Rock-Paper-Scissors Detection and Game Automation Using YOLO

Mahan Veisi[1], Shayan Kebriti[2], Erfan Abedi[3]
Shahid Beheshti University, Tehran, Iran.
Emails: [1]mahan8292@gmail.com, [2]shayankebriti@gmail.com, [3]erfuun.abd@gmail.com

Abstract. This report describes the development of a system to automate the Rock-Paper-Scissors game using deep learning techniques and real-time object detection. The YOLOv11 model was utilized to recognize and classify hand gestures (Rock, Paper, Scissors) with reliable accuracy. The project includes dataset creation, preprocessing, and augmentation to improve model performance. The system integrates these elements into a functional game environment, featuring automated cheating detection and winner recognition to provide an engaging user experience. The results demonstrate the successful implementation of a practical object detection application in a controlled setting.

1 Introduction

The Rock-Paper-Scissors game is a simple and widely known hand gesture-based game, making it a suitable choice for exploring computer vision and deep learning techniques. This report outlines the development of a system that detects and classifies hand gestures in real-time using the YOLOv11 object detection model. The project leverages the capabilities of modern machine learning to create an interactive gameplay system.

A significant aspect of this project was preparing a suitable dataset, as there were limited publicly available datasets tailored specifically for Rock-Paper-Scissors gesture detection. To address this, a dataset was created and annotated using Roboflow, along with preprocessing and augmentation steps to enhance model training. The trained YOLOv11 model was fine-tuned to achieve accurate gesture classification, and its performance was evaluated using metrics such as precision, recall, and mean Average Precision (mAP).

This project demonstrates the integration of computer vision and deep learning to automate a real-time interactive system. The Rock-Paper-Scissors game was further enhanced with features like cheating detection and winner recognition, providing a complete and functional example of applying object detection techniques to a practical problem.

2 Dataset Preparation

2.1 Data Collection and Strategy

The quality and quantity of data play a crucial role in achieving high accuracy in any machine learning model, particularly in object detection tasks. Given the nature of our project, where accurate classification of Rock-Paper-Scissors gestures is required, our initial focus was on curating a clean and well-structured dataset. A larger and more diverse dataset generally leads to improved generalization of the model, making it more robust in real-world scenarios.

A major challenge we encountered was the **lack of publicly available datasets** that matched our specific requirement—detecting **both the hand gestures and their corresponding class labels** (Rock, Paper, or Scissors). While some datasets existed for general hand detection or basic classification, there was **no comprehensive dataset** that could detect **both the presence and type of gesture** in an image. Due to this limitation, we decided to create our own dataset from scratch.

To efficiently **annotate and manage the dataset**, we utilized **Roboflow**, a widely-used platform for dataset annotation and augmentation. Roboflow provides an intuitive interface for

bounding box annotation, dataset preprocessing, augmentation, and conversion into formats compatible with deep learning models. It also facilitates collaboration, allowing multiple contributors to **combine and refine** datasets seamlessly. Figure 1 illustrates an example of an annotated image using Roboflow.

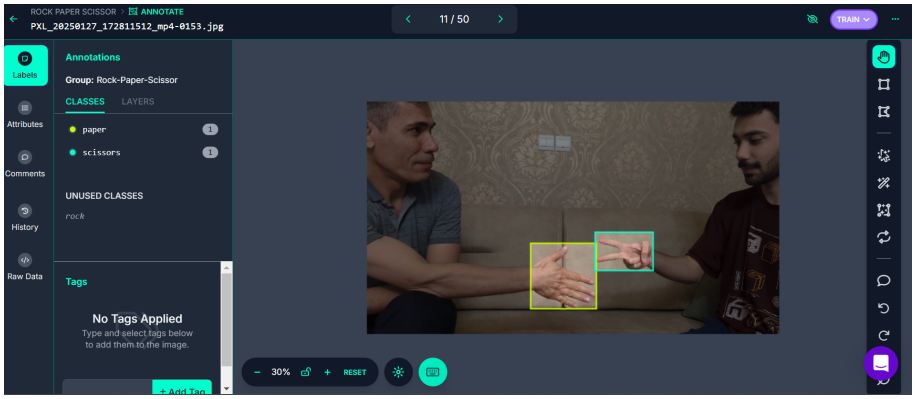

Figure 1: **Sample Image Annotation in Roboflow.** This figure shows a sample annotated image, where bounding boxes have been drawn around hands performing Rock, Paper, or Scissors gestures.

2.2 Data Collection Process

To ensure sufficient data diversity, we **recorded multiple gameplay sessions** under different conditions, including:

- **Various lighting environments** (indoors, outdoors, bright, and dim).
- **Different players** with varying hand shapes and sizes.
- **Diverse backgrounds**, including crowded areas, to enhance real-world performance.

Each recorded session was **manually reviewed**, and frames were extracted where gestures were clearly visible. We focused on maintaining **class balance** among Rock, Paper, and Scissors gestures. However, since every round of Rock-Paper-Scissors begins with a ”Rock” gesture during the countdown, there was **a natural tendency for Rock to appear more frequently**. Despite our efforts to balance the dataset, this imbalance was unavoidable to some extent.

2.3 Annotation and Dataset Statistics

After collecting raw images, we **annotated** each image using Roboflow by drawing bounding boxes around hands performing Rock, Paper, or Scissors gestures. Each annotation included:

- **Class labels** (Rock, Paper, or Scissors).
- **Bounding box coordinates** (x, y, width, height).

Following the annotation process, we successfully generated **1,500 labeled images** as part of our primary dataset. Figure 2 presents an overview of our **annotation workflow** in Roboflow.

2.4 Collaborative Data Sharing

Given the substantial data requirements for training a robust model, we **collaborated with multiple research groups** to **share and exchange datasets**. This collective approach enabled us to **expand our dataset efficiently** while maintaining high-quality annotations.

The following table (Table 1) presents the **contributions of each group** involved in dataset creation and sharing. Each group created and annotated their dataset, which was later merged to form a larger dataset.

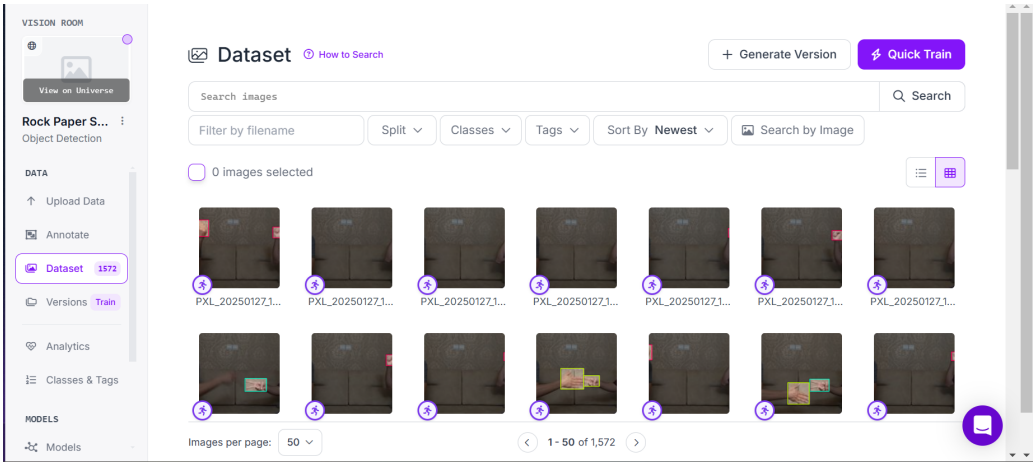

Figure 2: Dataset workspace in Roboflow.

Group Name	Number of Images
Erfan Abedi, Mahan Veisi, Shayan Kebriti	1500
Mohammad Matin Momeni, Amir Mohammad Piran, Sobhan Soririan	1300
Reyhaneh Ramazani, Mahdis Sepahvand	650 (shared with Group 3 and 5)
Fatemeh Mirzaei, Rashin Rahnemoun, Babak Fathi	650 (shared with Group 2 and 3)
Kosar Pakzad, Mobina Shahbazi	650 (shared with Group 2 and 5)
Amirali Vakili, Salar Jahanshiri	400 (shared with Group 7)
Sina Arabi, Sadaf Aghili, Arnoosha Najafi	400 (shared with Group 6)

Table 1: **Dataset Contributions by Groups.** Each group independently collected and labeled their data. Some groups shared their data with multiple other groups, facilitating dataset expansion.

2.5 Dataset Expansion and Refinement

At this stage, we had successfully gathered around **4,000** labeled images, providing a solid foundation for training our object detection model. Our first model, `fineTuned_best_V1.pt`, was trained on this dataset. However, during training, concerns arose regarding dataset size and generalization capability. To further enhance model performance, we sought additional data sources.

Additional Data Sources:

- **External Rock-Paper-Scissors Dataset**: We discovered an external dataset containing **3,500 images** that matched our required annotation format with properly labeled Rock, Paper, and Scissors classes.
- **Background Images**: To ensure that our model does not always assume the presence of hand gestures, we incorporated **1,500 background images** without any annotations. These images were selected from diverse environments to improve robustness against false detections.
- **General Hand Detection Data**: An additional **1,200 images** were sourced from various datasets that originally contained Rock-Paper-Scissors gestures but were labeled under a single class, "Detected Rock-Paper-Scissors." We manually re-annotated these images to classify them correctly into the three individual gesture classes.

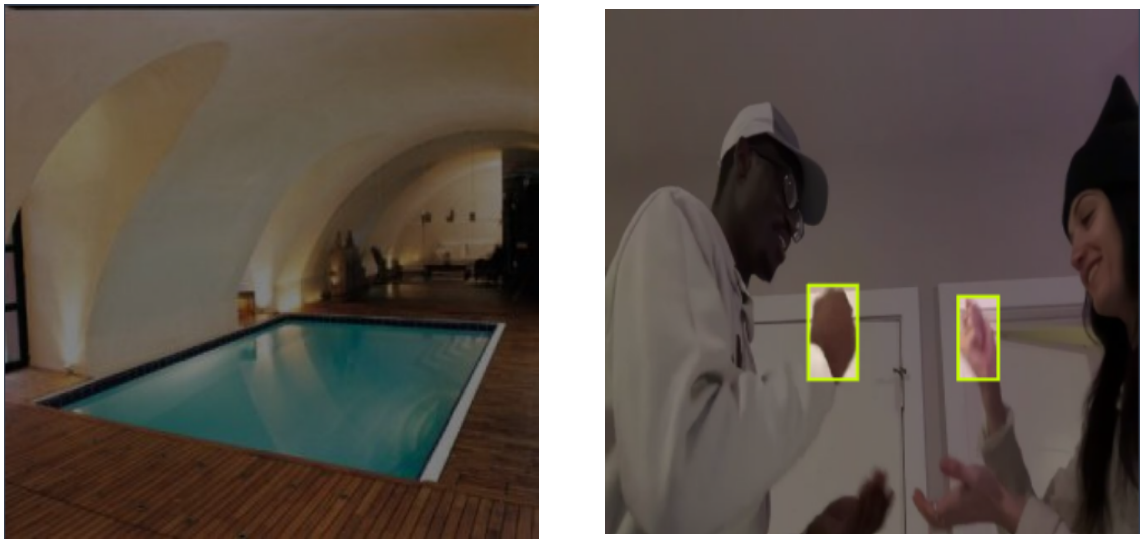

Figure 3: **Additional Dataset Samples.** (Left) A background image without any Rock-Paper-Scissors gestures to improve generalization. (Right) A sample image manually re-annotated to classify gestures correctly.

After incorporating these additional data sources, our dataset reached approximately **9,000 images**. Figure 4 shows the updated dataset workflow in Roboflow.

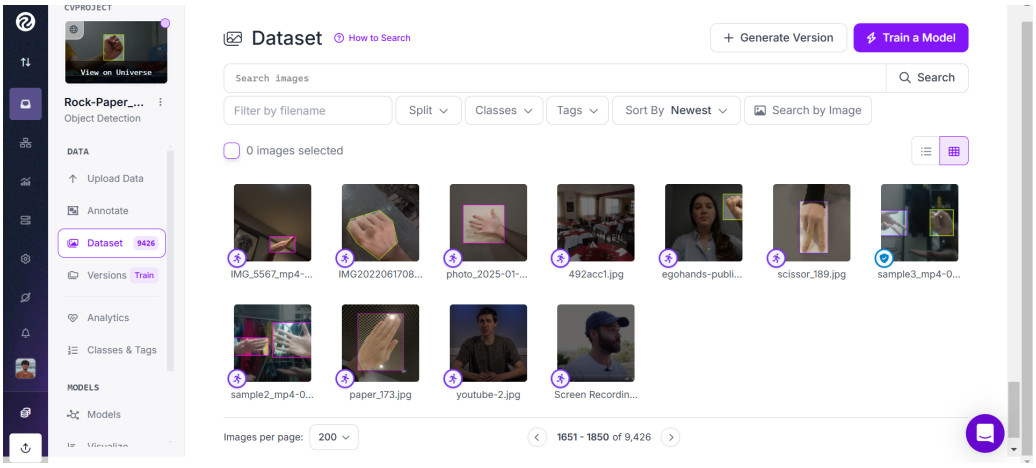

Figure 4: **Updated Dataset Workflow in Roboflow.** This figure presents an overview of the finalized dataset, consisting of around 9,000 images after merging external data sources and manual annotations.

2.6 Dataset Splitting and Augmentation

To effectively train our model while evaluating its performance, the dataset was split as follows:

- **90%** (**8,100** images) for training.
- **10%** (**900** images) for validation.

Preprocessing and Augmentation: Before feeding the images into the model, we applied preprocessing and data augmentation techniques to enhance model generalization. The following transformations were applied:

- **Auto-Orient**: Adjusted image rotation for correct orientation.

- **Resize**: All images were resized to a fixed dimension of **640×640**.

Additionally, we applied the following augmentations:

- **Flipping**: Random **horizontal** and **vertical** flips.
- **Rotation**: Random rotation between **-11° to +11°**.
- **Saturation Adjustment**: Random saturation shifts between **-16% to +16%**.
- **Brightness Adjustment**: Random brightness variations between **-7% to +7%**.

As a result of the applied augmentations (**2X augmentation factor**), the total dataset size increased to **17,314 images**.

2.7 Final Dataset Statistics

After preprocessing, augmentation, and dataset merging, the final distribution of the three gesture classes was as follows:

- **Scissors**: 5,636 images
- **Paper**: 5,020 images
- **Rock**: 9,044 images

Figure 5 presents a visual representation of class distribution.

Figure 5: **Class Distribution of Final Dataset.** The chart illustrates the number of images assigned to each gesture class.

This dataset, enriched through multiple sources and rigorous preprocessing, provided a strong foundation for training our model. In the next section, we discuss the model training process and performance evaluation.

3 Model Training and Evaluation

This section details the training and evaluation of the YOLOv11 model for Rock-Paper-Scissors gesture detection. The model was fine-tuned using transfer learning techniques and evaluated based on key performance metrics.

3.1 YOLOv11 Model and Pre-Trained Weights

For this project, we employed **YOLOv11**, a state-of-the-art object detection model with enhanced feature extraction and improved inference speed. Initially, we explored a pre-trained model available from an existing **YOLOv11 Rock-Paper-Scissors detection project**[1].

- **Observations**: Using the pre-trained weights, we noticed that the model exhibited **low accuracy in detecting Paper and Scissors**, particularly in real-time video inference.
- **Approach**: Instead of training from scratch, we utilized these pre-trained weights and fine-tuned the model on our enhanced dataset.

3.2 Training Configuration

The training process was conducted on the **Kaggle environment** due to its GPU acceleration capabilities. Since training was time-intensive (**approximately 8 hours per model**), we trained the model under two configurations:

- **V1 Model**: Trained on our initial **5K dataset** (student version).
- **V2 Model**: Trained on the full **9K dataset**, including external data.

The complete training history and source code can be found in the project repository:

`notebook/Rock-Paper-Scissors-Object-Detection-Yolo11-training`

Additionally, a **Google Colab** version of the code is available for reference[2].

3.3 Training Parameters

The model was trained using the following hyperparameters:

- **Epochs**: 250 (for both V1 and V2 models).
- **Batch Size**: 64 (*Batch size of 128 caused GPU memory issues*).
- **Image Size**: 640×640 pixels.
- **Optimizer**: Adaptive momentum estimation (Adam).
- **Validation**: Enabled at the end of each epoch.

The training was conducted using the following command:

[1] Gholamreza Dar, *YOLOv11 Rock-Paper-Scissors Detection*, GitHub Repository. Available at: `https://github.com/Gholamrezadar/yolo11-rock-paper-scissors-detection`.

[2] Colab Notebook: *YOLOv11 Rock-Paper-Scissors Training*, `https://colab.research.google.com/drive/1Bi7lbf_yCVDDLcW-oUydFGI7fHMxbqRo?usp=sharing`.

```
results = model.train(
    data=dataset_yaml_path,
    epochs=80,
    imgsz=640,
    batch=64,
    name="yolo11n_finetuned",
    val=True,
    plots=True
)
```

3.4 Performance Comparison: V1 vs. V2

We evaluated the performance of both the **V1** and **V2** models. Figure 6 illustrates the **Precision-Recall curves** for both versions.

- The V1 model achieved **mAP@0.5 = 0.98**, which was already a strong result.
- The V2 model further improved performance, reaching **mAP@0.5 = 0.982**, showing noticeable enhancements, particularly for the Paper and Scissors classes.

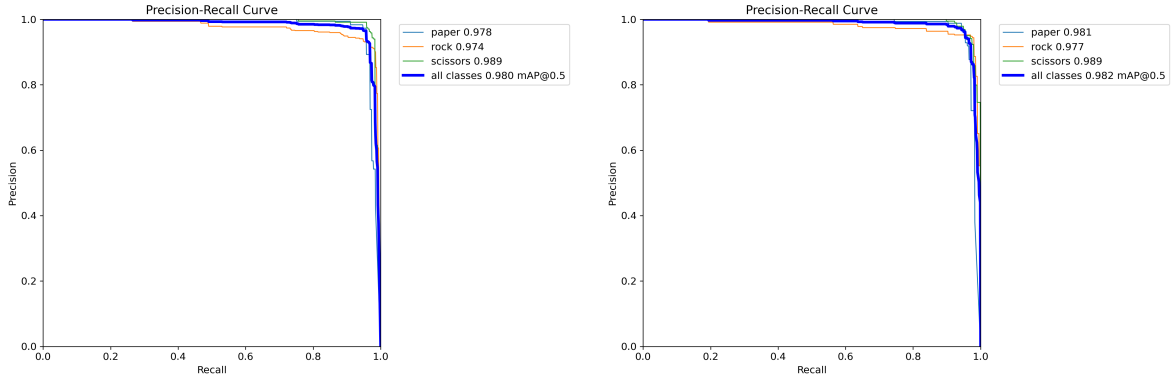

Figure 6: **Precision-Recall Curves.** (Left) V1 model trained on 5K dataset. (Right) V2 model trained on 9K dataset, showing improved results.

After confirming performance improvements in V2, we proceeded with evaluating its detection capabilities.

3.5 Model Predictions

The trained **V2 model** was tested on validation images to assess its predictive accuracy. Figure 7 presents a sample image with predicted bounding boxes.

3.6 Evaluation Metrics

To further assess model performance, we analyzed multiple evaluation metrics, as shown in Figure 8:

- **Confusion Matrix**: Displays classification accuracy for each gesture.
- **Precision-Confidence Curve**: Measures the confidence level required for precise predictions.

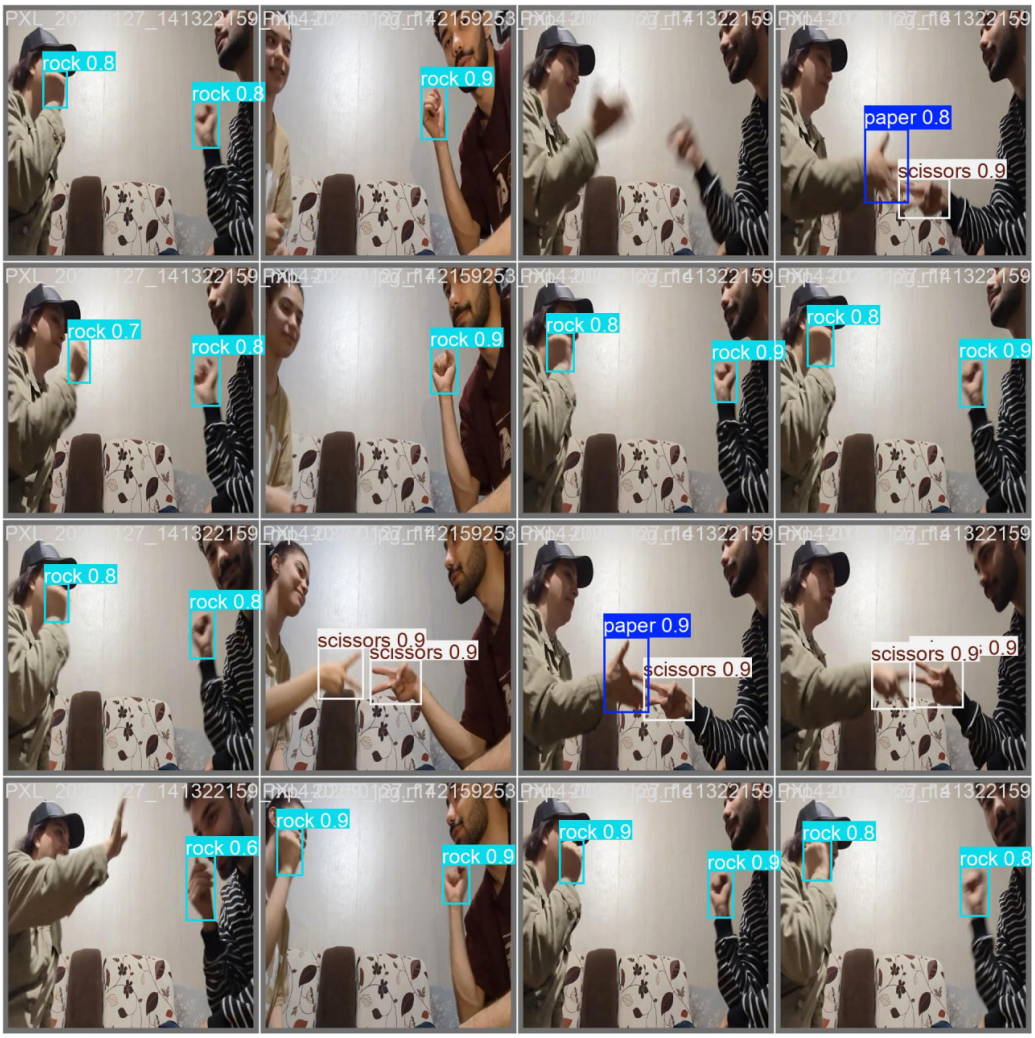

Figure 7: **Validation Sample Predictions.** The trained YOLOv11 model accurately detects and classifies hand gestures in the validation set.

- **Recall-Confidence Curve**: Evaluates the trade-off between recall and confidence thresholds.

These evaluations confirm that the **fine-tuned YOLOv11 model achieved high accuracy**, successfully detecting Rock-Paper-Scissors gestures with **minimal false detections**.

4 Game Design and Implementation

The Rock-Paper-Scissors game is designed to be fully automated using computer vision and deep learning techniques. The system captures real-time hand gestures, tracks facial landmarks, and overlays visual effects such as masks and crowns. The primary goal is to ensure fair gameplay while making the experience visually engaging and interactive. Figures 9 and 4.3 illustrate the game flow and key stages.

4.1 Game Flow and Mechanics

The game follows a structured flow where players start by showing a 'rock' gesture to initiate the countdown. During the countdown, they move their hands across specific boundary lines. At the end of the countdown, they reveal their final gestures, and the system determines the winner based on traditional Rock-Paper-Scissors rules. If a player changes their gesture after the countdown, the system flags it as cheating. The winner is celebrated with a crown overlay

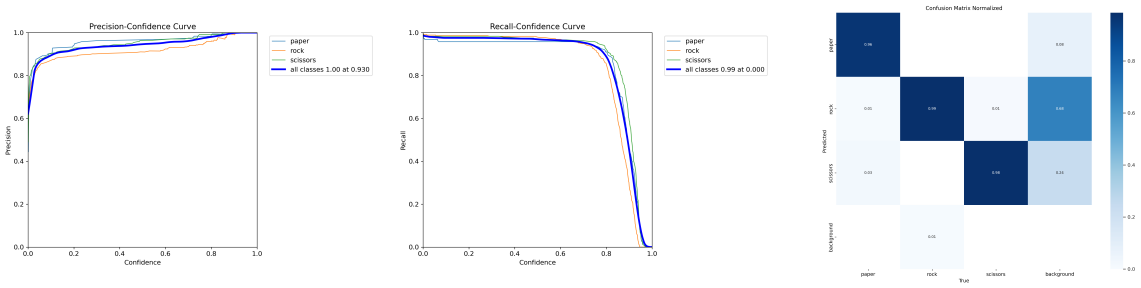

Figure 8: **Evaluation Metrics.** (Left) Precision-Confidence Curve, (Middle) Recall-Confidence Curve, (Right) Confusion Matrix.

and camera zoom, creating an engaging gameplay experience. Figure 9 visualizes this flow, highlighting the sequence from the home page to the winner declaration.

4.2 Computer Vision Models and Techniques

To ensure real-time processing and accurate detection, multiple deep learning models are utilized. The YOLO model is fine-tuned to recognize hand gestures corresponding to 'rock', 'paper', and 'scissors'. Meanwhile, facial detection and landmark tracking are managed using Dlib's 68-point facial landmark predictor and MediaPipe's FaceMesh. These tools allow the game to accurately position visual overlays such as masks and crowns while maintaining fluid motion tracking.

4.3 Cheating Detection and Winning Recognition

To prevent players from altering their gestures post-countdown, the system monitors for gesture inconsistencies. If a player modifies their hand position or changes their gesture after the countdown expires, a cheating violation is flagged. In response, a red mask is dynamically applied over the player's face using real-time facial landmark tracking, as seen in Figure 4.3 (top-left). Upon determining the winner, a celebratory animation is triggered, overlaying a golden crown on the victorious player's head, as shown in Figure 4.3 (top-right). Additionally, the home page provides the initial setup for players, as illustrated in Figure 4.3 (bottom-left).

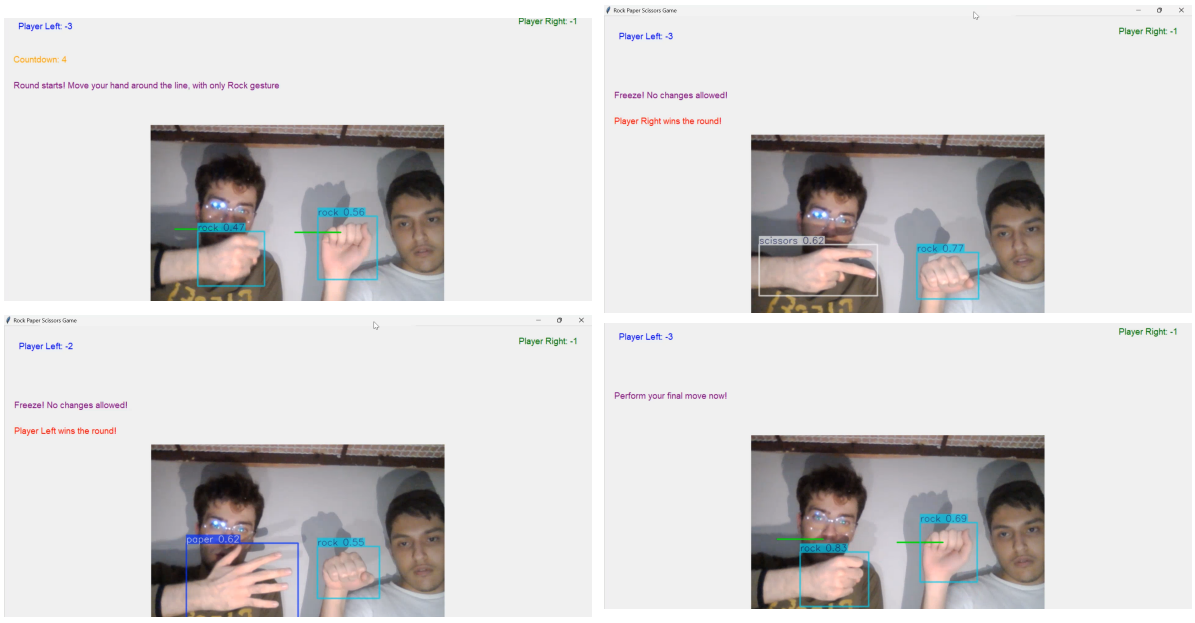

Figure 9: **Game Flow Visualization.** The flow begins with the home page, followed by players using fists to start the game, executing countdown movements, and revealing their gestures to determine the winner.

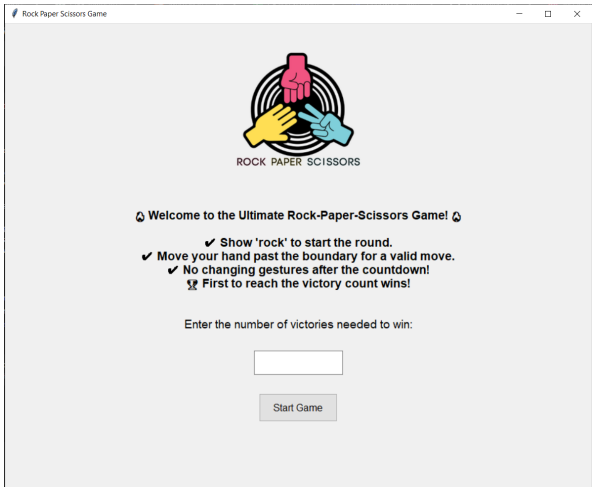

Figure 10: **Game Home Page.** The initial interface where players prepare to play.

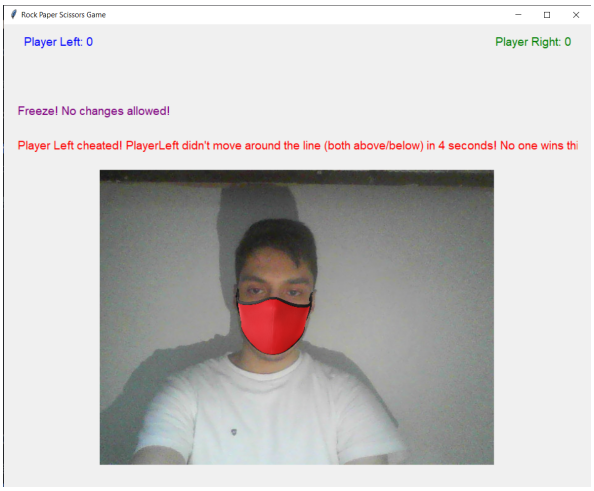

Figure 11: **Cheating Detection.** A red mask is applied to the cheater's face to indicate rule violation.

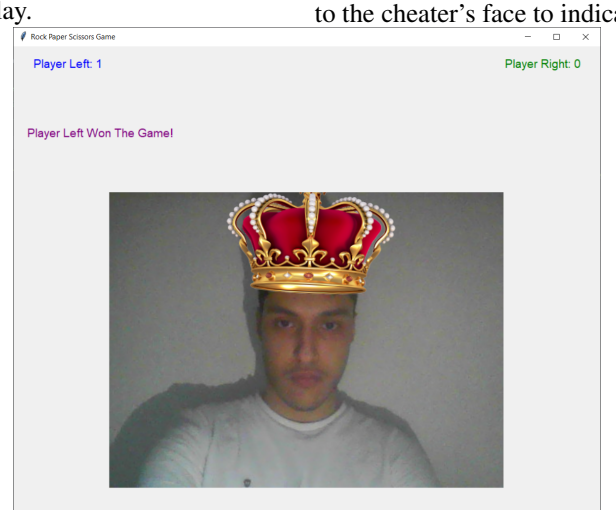

Figure 12: **Winner Celebration.** The winner is awarded a golden crown with a zoom-in effect.

5 Conclusion

In this report, we presented the development and evaluation of a system for automating the Rock-Paper-Scissors game using deep learning techniques and real-time object detection. The YOLOv11 model was trained and fine-tuned on a carefully curated and augmented dataset, achieving strong performance in detecting and classifying hand gestures.

The two versions of the trained model demonstrated reliable accuracy:

- **V1 Model**: Achieved a mean Average Precision (mAP@0.5) of **0.98**, indicating effective classification for the initial dataset of 5,000 images.
- **V2 Model**: Improved to a higher mAP@0.5 of **0.982** after training on the expanded dataset of 9,000 images, showing enhanced detection performance across all gesture classes.

The system was successfully integrated into an interactive gameplay environment, featuring additional functionalities such as cheating detection and dynamic winner recognition. The results confirm the feasibility of applying deep learning models to real-time applications, such as object detection in games. While this project demonstrates the effectiveness of YOLOv11 for this specific task, future work could explore further optimization techniques, larger datasets, and the integration of additional gesture recognition features for enhanced user experience.