automate the tennis ball labeling pipeline, and significantly decrease the time and resources necessary to create a properly labeled dataset.

III. Method Explanation

This project explores an introductory level understanding of fast-velocity small-area object tracking by applying cutting-edge neural network models to live tennis videos presented by the ATP. The objective of this paper is to find a solution that can effectively predict the location of a tennis ball in contrast to its background as its speed is rapidly changing through the dynamic environment of a professional match. By successfully tracking the location of a tennis ball, it can be possible to gain an understanding as to how industrial deep learning applications work, such as automatic line calling systems.

This study employs an experimental research design to investigate two independent deep neural network architectures towards live object detection. It employs a computer-vision based approach to detect and localize tennis balls in images and video frames. This decision was motivated by the need for a robust solution to be applied to diverse environmental conditions and player actions. Object detection allows for the simultaneous localization and classification of tennis balls, providing rich information for subsequent analysis.

A. *Dataset Creation*

The first step in this experimental procedure is to obtain access to sufficient training data for the network architectures. To properly apply tennis ball detection across a diverge range of environments, it is important to have access to a diverse array of tennis ball images across different environments. For the purpose of simply detecting a tennis ball through the standard position of a camera through live broadcasting of matches by the ATP, it is important to have access to many different frames of training videos extracted into frames with tennis ball locations labeled. However, since this research is applied to subsequent analysis, it is also important to recognize the tennis ball pattern outside of a zoomed-out top-down perspective. Therefore, the creation of the dataset consists of over 2000 images composed of both image frames from replayed matches, as well as generic stock images of tennis balls. This process will be explained further in the Data Acquisition section.

B. *Models*

To address the object detection challenge in this research, two prominent deep learning frameworks have been employed. These frameworks were chosen due to their proven effectiveness in handling object detection tasks and their distinct architectural characteristics.

The first solution utilizes Mask R-CNN, which is renowned for its precise instance segmentation capabilities. This model was employed to provide pixel-wise localization of tennis balls. The model extends the Faster R-CNN architecture by incorporating an additional branch that predicts segmentation masks for each detected object. The utilization of Mask R-CNN aims to enhance the granularity of our object detection results.

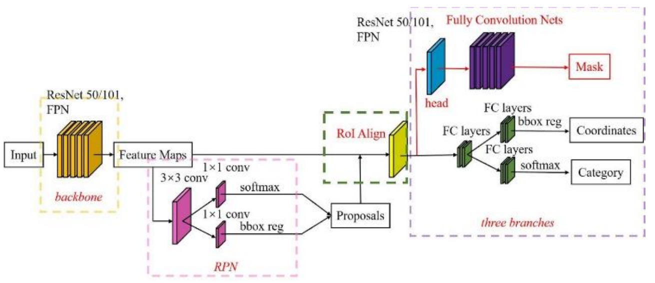


Figure 1: The Mask R-CNN model architecture for instance segmentation. Referenced via [9].

Additionally, the second solution utilizes YOLOv8, which is a state-of-the-art real-time object detection framework known for its speed and accuracy. YOLOv8 employs a single neural network to simultaneously predict bounding boxes and class probabilities for multiple objects within an image. The architecture divides the image into a grid and predicts bounding boxes with associated confidence scores for each grid cell. This real-time processing capability makes YOLOv8 particularly well-suited for applications demanding low-latency responses, such as tracking fast-moving objects in live tennis game scenarios.

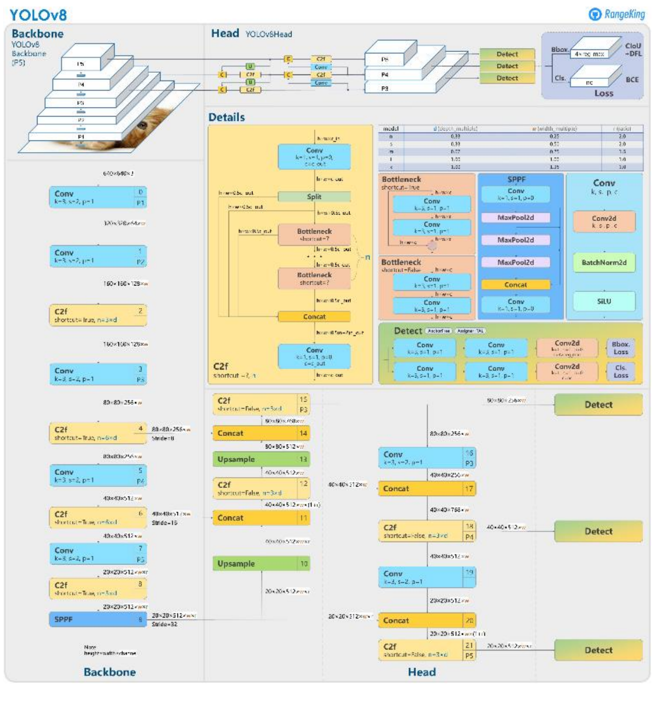


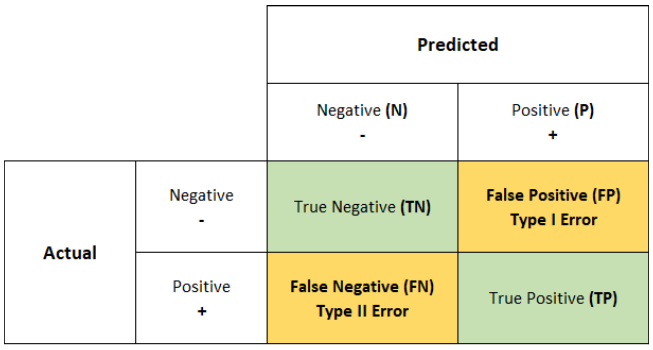
Figure 2: A brief summary of the YOLOv8 model architecture. Referenced via [10]

The implementation procedure of both of these models is as follows. First, both frameworks were initialized with pre-trained weights on general object detection datasets to leverage prior knowledge. Specifically, they were both initialized as checkpoints for a tuning process. The second step involves tuning these models through brute-force methods to optimize their performance for tennis ball detection. The results of this tuning process will be explained through the Experiment Settings section.

Lastly, both models were deployed and ran through two independent sets: a validation set a test set. The validation set is composed of labeled images from both generic tennis balls and extracted frames from live matches. This way, the mAP can be algorithmically computed and compared. Moreover, the test set is consistent of purely extracted frames from tennis videos, and the precision was manually determined by individually observing each output frame and constructing a confusion matrix.

C. *Evaluation Metrics*

There are three evaluation metrics being used to compare the performance of these models. Each individual metric can provide a valuable insight to a specific detail presented by the results. However, each of these metrics will be measured using a confusion matrix, which is shown below.



		Predicted	
		Negative **(N)** -	Positive **(P)** +
Actual	Negative -	True Negative **(TN)**	**False Positive (FP) Type I Error**
	Positive +	**False Negative (FN) Type II Error**	True Positive **(TP)**

Figure 3: A confusion matrix stores the truth values through a classification algorithm. In this case, binary classification is employed, and each square represents either a correct or incorrect classification depending on its label. Referenced via [11].

Figure 3 represents a binary confusion matrix, which presents true negatives (TN), true positives (TP), Type I errors, and Type II errors. For the sake of explanation, assume that a binary classifier can label an object as "Positive" or "Negative."

- **TN:** The classifier labels an object as "Negative" that is truly "Negative."
- **TP:** The classifier labels an object as "Positive" that is truly "Positive."
- **Type I Error:** The classifier labels an object as "Negative" when it is really "Positive."
- **Type II Error:** The classifier labels an object as "Positive" when it is really "Negative."

There are two important equations that can be derived from a confusion matrix, which are shown below.

$$Precision = \frac{TP}{TP + Type\ I\ Error}$$

Equation (1): Precision Score

$$Recall = \frac{TP}{TP + Type\ II\ Error}$$

Equation (2): Recall Score

Precision is defined by Equation (1) and counts the total number of "Positive" labels that were made correctly against the total number of "Positive" labels assigned overall. Precision is a good metric to use when the cost of a Type I error is high. In this context, a Type I error is when an object is labeled as a tennis ball, but is not actually a tennis ball.

Recall is defined by Equation (2) and counts the total number of "Positive" labels that were made correctly against the total number of "Positive" values that exist in the dataset. Recall is a good metric to use when the cost of a Type II error is high. In this context, a Type II error is when the object is not classified as a tennis ball, but actually is one.

In the context of this research, precision is a more powerful metric to use. Making sure that a tennis ball is found always is important for integration into industrial applications such as automatic line calling, so making sure that an object is detected serves as the primary objective. Therefore, precision will be evaluated in different metrics for analysis.

The first evaluation metric is mean average precision, or mAP. The mAP, or sometimes just AP, and can be computed algorithmically, which saves time on the computation step. Specifically, this study will focus on mAP@IoU=50%.

mAP is the average of the precision values across different recall levels. Since this project only has one class, the mAP can be substituted with AP, which determines the average precision among all classifications. Intersection over union (IoU) is a threshold specifier for considering a detection as correct. If the IoU between the predicted bounding box and the ground truth is greater than or equal to 50%, then the detection is considered correct.

mAP@IoU=50% provides an aggregated measure of how well the object detection model performs across different levels of recall, with a specific focus on bounding boxes that have at least a 50% overlap with the ground truth. Therefore, for mAP@IoU=50% to be completed, a validation set will need to be used. This metric will be computed using 30% of the training data, and comparing the predicted bounding box to the actual label.

The second evaluation metric is total precision across all values in the test dataset. The test dataset will be composed of image frame extractions from live tennis videos, and sequentially fed into the network for analysis. By creating a confusion matrix for each network, it is possible to determine the precision score by referencing Equation (1). However, to create the confusion matrix, this will require manual work by labeling each "Positive" and "Negative" prediction as correct or incorrect.

The third evaluation metric is total correct classifications across the test dataset. Through the previous step, a confusion matrix will be created. Therefore, the total number of correct classifications can be computed by the following equation.

$$Correct\ Classifications = \frac{TP + TN}{Total\ Number\ of\ Images}$$

Equation (3): Determine the correct number of classifications using a testing dataset's confusion matrix.

Thus, by calculating each of these scores, it can be comprehensively determined where each model is strong, and where any potential shortcomings may exist.

IV. Data Acquisition

The primary objective of this project is to apply object detection to live professional tennis matches. In accordance with the ATP official rulebook – updated in 2021, Chapter 10, Exhibits 07 [12] – there is a tournament requirement to position a broadcasting camera along the back of a tennis court parallel to its sidelines. That way, a standardized recording format can be used for online viewership of these events.

Therefore, the initial strategy towards data acquisition involved extracting frames from a diverse number of professional matches in various countries, spread across the three court surfaces: hard (concrete), clay, and grass. Approximately 1000 images were collected this way, but upon inspection, at a high resolution, tennis balls still only compose several pixels across the image at every given time.

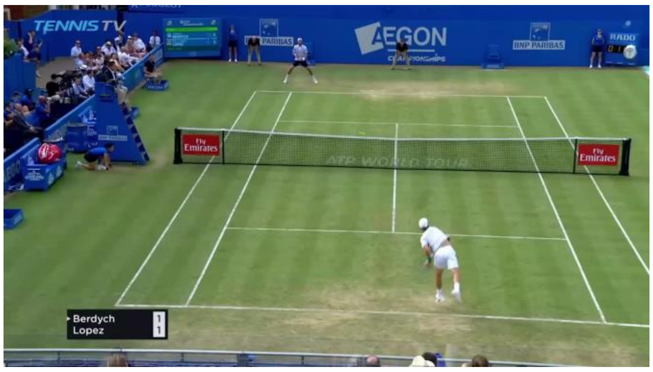


Figure 4: An example of an HD (1920x1080 pixels) frame taken from a professional tennis match. The ball represents a very small number of pixels taken above the net towards the center of the screen. Referenced via [13].

The training dataset employed in this project comprises two distinct clusters of images. The first cluster, exemplified by Figure 4, consists of 959 images extracted from video replays of professional ATP matches. This subset is further categorized into different court surfaces, being collected among hard courts, clay courts, and grass courts. The second set of training images consists of 1285 generic stock tennis ball images obtained from Adobe Stock [14]. In total, the training dataset encompasses 2244 images, with 30% of these images earmarked for validation purposes.

The next step in this process is image annotation, which is a crucial aspect of computer vision. It involves the meticulous process of labeling the boundaries of tennis balls within the training images, which provides the models necessary context to recognize and comprehend visual elements. In this context, effective image annotation is necessary for enhancing the accuracy and reliability of object detection algorithms.

As mentioned prior, there are over 2000 tennis ball images in the training dataset. Hence, automating the process became imperative, as manually labeling every image surpasses the limit of human energy. Therefore, several different strategies were used to streamline this process.

The first model of this annotation procedure used color as a primary discriminant for tennis ball identification. Introducing a color classification method rooted in the HSV color space, the algorithm was designed to delineate pixels corresponding to the color characteristics of tennis balls. Regrettably, this approach exhibited limitations, notably in misidentifying extraneous elements, as illustrated in Figure 5. Despite iterative adjustments to fine-tune the algorithm, it failed to yield results of sufficient proximity for practical deployment as an image labeling system, given the utilization of mAP@IoU=50% as the benchmark evaluation metric.

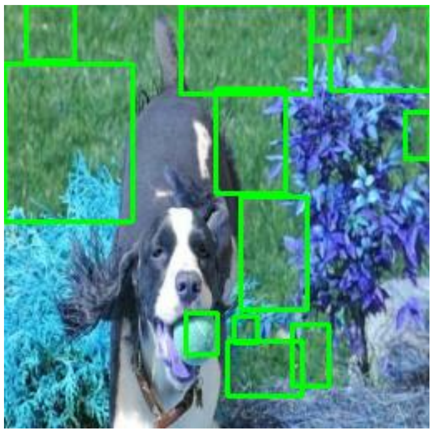


Figure 5: An example of using color to localize tennis balls.

The second model used for this annotation procedure gravitated towards the Hough Circle Transformation. This is a technique used for detecting circular objects within an image, which can be transferred to detecting tennis balls. Operating on the principle of transforming the image space to a parameter space, the Hough Circle Transformation excels in recognizing circular patterns through the identification of relevant parameters, including the circle's center coordinates and radius. However, as shown by Figure 6, this center coordinate detection presents errors when applied to diverse environments, and other regions of interest end up being given higher priority than the tennis ball during processing. After fine-tuning, the model could not successfully automate the tennis ball labeling process either.

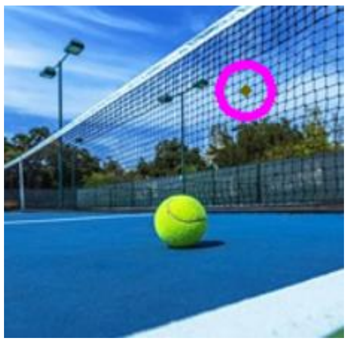


Figure 6: An example of using the Hough Circle Transformation to localize tennis balls.

The third attempt at solving this problem was deemed successful through the usage of Roboflow. Roboflow is an online neural network resource that provides tools for enhancing the image annotation process, involving the use of smart labeling tools trained on their connected networks. It empowers developers to easily create computer vision applications, without a dependency on code. It can be used for dataset annotation, preprocessing, transferring, exporting, and model training.

Firstly, several hundred stock images of tennis balls were labeled using the smart polygon tool. [15] This is an annotation assistant available through the Roboflow annotate program that uses a machine learning model to suggest a shape for the object being presented. It requires the user to mouse click the center of an object to be detected but can speed up annotation by removing the process of drawing boundaries around each object. Additionally, it allows for better fits by applying extra clicks inside and outside the suggested region to fine-tune the suggestion.

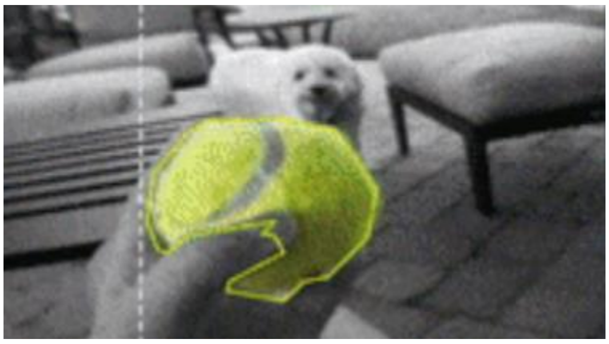


Figure 7: An example of using the smart polygon feature from Roboflow to localize tennis balls.

Second, Roboflow supports online model training, providing a valuable avenue for deployment in forthcoming projects, or unlabeled existing datasets – which can be applied to remainder of the tennis ball data. Leveraging this functionality, a fast object detection model was trained, yielding an impressive mean Average Precision (mAP) score of 94.0%, thereby streamlining the annotation process for the remaining tennis ball dataset. This transformative capability significantly reduced the annotation timeline from what would have been nearly impossible to a matter of several hours.

Nevertheless, a challenge persisted, stemming from the very small size of tennis balls within frames extracted from professional tennis matches, rendering them elusive for detection through the classifying model. Consequently, the solution involved the manual classification of all 959 images, accomplished with precision and efficiency utilizing Roboflow's advanced bounding box drawing tool.

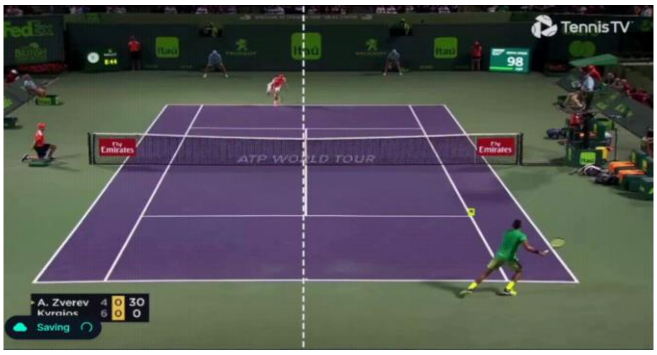


Figure 8: A bounding box drawn around a tennis ball during a professional tennis point using Roboflow's bounding box annotation feature.

Through these methods, the complete labeling of all 2244 images within this dataset was successfully achieved. Next, the dataset was split using a 70:15:15 ratio for the training, validation, and test sets, respectively. 70% of the images were split for training the model, 15% for the validation set to fine-tune hyperparameters, and the remaining 15% for evaluation purposes in the testing set, employing the mAP@IoU=50% metric. It is noteworthy that Roboflow Universe [16] allows for online dataset reference via API calls in any live Jupyter Notebook file, a capability that proves particularly valuable when using Google Colab. [17] Further details on this training process will be expounded upon in the Experiment Settings section.

Moreover, the other evaluation metric resides on manual observation of tennis ball classification through video data feed. Therefore, this testing dataset also needs to be created to allow evaluation of precision and correct classification percentage. Therefore, two entire points on hard courts, clay courts, and grass courts were extracted into 1729 images, which composes the video training set. After each model is deployed, they will be fed in each image frame as a sequence, and return the corresponding sequence with either a label drawn around one specific object, or no label drawn signifying the absence of a detection.

V. PSEUDOCODE

For the scope of this project, there are two main segments that can be explained through pseudocode: Mask R-CNN model implementation and YOLOv8 model implementation. Both frameworks rely on using a model checkpoint and fine tuning hyperparameters to transfer the classification task to tennis balls. By viewing each framework independently, an insight on their differences and strengths can be outlined. It is important to note that both models have been implemented through the use of Google Colab and its integrated T4 GPU.

A. *Mask R-CNN*

It is important to note that a detailed walkthrough of loading a Mask R-CNN checkpoint is provided by the official Google Colab tutorial of Detectron2. [18] By referencing this process, it is simple to transfer responsibility of the object detection to a custom dataset, as well as introduce sanity checks to ensure functionality is correct. Moreover, the first step in this implementation is to install detectron2, which can be done by the line of code below.

```
!python -m pip install
'git+https://github.com/facebookresearch/detectron2.git'
```

Moreover, this implementation will be using PyTorch, [19] so it is important to import the required libraries, as shown below.

```
import torch, detectron2
```

The first step to train any network is to accumulate training data. This data has been labeled previously using Roboflow, and can be accessed by using an API call. Specifically, we can use the segment of code below to access the specific project. However, the API key is censored for security reasons.

```
rf = Roboflow(api_key="______")
project = rf.workspace("joshuaglaspey-n5m2m").project
("tennis-ball-segmentation")
dataset = project.version(1).download
("coco-segmentation")
```

From here forward, the implementation will be described using pseudocode to illustrate the process. Once the dataset is locally downloaded (or saved to RAM if using Google Colab), it can be registered to the COCO database. This is necessary to call the datasets during training. As mentioned previously, the dataset has already been partitioned into three separate categories using a 70:15:15 split for training, validation, and testing.

```
# Create names to represent file data
TRAINING_DATA_NAME = "name to be registered to
the COCO database"
TRAINING_DATA_IMAGES_PATH = "local directory to
the training data"
TRAINING_DATA_ANNOTATIONS_PATH = "local directory
to the training data annotation json file"

# Register the data to COCO dataset
register body to coco instances (
    name = TRAINING_DATA_NAME,
    json file = TRAINING_DATA_ANNOTATIONS_PATH
    image root path = TRAINING_DATA_IMAGES_PATH
)

# Repeat for Validation and Test sets
```

Now that the data has been imported into the project, it is time to import the checkpoint of the Mask R-CNN framework. This can be done through referencing the following configuration:

- COCO-InstanceSegmentation/mask_rcnn_R_101_FPN_3x

COCO is included with the installation of detectron2, and includes loadable checkpoints for many different models, which includes Mask R-CNN as provided by the line above. Additionally, the following hyperparameters are configurable for this model.

```
# Hyperparameters
max_iterations
evaluation_period
base_learning_rate
number_of_classes
number_of_workers
batch_size

# Transfer checkpoint model
model weights = "Mask R-CNN Configuration"
model training data = TRAINING_DATA_NAME
model validation data = VALIDATION_DATA_NAME
model testing data = TESTING_DATA_NAME

# Transfer the hyperparameters
to the checkpoint model
```

At this point, it is now a matter of training the model. This can be done by creating a Training object, and calling it to train the detectron2 configuration, which is visualized below.

```
# Train the model
trainer = Trainer(detectron2 configuration)
trainer = Begin new train
trainer.train()
```

Now, after waiting for the training process to complete, the Mask R-CNN model is completed. The mAP@IoU=50% can be evaluated using the method below.

```
# Determine mAP through test data
evaluator = new COCOEvaluator using TEST_DATA_NAME
test_loader = loader using TEST_DATA_NAME
print(stastics:
        Model: detectron2 configuration
        Loader: test_loader
        Evaluator: evaluator
    )

# The mAP will be displayed using this
```

Lastly, to create precision and total correct percentage metrics, the video testing set needs to be annotated using the model's predictions. This can be implemented using the method below.

```
# Register video testing data to COCO

# Run predictions on the data
for each image:
    outputs = detectron2.predict(image)

    # Filter detection based on threshold, and only
       display the highest
    detections_above_threshold =
       outputs[all detections] > threshold_amount
    if detections_above_threshold.any():
        highest_confidence_index = max
               (detections_above_threshold)

    # Apply annotation
    visualizer = new Visualizer object
    output = visualizer.draw_prediction(
       outputs[all detections][highest_confidence_index])
    store annotated image to new file path
    save the image

# Download all images
```

After implementing this code, uploading and registering the data sets, and executing the program, the 1729 video frames can be manually observed to construct a confusion matrix for performance interpretation.

B. *YOLOv8*

The YOLOv8 implementation through Google colab follows a similar structure to the Mask R-CNN implementation. It is more concise overall, but references data from Roboflow all the same. Therefore, the first step is to import the necessary libraries and data.

```
!pip install ultralytics==8.0.196
!pip install roboflow
from roboflow import Roboflow
rf = Roboflow(api_key="_________")
project = rf.workspace("joshuaglaspey-n5m2m")
.project("tennis-ball-detection-vmjer")
dataset = project.version(5).download("yolov8")
```

Once the data is imported, Ultralytics, the creator of the YOLO framework, provides a simple method call for tuning a network. For this project, the call can be shown by the logic below.

```
# Tune hyperparameters
load YOLO "nano" model checkpoint
model.tune(data = data.yaml,
    Input hard-coded parameters here)
```

Once the model.tune() function is called, the best hyperparameters are stored to a file with the name: "best_hyperparameters.yaml." Therefore, by combining these hyperparameters with the data file that contains training, validation, and testing image directories, it is possible to automate the tuning process to feed hyperparameters into the model to be trained.

This can be shown by the logic below.

```
# Function to take the paramaters of a tuned model
# and import them to the .yaml file of the dataset
def move_yaml_contents():

  # Path to the first YAML file
  file1_path = "best_hyperparameters.yaml"

  # Path to the second YAML file
  file2_path = "data.yaml"

  Read the contents of the first YAML file

  Save it to a buffer

  Append the contents of the first YAML file
  to the second YAML file using the buffer
```

Moreover, once the data file has been setup, it is now possible to train the YOLOv8 network. Once again, this is simplified using Ultralytics' YOLO model function calls, where the logic is shown below.

```
load YOLO "nano" model checkpoint
move_yaml_contents()

# Train the YOLO model
!yolo
    task=detect
    mode=train
    model=yolov8n.pt
    data=data.yaml
```

From this point, the model is trained. Now it is a measure of determining the three metrics.

The first metric, mAP@IoU=50% can be determined using a validation function provided by Ultralytics. This is shown below.

```
!yolo
    task=detect
    mode=val
    model=best weights model
    data=data.yaml
```

Lastly, to determine the precision and correct classification percentage, the code imports an additional unlabeled dataset and runs a prediction task on it. Once again, this is simplified to one line of code.

```
!yolo
    task=detect
    mode=predict
    model=best weights model
    confidence threshold=some value
    source=unlabeled images
```

Thus, the final steps are to download and observe all of the images to construct a confusion matrix.

VI. Experiment Settings

The experiment involves having a fine-tuned configuration to produce optimal results in the realm of tennis ball detection.

A. Dataset

The training dataset is consistent of two groups of images. The first group is comprised of generic tennis ball stock images across diverse environments. This way, the tennis ball's color and pattern can be learned against different colors and shapes in the background. Additionally, it is also consistent of frames taken from professional points. These images, as shown by Figure 4, are zoomed out, and each tennis ball makes up several pixels across an HD image. All of the stock images are standardized to the same 256x256 resolution, and all of the video frames are default exported at 1920x1080 pixels. There is a total of 2244 images, 70% of which have been designated for training. Therefore, 1570 images are within the training dataset.

The validation and testing datasets are consistent of the same types of data as the training dataset – tennis ball stock images and video frames from professional points. Each of these sets contains 15% of the dataset pool, so both of these sets contain 337 images.

The testing dataset will be used to determine the mAP for each model. Both implementations contain an algorithmic method for determining model performance, which includes the mAP@IoU=50% metric.

An additional testing dataset, known as the video testing dataset, will be consistent of 1729 extracted frames from six different tennis points. Two of these points will be on hard courts, two on clay courts, and two on grass courts. This data will be fed into each deployed network, and the number of correct and incorrect classifications will be manually counted and constructed into a confusion matrix.

B. *Machine Configuration*

Both of these networks are deployed through Google Colab. Therefore, the machine performance is dependent on the resources provided by this environment. For this implementation, it is necessary to use a CUDA-enabled GPU, which can be accessed via Google Colab's Tesla T4 GPU. [20] This GPU features:

- 2560 CUDA cores
- 320 Tensor cores
- 16GB of GDDR6 memory
- 320GB memory bandwidth

This information makes the Tesla T4 GPU a strong resource to be used for training these networks.

C. *Parameters*

Each individual framework features specific hyperparameters that can be used to boost the performance of their deployed networks. A list for each of these is shown below.

a) Mask R-CNN

- Maximum number of iterations: 2000
- Evaluation period: 200
- Base learning rate: 0.001
- Number of classes: 1
- Weights: Load from "mask_rcnn_R_101_FPN_3x" checkpoint
- Batch size: 64
- Number of workers: 2
- Instances per batch: 2
- Mask format: "bitmask"

b) YOLOv8

- Epochs: 50
- Learning rate: 0.00801
- Learning rate factor: 0.00787
- Momentum: 0.88389
- Weight decay: 0.00058
- Warmup epochs: 4.38397
- Warmup momentum: 0.95
- Box localization: 6.84435
- Classification loss coefficient: 0.27659
- Dynamic feature learning coefficient: 1.78844
- HSV Hue: 0.01879
- HSV Saturation: 0.6589
- HSV Value: 0.4689
- Degrees: 0
- Translation: 0.08389
- Scale: 0.47946
- Shear: 0
- Perspective: 0
- Flip up-down: 0
- Flip left-right: 0.4642
- Mosaic: 1.0
- Mixup: 0
- Copy/paste: 0

VII. Experimental Results

Initially, the experimental procedure followed this original structure:

1. Research state-of-the-art deep neural network models that excel in object detection.
2. Use a collection of back-perspective tennis videos along with stock images of tennis balls to train the networks with diverse inputs.
3. Perform metric calculations on the models algorithmically. Specifically, determine the mAP@IoU=50%.
4. Input a frame sequence from live tennis points and manually determine the precision of each model.

A. *Mean Average Precision*

For the experimental results, the first metric to be determined is the mAP@IoU=50% score. Using the algorithms provided by both detecton2 and YOLO, it is possible to algorithmically determine this metric using the same testing dataset. This way, since both models have been trained, validated, and tested using the same images, bias can be minimized.

The results of the mAP@IoU=50% analysis are shown by Table 1. By direct comparison, Mask R-CNN's model outperformed YOLOv8 in terms of average precision using an IoU threshold of 50% by 45%. In other words, on average, the Mask R-CNN model could predict a tennis ball's region with at least 50% confidence in 45% more circumstances than the YOLOv8 model.

Mean Average Precision of the Two Frameworks

Framework	**mAP@IoU=50%**
Mask R-CNN	0.787
YOLOv8	0.541

Table 1: The algorithmic collection of mAP@IoU=50% for each framework. Detectron2 and YOLO provide built-in methods for computing these values given an annotated dataset.

However, this metric is used to determine overall model quality in all circumstances. For a combination of stock tennis ball images and back-perspective professional tennis point frames, these values are calculated. That means, in the general application, Mask R-CNN is the stronger selection.

B. *Difficulties with Original Precision Metric*

Moving forward, to apply these frameworks to live tennis matches, it is important to understand the precision score when taken from the perspective of a camera positioned on the perimeter of the tennis court.

Originally, the plan for determining precision was to reference six unlabeled tennis points – two from hard courts, two from clay courts, and two from grass courts – and allow each model to predict the tennis ball in 1729 extracted frames. Then, using manual labor, each image would be observed, and a confusion matrix would be constructed.

However, after applying this method to both trained models, the results were diminishing. Table 2 summarizes these results for one point analyzed by each model. Every frame was not analyzed due to the burden of limited manpower, and once the results began accumulating, finishing the procedure was deemed inconclusive.

Object Detection Applied from Back-Perspective Viewpoint

Framework	**Total Frames**	**Total Frames with a Detected Tennis Ball**
Mask R-CNN	263	94
YOLOv8	332	6

Table 2: The number of correctly classified frames given a distinct input of a different tennis point for each framework. The totals did not yield strong results, so this presented questions towards the setup of the procedure.

Mask R-CNN was able to classify the tennis balls at a higher percentage level, but overall, neither model produced results nearly up-to-par as to what they each sell as. After investigating, the most likely cause for this is the format in which the input data is acquired. It is a popular topic for high velocity low area object detection in the case of using deep neural networks, but the provided input data is in HD format (1920x1080 pixels). In each frame, the tennis ball contains a radius of two to five pixels overall. This essentially blurs it into the background and makes it no more apparent than many regions proposed by the background.

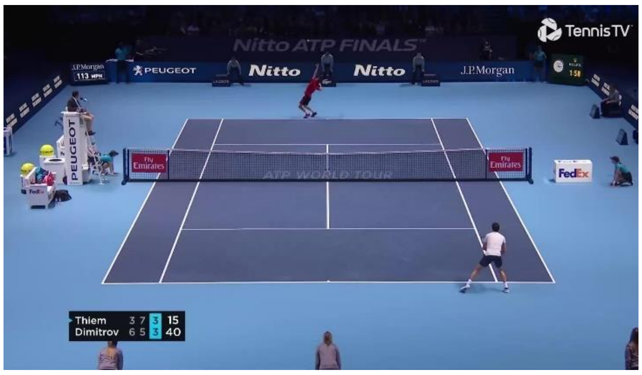


Figure 9: An HD (1920x1080 pixels) frame taken from a professional tennis match. The ball, positioned over the far tennis player, is nearly impossible to see from this resolution, making it difficult for a computer to perform essential training from this data.

In fact, there are a couple of reasons as to why, despite the tennis ball being as easy to watch from a broadcasting perspective, the tennis ball is impossible to locate from a still image as shown by Figure 9. Firstly, the human brain can detect motion tracking. Hyönä [21] references motion tracking using human eyes in terms of saccades, which are rapid, involuntary eye movements. The human visual system is adept at following moving objects through a series of saccadic eye movements, allowing for continuous tracking of the object's trajectory. This inherent capability of the human eyes to engage in motion tracking aids in real-time perception, which is a vast benefit to the limitations of the current setup for this project.

The limitations of still images in capturing the essence of motion tracking becomes evident through the use of a tennis ball. A single image, as shown by both frameworks, fails to convey the temporal aspect of the ball's movement and the dynamic interplay of forces involved. Motion tracking using human eyes involves not only the ability to smoothly follow the movement of an object, but also to predict its future location based on visual cues. Still images lack the temporal dimension required for accurate motion tracking, making it challenging for observers to precisely locate the tennis ball's position at a given moment.

Secondly, humans have the benefit of binocular vision, which contributes to depth perception. Boyd [22] explains binocular vision as the utilization of input from both eyes of a human. By nature, this is a limitation of the current project, as the input is the result of a single camera feed. Using two inputs enables depth perception through convergence, where each eye captures an object from slightly different angles. The brain integrates these perspectives – a process known as stereopsis – providing a three-dimensional understanding of the environment. This binocular advantage proves essential in accurately tracking moving objects, especially in the context of a small tennis ball.

Conversely, individuals with monocular vision may encounter challenges in depth perception, lacking the stereoscopic input of binocular vision. While it is possible for prolonged monocular vision individuals to adapt, their depth perception differs. This is the problem faced by using a one-camera system for observing a tennis ball: there is no direct implementation of depth perception. This makes spatially tracking the small object difficult as it is presented with such a small area.

Thirdly, humans rely on cognitive processing, which is the processes of the human brain filtering out irrelevant information to focus on the most salient aspects of a scene. Zhao [23] studies cognitive processing for object tracking with multiple features. When tracking a tennis ball, the brain engages in this to prioritize and selectively attend the critical features of a ball while disregarding distracting background details. The challenge arises for these frameworks because the tennis ball is only several pixels in size, and its visual representation may align with various features in the background.

This task becomes even more challenging when the visual environment is dynamic, with changing background and lighting conditions between facilities. The brain must continuously adapt its cognitive process to ensure that it accurately tracks the ball's movement, even when it aligns with various background features. Despite these challenges, the human visual system demonstrates a remarkable ability to filter out irrelevant information and focus on crucial elements. This is a limitation of this current project, as not having a cognitive process system yields setbacks on understanding changes in the environment and not attributing the small tennis ball region to other unrelated areas.

C. *Refined Precision Metric for Object Detection*

Thus, considering all of the factors of the original method, it is worth noting that, to stay true to the original objective of this project, it is unwise to train new networks. Rather, it is imperative to change the means at which the unlabeled input data is presented. Due to the nature of HD images, the scale of a tennis ball is too small to properly be analyzed by deep neural networks that have not been trained with prediction calculations or binocular vision. However, this project's scope is to provide accurate tennis ball tracking during professional match play from the perspective of the cameras along the peripheral of the court.

As mentioned prior, the ATP positions cameras along the entire perimeter of the tennis court to track the tennis ball's location at all times. This is used for automatic line calling, as well as other topics. Thus, rather than provide the video feed from a zoomed-out backwards perspective, it is possible to use zoomed-in replays of points to accurately track the tennis ball's location. After a tennis point is completed on a live broadcast, the final couple swings are replayed while zooming in on the tennis player and the ball. This drastically increases the size of the tennis ball within the input image, as shown by Figure 10.

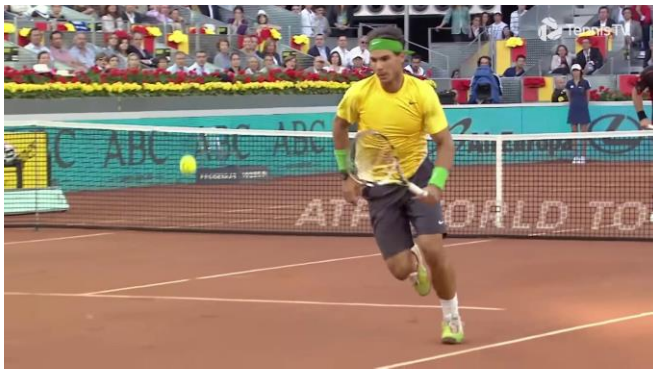


Figure 10: An HD (1920x1080 pixels) taken from a replay of a live professional tennis match. The tennis ball is large because the camera is zoomed in to replay a specific shot from the last point.

Therefore, by using these zoomed-in videos, it is possible to redeploy each of the two frameworks to a new collection of 1814 unlabeled frames. Therefore, the new project procedure is shown below:

1. Research state-of-the-art deep neural network models that excel in object detection.
2. Use a collection of back-perspective tennis videos along with stock images of tennis balls to train the networks with diverse inputs.
3. Perform metric calculations on the models algorithmically. Specifically, determine the mAP@IoU=50%.
4. Input 1814 extracted frames from replays of live professional tennis points, and manually construct feature matrices from the results.

D. *Precision*

Each network is trained using the collection of 1570 combined back-perspective images extracted from professional tennis points and generic stock images of tennis balls. Then, using these models, 1814 extracted frames from replays of live professional tennis points can be used as inputs for object prediction. An example of successful predictions can be shown by Figures 11 and 12.

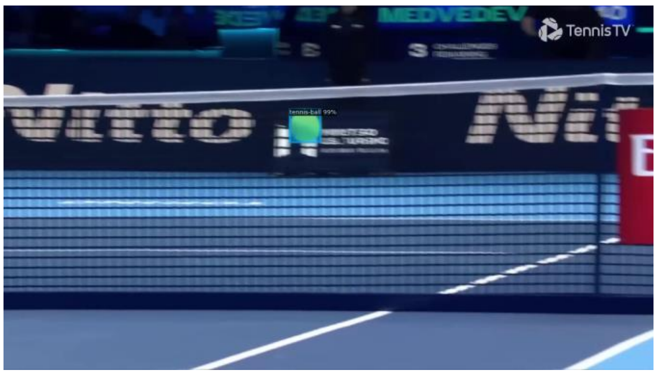


Figure 11: A successful detection of a tennis ball using Mask R-CNN.

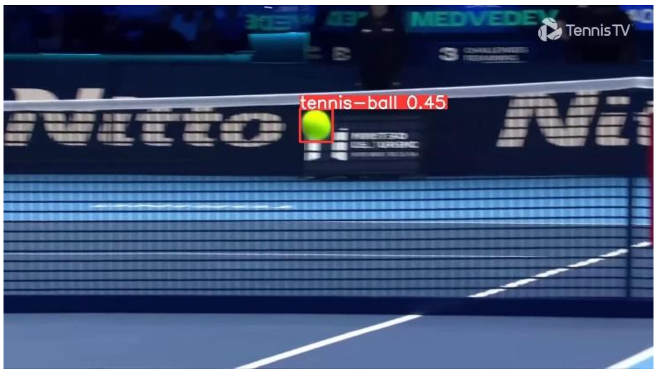
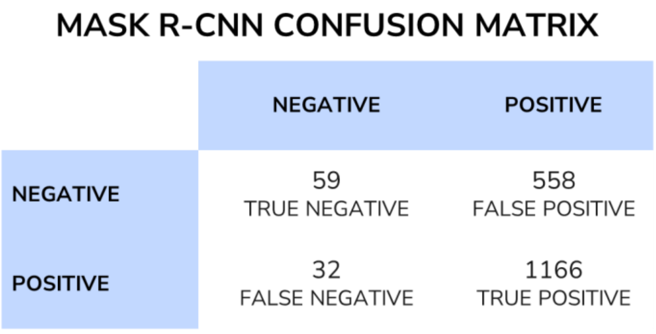


Figure 12: A successful detection of a tennis ball using YOLOv8.

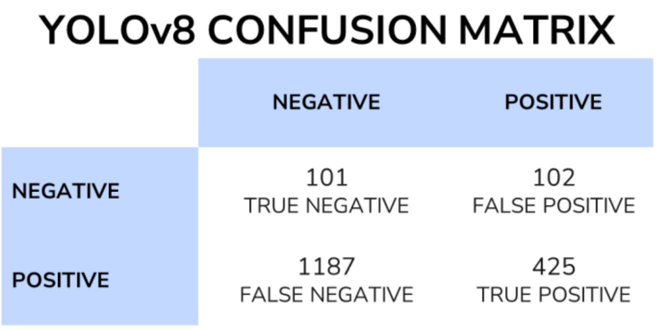
The confusion matrices for both frameworks are shown by Figures 13 and 14. To reiterate, these matrices were computed using manual labor, as the data is unlabeled and presents no algorithmic method to determine precision nor correct classifications.



MASK R-CNN CONFUSION MATRIX

	NEGATIVE	POSITIVE
NEGATIVE	59 TRUE NEGATIVE	558 FALSE POSITIVE
POSITIVE	32 FALSE NEGATIVE	1166 TRUE POSITIVE

Figure 13: The confusion matrix for the classification results determined by the Mask R-CNN model.



YOLOv8 CONFUSION MATRIX

	NEGATIVE	POSITIVE
NEGATIVE	101 TRUE NEGATIVE	102 FALSE POSITIVE
POSITIVE	1187 FALSE NEGATIVE	425 TRUE POSITIVE

Figure 14: The confusion matrix for the classification results determined by the YOLOv8 model.

For this study, it is imperative to understand the implications of a Type I and Type II error. The Type I error, signified by the false positive total, is when an object is labeled as a tennis ball, but is not actually a tennis ball. The Type II error, signified by the false negative total, error is when the object is not classified as a tennis ball, but is a tennis ball. It has been established that precision will be utilized because the objective of this project is to ensure that the tennis ball is tracked for the entire duration of the input video. Therefore, for precision, the false positive proportion to total positive classifications will be analyzed.

On a separate note, each of these two models exhibits distinct behaviors. For the Mask R-CNN model, there is a large number of false positives, signifying that frequently the framework selected a separate region of interest as what it believed to be a tennis ball. This occurred nearly one-third of the total positive classifications. Moreover, there was almost never an unclassified frame, as there were only 91 total negative classifications.

The YOLOv8 model exhibited nearly opposite behavior. There is a much smaller quantity of false positive classifications in comparison to the Mask R-CNN model, but as a proportion to overall positive classifications, it makes up about a fifth of the total. Furthermore, there were only 527 positive classifications, which is diminishing in comparison to the 1724 positive classifications of the Mask R-CNN model, and of the 1814 total frames used as input. Conversely, there were 1288 negative classified frames, where 1187 of those were made incorrectly. In other words, in 1187 frames extracted from live replays, there was no tennis ball detected when a tennis ball was present.

Moving on, using Equations 1 and 3, it is possible to determine the precision and correct classification percentage of these two models. These results are shown in Table 3.

Precision and Correct Classification Percentage of the Two Frameworks

Framework	**Precision**	**Correct Classification Percentage**
Mask R-CNN	0.676	67.49%
YOLOv8	0.806	28.98%

Table 3: The precision and correct classification percentages of both Mask R-CNN and YOLOv8. These metrics were calculated using the results of their respective confusion matrices.

As shown by the results, in terms of precision, YOLOv8 has stronger results. In approximately four out of five positively classified images, the YOLOv8 model will output the tennis ball as the highest confidence prediction. Regarding Mask R-CNN, approximately two out of three frames will have the tennis ball given the highest confidence prediction. In terms of ensuring correctness, YOLOv8 is the stronger model.

However, in terms of the number of frames correctly classified, Mask R-CNN is the heavy favorite. In approximately two-thirds of the images, this model either correctly classified a tennis ball as in the image, or correctly did not classify anything because a tennis ball is missing. However, the YOLOv8 produced poor results on this topic. Only approximately 29% of the images sent through the model were classified correctly. This can be attributed to the large false negative total present in Figure 14.

VIII. DISCUSSION

The initial analysis focused on algorithmically determining the mAP@IoU=50%. Using detectron2 and YOLOv8, this metric was computed, revealing that Mask R-CNN outperformed YOLOv8 by 45% of generic tennis ball detection cases. These results imply that, in a general application with diverse inputs, Mask R-CNN stands out as the stronger choice.

Separately, the attempt to assess precision from a back-perspective viewpoint using six unlabeled tennis points encountered challenges. Both Mask R-CNN and YOLOv8 yielded inconclusive results due limitations of the project setup and input data resolution. Therefore, to address these limitations, which are outlined in Section 7.B, the project adopted a refined method of accumulating precision. Instead of training new networks, the focus shifted to using zoomed-in replays of live professional tennis points to enhance the visibility of the tennis ball. This approach aimed to present a more realistic scenario for the models, as well as allow for feasible object detection given HD (1920x1080 pixels) images.

Using this refined process, the Mask R-CNN and YOLOv8 models predicted tennis balls across a new unlabeled data set of 1814 frames. Precision and correct classification percentages were computed from their confusion matrices.

While YOLOv8 showcased a higher precision metric, meaning that a larger proportion of its positive classifications were accurate, Mask R-CNN emerged as the frontrunner in terms of correctly classifying a tennis ball overall. The balance between precision and consistency is a critical consideration, as it determines the effectiveness of each of these models in different scenarios.

In terms of precision, YOLOv8 demonstrated a high precision of 0.806, indicating that when it predicted the presence of a tennis ball, it was correct 80.6% of the time. This is beneficial in the scope of correctness, as YOLOv8 was stronger in making sure that the highest confidence region of interest was the tennis ball and not an arbitrary section of the background.

In terms of correct classification percentage, Mask R-CNN excelled in classifying a larger number of frames correctly. In 67.5% of the frames, the model either correctly identified a tennis ball or correctly refrained from making a prediction when a tennis ball was absent. This metric is critical in ensuring tracking is consistent.

It would originally seem that YOLOv8 would be the preferred model for this problem, as it yields a high precision score. However, the very small overall correct classification makes the model unusable for industrial applications. The Mask R-CNN model also suffers from the fact that it tends to attribute a separate section of the background to being the tennis ball rather than the actual tennis ball. Thus, this model also suffers in accuracy, and could not be deployed industrially. However, this falls to the setup of the project, and future ways to better this process are explored in the Future Works section.

Overall, the Mask R-CNN model is preferred from these results. The goal of this project is to ensure a comprehensive and accurate tracking of the tennis ball, while also consistently having it detected. The Mask R-CNN does not fall far behind

YOLOv8's precision, and far excels in correctness. Its ability to classify a higher percentage of frames implies a more reliable performance in scenarios where the tennis ball might be challenging to detect due to its small size and dynamic backgrounds.

IX. Conclusion

In this research process, the application of deep neural network models, specifically Mask R-CNN and YOLOv8, was explored for the challenging task of object detection in live professional tennis matches. The project aimed to track tennis balls in real-time, following the ATP's guidelines for camera placement and considering the diverse conditions of different court surfaces.

The evaluation began with the acquisition of data. The project utilized HD images of the back-perspective viewpoint of live professional tennis points, as well as generic stock images of tennis balls as the training data collection. This allowed for generalization of the data, as well as knowledge for detecting the tennis ball in a live point scenario. The models were tested using unlabeled data from zoomed-in replays.

The first metric determined was mAP@IoU=50% for both models. The results indicated that Mask R-CNN outperformed YOLOv8 by 45%, suggesting its superiority in generic tennis ball detection across diverse scenarios.

However, when transitioning to a more practical scenario involving the back-perspective viewpoint commonly used in live broadcasts, both models faced challenges. The original precision metric, determined by manual observation of six unlabeled tennis points, provided inconclusive results. Limitations arose from the small size of tennis balls in HD images, lack of motion tracking, monocular vision constraints, and the absence of cognitive processing akin to human visual systems.

Recognizing these limitations, a refined approach was adopted, utilizing zoomed-in replays of live professional tennis points. This sought to enhance the visibility of the tennis ball and present a more realistic input scenario. The subsequent precision and correct classification analyses produced nuanced insights into the models' performance.

While YOLOv8 exhibited a higher precision, indicating more accurate prediction when a tennis ball was detected, Mask R-CNN excelled in overall correctness. Mask R-CNN demonstrated its ability to classify a larger number of frames correctly, ensuring a more reliable performance in scenarios where detecting the small-sized tennis ball amid dynamic backgrounds proved challenging.

Considering the project's objective of comprehensive and accurate tennis ball tracking during professional matches, Mask R-CNN emerged as the preferred model. Despite its tendency to attribute separate sections of the background as tennis balls, its higher correct classification percentage and balance between precision and consistency make it more suitable for industrial applications where reliable tracking is imperative.

In conclusion, this study contributes valuable insights into the complexities of deploying deep neural network models for real-time object detection in professional tennis matches. The findings underscore the importance of considering specificities in camera perspectives and image resolutions for effective model deployment in dynamic sports environments.

X. Future Works

This research lays the foundation for further exploration in refining object detection models for live tennis match scenarios. As discovered by the original precision metric calculations, many factors can be weighed into the unlabeled data acquisition process. Being restricted to one zoomed-out camera and only HD image frames made the object detection problem struggle with accuracy. There are ways to address this problem so that the original broadcasted angle can be used, and successful tracking may be implemented.

One promising avenue for future research is incorporating motion tracking capabilities into the object detection models. The human visual system's ability to engage in motion tracking provides an extra dimension to real-time perception. Integrating motion tracking algorithms into deep neural networks can enhance the models' understanding of the temporal aspect of the tennis ball's movement. Exploring and adapting existing motion tracking techniques for object detection in dynamic sports environments could significantly improve the model's predictive capabilities.

To ensure the continuity and realism of tennis ball tracking, future works could explore methods to incorporate motion tracking data for predicting the subsequent ball location. By leveraging the information from the motion tracking algorithms, the models can make predictions that are not unreasonably far from the previous frame's detection. This approach aims to emulate the human visual system's ability to anticipate an object's future location based on its trajectory. Implementing predictive elements into the models would contribute to more accurate and consistent tennis ball tracking.

Moreover, expanding the data acquisition setup to include more than one camera could address the depth perception challenge outlined in the limitations. Drawing inspiration from the human binocular vision system, utilizing multiple cameras placed around the tennis court, similar to the ATP's guidelines on camera placements, can provide distinct perspectives, allowing for convergence and stereopsis. This binocular advantage contributes to enhanced depth perception, potentially making it easier for the models to spatially track the small tennis ball accurately.

Additionally, the insights gained from this research can be extrapolated to other domains facing challenges in fast-velocity, small area detection. By transferring the knowledge and methodologies developed for tennis ball tracking to similar scenarios, such as other sports or robotics, the applicability and effectiveness of the models can be extended. This cross-disciplinary approach could open avenues for advancements in real-time object detection in diverse fields where precision and speed are critical.

References

[1] Wu, D.; Xiao, A. Deep Learning-Based Algorithm for Recognizing Tennis Balls. Appl. Sci. 2022, 12, 12116. https://doi.org/10.3390/app122312116

[2] "OpenCV: Hough Circle Transform," docs.opencv.org. https://docs.opencv.org/4.x/da/d53/tutorial_py_houghcircles.html

[3] Y.-C. Huang, I.-N. Liao, C.-H. Chen, T.-U. İk, and W.-C. Peng, "TrackNet: A Deep Learning Network for Tracking High-speed and Tiny Objects in Sports Applications," arXiv:1907.03698 [cs, stat], Jul. 2019, Available: https://arxiv.org/abs/1907.03698

[4] M, Archana & Geetha, M.. (2015). Object Detection and Tracking Based on Trajectory in Broadcast Tennis Video. Procedia Computer Science. 58. 225-232. 10.1016/j.procs.2015.08.060.

[5] W. Li, "Analysis of Object Detection Performance Based on Faster R-CNN," Journal of Physics: Conference Series, vol. 1827, no. 1, p. 012085, Mar. 2021, doi: https://doi.org/10.1088/1742-6596/1827/1/012085.

[6] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1–1, 2018, doi: https://doi.org/10.1109/tpami.2018.2844175.

[7] D. Reis, J. Kupec, J. Hong, and A. Daoudi, "Real-Time Flying Object Detection with YOLOv8," May 2023. Available: https://arxiv.org/pdf/2305.09972.pdf

[8] "Roboflow: Go from Raw Images to a Trained Computer Vision Model in Minutes.," roboflow.ai. https://roboflow.com/

[9] "What is Mask R-CNN? The Ultimate Guide.," Roboflow Blog, Aug. 09, 2023. https://blog.roboflow.com/mask-rcnn/

[10] "Brief summary of YOLOv8 model structure · Issue #189 · ultralytics/ultralytics," GitHub. https://github.com/ultralytics/ultralytics/issues/189

[11] A. Suresh, "What is a confusion matrix?," Medium, Nov. 20, 2020. https://medium.com/analytics-vidhya/what-is-a-confusion-matrix-d1c0f8feda5

[12] 2021 Rulebook - ATP Tour, https://www.atptour.com/-/media/files/rulebook/2021/2021-atp-rulebook-chapter-10_exhibits-07apr.pdf (accessed Dec. 3, 2023).

[13] "ATP Tennis Streaming Online - Watch Tennis Live," Tennis TV - ATP Tennis Streaming Online - Watch Tennis Live. https://www.tennistv.com/

[14] Adobe, "Stock photos, royalty-free images, graphics, vectors & videos," Adobe Stock, 2019. https://stock.adobe.com/

[15] "Smart Polygon - Roboflow Docs," Roboflow.com, 2023. https://docs.roboflow.com/annotate/use-roboflow-annotate/smart-polygon (accessed Dec. 03, 2023).

[16] "Roboflow Universe: Open Source Computer Vision Community," Roboflow. https://universe.roboflow.com/

[17] "AI-powered coding, free of charge with Colab," Google, May 17, 2023. https://blog.google/technology/developers/google-colab-ai-coding-features/

[18] "How to Train Detectron2 Segmentation on a Custom Dataset," colab.research.google.com. https://colab.research.google.com/drive/16jcaJoc6bCFAQ96jDe2HwtXj7BMD_-m5

[19] PyTorch, "PyTorch," Pytorch.org, 2023. https://pytorch.org/

[20] "NVIDIA Tesla T4 GPU," www.itcreations.com. https://www.itcreations.com/nvidia-gpu/nvidia-tesla-t4-gpu#:~:text=Featuring%2013.6%20billion%20transistors%2C%20the (accessed Dec. 03, 2023).

[21] Hyönä, Li, and Oksama, "Eye Behavior During Multiple Object Tracking and Multiple Identity Tracking," Vision, vol. 3, no. 3, p. 37, Jul. 2019, doi: https://doi.org/10.3390/vision3030037.

[22] K. Boyd, "Depth Perception," American Academy of Ophthalmology, Mar. 23, 2018. https://www.aao.org/eye-health/anatomy/depth-perception#:~:text=Depth%20perception%20is%20the%20ability

[23] C. Zhao et al., "How do humans perform in multiple object tracking with unstable features," Frontiers, https://www.frontiersin.org/articles/10.3389/fpsyg.2020.01940/full (accessed Dec. 7, 2023).