Real-Time Hand Tracking for Gesture-Based Tennis Simulation

Jennifer Chen, Kate Choi, Rohan Phanse

CPSC 5800

1 Introduction

Physical sports such as tennis or pickleball provide engaging and active forms of entertainment, but they often require specialized equipment, access to a sufficiently large playing space, and appropriate environmental conditions. These constraints make casual or frequent play difficult for many people. While virtual alternatives such as Wii Sports and virtual reality simulations reduce the need for physical space, they still rely on specialized platforms and accessories, which introduce additional cost and accessibility barriers.

Motivated by these limitations, our project explores an alternative solution that capitalizes on the accessibility of virtual play to allow anyone to play simulated tennis with just a laptop. We use computer vision to track a player's hand movements using only a standard webcam, eliminating the need for specialized hardware.[1] By mapping natural hand gestures to the movement of a simulated player and tennis racket, players can interact with the game in a physically intuitive way that is analogous to real-world tennis play.

Our project extends an existing Unity-based tennis simulation that originally accepted keyboard inputs for controlling the player. We modified this system to instead accept real-time webcam input processed by a hand detection model. To enable intuitive control, we defined two distinct interaction modes based on hand pose: closed-hand movement controls lateral player movement across the court, while open-hand movement initiates racket swings in the same lateral direction. A swing is triggered when the velocity of an open hand surpasses a predefined threshold, allowing the direction and speed of the hand motion to directly influence the simulated racket's behavior.

To detect hands and distinguish between open and closed hand poses, we explored a variety of approaches. These included training YOLO models and leveraging MediaPipe for hand detection. We evaluated methods that directly classified open and closed hands, as well as approaches that first detected hand bounding boxes and then applied a separate classifier to determine hand pose. Through experimentation, we determined that MediaPipe Hands provided the best balance of accuracy, robustness, and real-time performance for our application.

During this process, we encountered several challenges. One challenge we faced was finding appropriate data to train and test our models, as we needed datasets that contained

[1] `https://github.com/rohanphanse/CPSC5800-Final`

open and closed hand poses with bounding boxes. To account for a lack of existing datasets designed for this purpose, we found and repurposed other datasets (e.g., American Sign Language and broader gesture datasets) to aggregate suitable data for our project.

Another significant challenge involved integrating the computer vision outputs with the Unity game in a way that felt seamless and intuitive to the player. This required careful design of state transitions between player movement and racket swings, as well as robust detection of swing start and end points to avoid unintended actions or lag. Balancing responsiveness with stability was critical to creating a smooth gameplay experience that felt natural rather than error-prone.

Through this simulation, we aim to remove many of the constraints associated with real-world sports and existing virtual simulations. Our goal is to enable users to engage in tennis-like gameplay anytime and anywhere, using only a laptop with a webcam, making the experience more accessible, affordable, and convenient while preserving the core mechanics of the sport.

2 Related Work

Object detection and classification are foundational problems in computer vision, concerned with identifying and localizing objects within an image. Early approaches to these tasks relied on the use of handcrafted features, extracting relevant attributes such as edges and colors from images with methods like SIFT [1]and histograms of oriented gradients (HOG) [2], paired with sliding-window classifiers or region proposal mechanisms.

While successful in controlled settings, handcrafted feature approaches were computationally expensive and faced challenges when facing more complex tasks that required algorithms to be invariant to factors such as scale, lighting, and viewpoint. These limitations were addressed by the introduction of deep convolutional neural networks (CNNs), which apply learnable filters called kernels across an image to produce sophisticated feature maps.

Current object detection and classification methods are largely shaped by region-based and single-stage learning frameworks. Region-based methods, such as R-CNN (Region-Based CNN) [3], utilize learned region proposals to identify likely object locations in an image, followed by classification and bounding box refinement. These methods are able to achieve high accuracy on object detection and classification problems, but at the cost of greater computational complexity.

By contrast, single-stage learning models learn to perform tasks in one step, optimizing for speed and efficiency. Key examples of single-stage models are YOLO (You Only Look Once) [4] and SSD (Single Shot MultiBox Detector) [5], which are object detection models that predict object bounding boxes and classes in one forward pass. The speed of single-stage methods make them optimal choices for applications that require low latency, such as real-time tracking.

Hand detection and classification present unique challenges compared to general object detection, due to the high articulation of the human hand, frequent self-occlusion, and large variation in appearance across viewpoints and lighting conditions. Single-stage detectors such as YOLO have been widely applied to hand detection tasks due to their real-time performance and architectural simplicity. These models can be trained to detect hands directly with bounding boxes.

An alternative approach focuses on hand pose estimation through landmark prediction rather than bounding-box classification. For example, MediaPipe Hands [6] employs a two-stage pipeline that first detects coarse hand regions and then regresses a set of hand keypoints representing finger joints and palm structure. Landmark-based representations enable fine-grained reasoning about hand configuration and motion, and they have become particularly popular in real-time interactive applications due to their temporal stability and low inference latency.

The development of large-scale annotated datasets has played a central role in advancing object detection and classification methods. Early breakthroughs in deep learning for vision were enabled by ImageNet [7], a massive hand-annotated dataset of images with unprecedented scale and diversity that supported the emergence of CNNs such as AlexNet [8]. While ImageNet proved instrumental for supporting object detection tasks, it lacked localization annotations. This motivated datasets like COCO (Common Objects in Context) [9], which includes bounding boxes and segmentation masks for objects in its images across a wide range of everyday categories. Within COCO, hands are present in pictures of people, but are not explicitly annotated. COCO-Hand extends the original COCO dataset by explicitly annotating hand regions, enabling more focused study of hand detection and localization [10]. More recent datasets such as HaGRID (HAnd Gesture Recognition Image Dataset) [11] provide additional large-scale, labeled hand gesture data designed for detection tasks, capturing a variety of hand poses, viewpoints, and contexts.

3 Methodology

Our system enables gesture-based control of a tennis simulation using only a standard webcam and real-time computer vision. The overall pipeline consists of three main components: webcam input processing, hand pose classification, and game control within a Unity-based tennis simulation, MinimumTennis[2]. MinimumTennis was designed to take keyboard presses as input to control the player and racket in the simulation. We modified the game to instead accept control signals derived from real-time hand tracking using MediaPipe Hands on webcam input.

Video input is captured continuously from the user's webcam and processed frame by frame using MediaPipe Hands. MediaPipe detects the presence of hands in each frame and estimates a set of 21 hand landmarks, including the wrist and finger joints. Based on the configuration of these landmarks, the system classifies each detected hand as either open or closed. This binary pose classification serves as the primary mode switch for interaction, determining whether the user's hand controls player movement or racket swings.

To classify whether a hand is open or closed using MediaPipe Hands, we develop a mathematical approach detailed in Algorithm 1. The method computes an average extension score by comparing the distance from each fingertip to the wrist with the distance from the corresponding proximal interphalangeal (PIP) joint (i.e. the middle knuckle) to the wrist. For open hands, the fingertip lies farther from the wrist than the middle knuckle, producing ratios greater than one, whereas for closed hands the fingertip curls inward and

[2]`https://github.com/open-video-game-library/MinimumTennis`

Algorithm 1 Open vs. Closed Hand Classification Using MediaPipe Hands Landmarks

```
Require: image, open_threshold = 1.1
Ensure: hand_state $\in \{0 \text{ (open)}, 1 \text{ (closed)}\}$
 1: Get 2D hand landmarks $\{(x_i, y_i)\}_{i=0}^{20}$ from MediaPipeHands(image)
 2: wrist_point ← $(x_0, y_0)$
 3: finger_tip_indices ← $[4, 8, 12, 16, 20]$
 4: finger_pip_indices ← $[2, 6, 10, 14, 18]$
 5: distance_ratios ← [ ]
 6: for all $(tip\_index, pip\_index) \in (finger\_tip\_indices, finger\_pip\_indices)$ do
 7:     finger_tip_point ← $(x_{tip\_index}, y_{tip\_index})$
 8:     PIP refers to the Proximal Interphalangeal Joint (i.e. the middle knuckle)
 9:     finger_pip_point ← $(x_{pip\_index}, y_{pip\_index})$
10:     pip_to_wrist_distance ← $\|\text{finger\_pip\_point} - \text{wrist\_point}\|_2$
11:     if pip_to_wrist_distance $\leq 10^{-6}$ then continue
12:     end if
13:     tip_to_wrist_distance ← $\|\text{finger\_tip\_point} - \text{wrist\_point}\|_2$
14:     Append tip_to_wrist_distance / pip_to_wrist_distance to distance_ratios
15: end for
16: average_extension_ratio ← $\frac{1}{|\text{distance\_ratios}|} \sum_{r \in \text{distance\_ratios}} r$
17: if average_extension_ratio $\geq$ open_threshold then return 0
18: else return 1
```

becomes closer to the wrist than the PIP joint. These ratios are averaged across fingers and thresholded at 1.1 to robustly distinguish open and closed poses, achieving the best performance in Table 3 compared to ResNet-based classifiers.

When a closed hand is detected, the system interprets the gesture as a command for lateral player movement. Specifically, the x-coordinate of the wrist landmark is checked to identify which of three possible zones it lies in: the leftmost 35% of the frame, the rightmost 35% of the frame, or the central 30%. If the wrist lies in the left zone, the system continuously issues a command to move the in-game player left across the court; if it lies in the right zone, the player is moved right. When the wrist is positioned in the central zone, no movement command is issued, allowing the player to remain stationary. Movement commands are applied continuously as long as the closed hand remains within the corresponding zone.

When an open hand is detected, the system calculates the speed of detected hand motion. Speed of the hand is computed between consecutive frames using the wrist landmark, and is defined as the sum of the absolute differences in the x- and y-coordinates between frames. If this value exceeds a predefined threshold of 0.05, the system triggers the tennis simulation's standard racket swing function, causing the player to attempt a shot. To prevent repeated unintended swings from a single gesture, a cooldown period of 0.5 seconds is enforced after each triggered swing, during which additional motion above the threshold does not result in further hit commands.

4 Data

Our task requires binary classification of hand poses into open and closed hands. Relatively few datasets are explicitly designed for this distinction, so we combine one such dataset for evaluation with several larger, high-quality hand gesture datasets that we adapt for training. We use these datasets to experiment with different model and training possibilities to compare their performance on hand detection and classification tasks.

We use the Hand Gestures Computer Vision Dataset[3] (uploaded to Roboflow by user “hand gestures”) as our primary test set (titled Open-Closed-1k for the remainder of this paper for clarity). This dataset contains 1,077 RGB images of hands annotated with bounding boxes and labeled as either open or closed. The images depict hands with diverse skin tones, captured from multiple viewpoints and distances, and against a wide variety of backgrounds and lighting conditions. Although comparatively small, its realistic diversity and provided open/closed labels make it well-suited for evaluating model performance.

For training, we rely primarily on larger gesture datasets that can be repurposed for open/closed hand classification. The largest of these is HaGRIDv2 [12] (HAnd Gesture Recognition Image Dataset Version 2), which contains 1,086,158 FullHD RGB images spanning 33 gesture classes, with bounding box annotations. HaGRIDv2 was crowdsourced from 65,977 subjects across over 100 countries, ages 18 and over, with subject-to-camera distances from 0.5–4 meters, and diverse natural lighting conditions. This closely matches our intended deployment scenario of webcam-based hand detection. We identify four gesture classes relevant to our task—fist, palm, stop, and stop inverted—and reclassify fist as closed hand, and palm, stop, and stop inverted as open hand. From this subset, we construct a balanced training set of 15,000 images (7,500 open, 7,500 closed; 18,139 bounding boxes) and a validation set of 1,500 images (750 open, 750 closed; 1,836 bounding boxes).

We also include COCO-Hand [10], a large-scale hand detection dataset from the MS COCO framework [13], which provides diverse real-world images with high-quality bounding box annotations for hands. As COCO-Hand only identifies hands without distinguishing hand poses, we use it to train and validate YOLO models to detect hands (not to classify open/closed hands). Specifically, the subset we use contains 10,000 training images (19,075 hand bounding boxes) and 1,000 validation images (1,811 hand bounding boxes).

In addition, we draw on ASL gesture datasets, which are widely available and carefully curated due to demand. As they are not originally labeled for open/closed classification, we manually categorize ASL letters B, C, and F as open-hand poses, and A, E, M, N, O, S, and T as closed-hand poses. Our first ASL dataset is MU HandImages ASL[4], obtained via the Kaggle ”American Sign Language Dataset” uploaded by Ayush Thakar. This dataset contains 2,515 color images of 36 static ASL gestures collected from 5 individuals. Images were captured under systematically varied lighting conditions (top, bottom, left, right, and diffuse sources), and color-based segmentation was used to crop the hands against a black background. Because bounding boxes are not provided, we generate them automatically using MediaPipe Hands [14], resulting in 1,819 general hand images and 326 open/closed hand images corresponding to our selected letters.

[3]`https://universe.roboflow.com/hand-gestures-tytyx/hand-gestures-9iwxv`

[4]`https://www.kaggle.com/datasets/ayuraj/asl-dataset`

Model / Split	COCO-Hand		HaGRIDv2		Open-Closed-1k		MU ASL		HG ASL		RPS		Total	
	Images	BB	Images	BB	Images	BB	Images	BB	Images	BB	Images	BB	Images	BB
Hand														
Train	10,000	19,075	15,000	18,139	–	–	1,819	1,819	4,618	4,618	2,629	2,629	34,066	46,280
Validation	1,000	1,811	1,500	1,836	–	–	–	–	–	–	–	–	2,500	3,647
Test	4,000	7,716	6,000	7,169	1,077	1,077	–	–	–	–	–	–	11,077	15,962
Open/Closed Hand														
Train	–	–	15,000	18,139	–	–	326	326	579	579	1,774	1,774	17,679	20,818
Validation	–	–	1,500	1,836	–	–	–	–	–	–	–	–	1,500	1,836
Test	–	–	6,000	7,169	1,077	1,077	–	–	–	–	–	–	7,077	8,246

Table 1: Dataset statistics for the single-class Hand dataset and the two-class Open/Closed Hand dataset. Columns report the number of images and the total number of hand bounding boxes (BB) for each dataset split. While bounding box annotations were automatically generated with MediaPipe Hands for MU ASL, HG ASL, and RPS to increase the amount of training data, the validation and test splits include only ground-truth bounding box annotations.

As MU HandImages ASL has limited subject diversity and tightly cropped images, we also use a second ASL dataset, Hand Gestures[5], published on Kaggle by Ashish Sharma. This dataset includes 5,119 RGB images of the 26 ASL letters, performed by at least two individuals across varied viewpoints, subject-to-camera distances, and backgrounds. After annotating images with hand bounding boxes using MediaPipe Hands, the selected subsets include 4,618 general hand images and 579 open/closed hand images.

Finally, we repurpose a Rock–Paper–Scissors dataset[6] uploaded to Kaggle by Alexandre Donciu-Julin. This dataset contains 2,717 RGB webcam images of three individuals performing the rock, paper, and scissor game gestures against a uniform gray background. We reclassify rock as closed hand and paper as open hand and generate bounding boxes with MediaPipe Hands, yielding 2,629 general hand images and 1,774 open/closed hand images.

We aggregated images from the individual datasets to construct two training datasets, Hand and Open/Closed Hand, as summarized in Table 1. The Hand dataset contains 34,066 images with 46,280 hand bounding boxes, all labeled under a single hand class. The Open/-Closed Hand Dataset includes 17,679 images with 20,818 hand bounding boxes, each labeled as either an open or closed hand. We use the Hand dataset to train single-class YOLO models for general hand detection, and the Open/Closed Hand dataset to train two-class YOLO for open/closed hand detection.

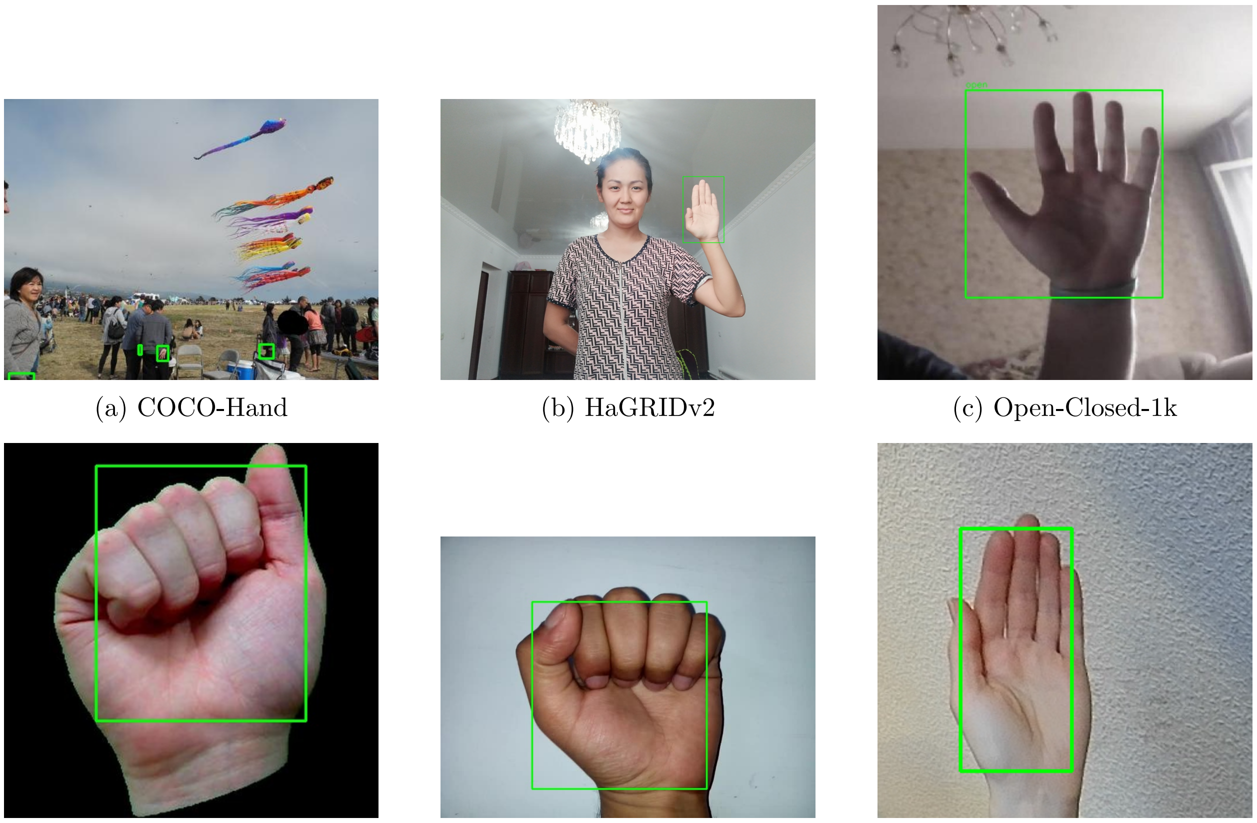
5 Implementation Details

Hand Detection Training. We trained all YOLO-based hand detection models using the Ultralytics YOLO framework[7]. The lightweight YOLOv8n architecture (3.2M parameters) was chosen to balance real-time inference requirements with strong detection accuracy. Two training configurations were evaluated: initialization from pretrained YOLOv8-COCO weights and training from random initialization. Models were trained for 50 epochs with an input resolution of 640×640 and a batch size of 32. All remaining hyperparameters

[5]`https://www.kaggle.com/datasets/ashish8898/hand-gestures`

[6]`https://www.kaggle.com/datasets/alexandredj/rock-paper-scissors-dataset`

[7]`https://github.com/ultralytics/ultralytics`


(a) COCO-Hand (b) HaGRIDv2 (c) Open-Closed-1k

(d) MU HandImages ASL (e) Hand Gestures (ASL) (f) Rock-Paper-Scissors

Figure 1: Sample images with bounding boxes from each dataset.

followed the default YOLOv8n configuration: optimization was set to `optimizer="auto"`, momentum to 0.937, initial and final learning rate to 0.01, weight decay to 5×10^{-4}, and warmup epochs to 3. Single-class YOLOv8n-Hand models were trained on the Hand dataset to localize hands, while two-class YOLOv8n-Open-Closed-Hand models were trained on the Open/Closed Hand dataset to jointly detect and classify hand poses.

Training progress was monitored using standard YOLO evaluation metrics, including mean Average Precision at an Intersection over Union (IoU) threshold of 0.5 (mAP@50) and mean Average Precision averaged over IoU thresholds from 0.5 to 0.95 in steps of 0.05 (mAP@50–95). Figure 2 shows the training curves for YOLOv8n-Hand and YOLOv8n-Open-Closed-Hand models, each trained with both pretrained weights and random initialization, including training box loss and validation mAP metrics across epochs.

Open/Closed Hand Classification Training. To evaluate open and closed hand pose classification independently of detection performance, we trained ResNet18 and ResNet50 image classifiers by repurposing the Open/Closed Hand detection dataset for classification. Each image was cropped to the annotated hand bounding box and treated as a classification example across the training, validation, and test splits. We evaluated two training configurations for each architecture: initialization from pretrained ResNet18-ImageNet-1k or ResNet50-ImageNet-1k weights and training from random initialization. ResNet18 and ResNet50 classifiers were trained for 20 epochs with an input resolution of 224×224 using

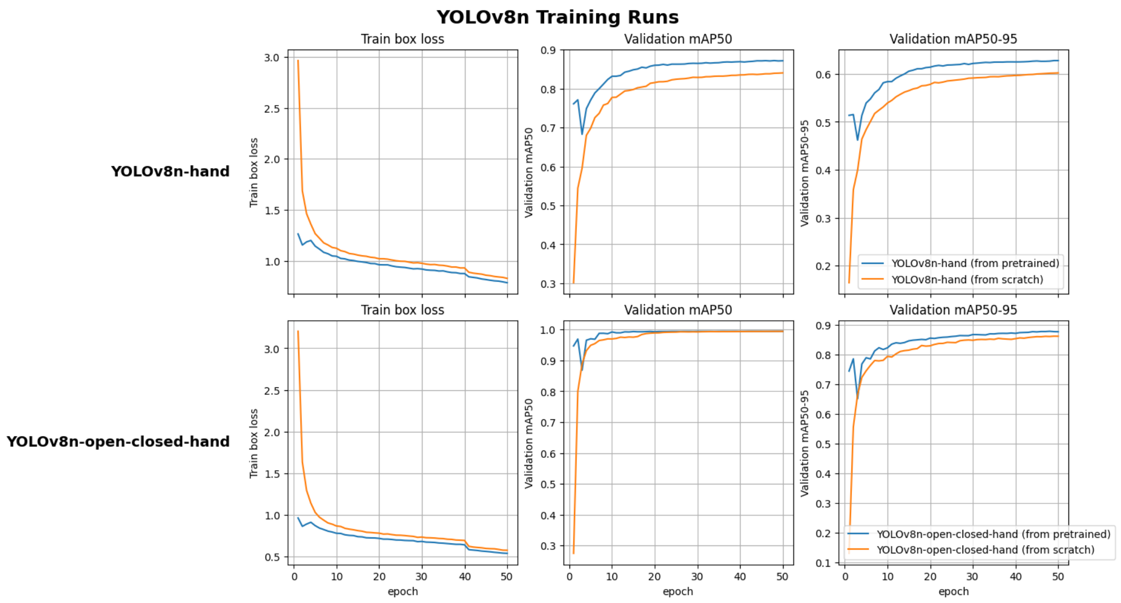


Figure 2: Training curves for YOLOv8n-Hand and YOLOv8n-Open-Closed-Hand models, trained on Hand and Open/Closed Hand datasets, respectively. Two weight initialization strategies were evaluated: training from pretrained weights and random initialization. The YOLOv8n models were trained using the Ultralytics YOLO framework, retaining default hyperparameters while setting the number of epochs to 50, the input size to 640, and the batch size to 32.

the Adam optimizer with a learning rate of 1×10^{-3} and weight decay of 1×10^{-4}, batch sizes of 64 and 32 respectively, and data augmentations including random resized cropping, horizontal flipping, color jitter, and normalization during training.

6 Results

We performed a comprehensive evaluation of hand detection and open/closed hand pose classification methods to understand their relative performance and suitability for our hand gesture-based tennis simulation. We benchmark trained YOLO models, ResNet-based pose classifiers, and MediaPipe Hands across multiple datasets to assess accuracy, robustness, and generalization under various conditions, with Open-Closed-1k serving as a left-out, diverse test set that resembles realistic gameplay conditions.

Single-Class Hand Detection. Table 2 reports single-class hand detection performance across the COCO-Hand, HaGRIDv2, and Open-Closed-1k subsets in the Hand test dataset. YOLOv8n-Hand performs strongly on HaGRIDv2 (mAP@50 of 99.49 and mAP@50–95 of 88.17) and achieves competitive performance on COCO-Hand and Open-Closed-1k, with pretrained initialization providing modest but consistent gains over training from scratch. On COCO-Hand, YOLOv8n-Hand achieves a mAP@50–95 of 31.85, substantially outperforming MediaPipe Hands. This gap reflects MediaPipe's two-hand detection limit and its tendency to produce tightly cropped bounding boxes that differ from dataset annotation conventions, whereas trained models can adapt to these box characteristics during training. The effect is

Model	COCO-Hand		HaGRIDv2		Open-Closed-1k	
	mAP@50	mAP@50–95	mAP@50	mAP@50–95	mAP@50	mAP@50–95
YOLOv8n-Hand	**66.35**	**31.85**	**99.49**	**88.17**	68.14	**32.73**
YOLOv8n-Hand (Scratch)	58.63	26.77	99.48	86.47	63.45	30.97
MediaPipe Hands	4.09	0.96	59.47	12.72	30.52	7.56
MediaPipe Hands (1.2x)	13.57	4.15	90.34	47.45	**78.08**	32.10

Table 2: Hand detection performance across single-class hand datasets. YOLOv8n-Hand (Scratch) denotes training the model from random initialization rather than from pretrained YOLOv8n-COCO weights. MediaPipe Hands (1.2x) denotes resizing all predicted bounding boxes by a factor of 1.2.

most pronounced on COCO-Hand, which contains multiple small or background hands and falls outside MediaPipe's intended close- to medium-range operating distance.

To help mitigate this mismatch between MediaPipe's tightly cropped bounding boxes and the looser ground-truth annotations used in the datasets, we additionally expanded MediaPipe-predicted bounding boxes by a factor of 1.2 in both width and height. This simple rescaling substantially improves IoU alignment, yielding large gains on HaGRIDv2 (mAP@50 from 59.47 to 90.34 and mAP@50–95 from 12.72 to 47.45) and the best mAP@50 performance on Open-Closed-1k of 78.08.

Open/Closed Hand Classification. Open/closed hand pose classification accuracy is shown in Table 3. To isolate pose classification from detection performance, ResNet-based classifiers were evaluated on cropped hand regions. ResNet18 achieves the best performance among the ResNet-based classifiers (92.37% on HaGRIDv2 and 81.34% on Open-Closed-1k), slightly outperforming the deeper ResNet50 across both subsets, with pretrained initialization yielding modest improvements over training from scratch. MediaPipe Hands, which is not designed to operate on extremely close-range hand crops, was instead evaluated on full images containing a single annotated hand. This applied to the majority of the HaGRIDv2 test set (4,831 of 6,000 images) and all images in Open-Closed-1k. The approach described in Algorithm 1 was used to compute a score from the MediaPipe wrist, fingertip, and knuckle landmarks to classify the hand as open or closed.

Because MediaPipe Hands includes an internal hand detection stage, it may fail to produce a prediction for some images. We therefore report two evaluation protocols: a strict setting that treats missing detections as incorrect predictions, and a more generous setting that excludes such cases. Across both datasets, MediaPipe Hands achieves the highest classification accuracy under both protocols, attaining high performance in the strict setting (94.99% on HaGRIDv2 and 84.22% on Open-Closed-1k) and near-perfect accuracy when MediaPipe Hands successfully detects a hand (99.87% on HaGRIDv2 and 99.80% on Open-Closed-1k).

Open/Closed Hand Detection. Two-class open/closed hand detection results are shown in Table 4. On HaGRIDv2, YOLOv8n-Open-Closed-Hand achieves strong performance (mAP@50 of 99.46 and mAP@50–95 of 88.46), with pretrained initialization providing modest but consistent gains over training from scratch. As in the single-class setting, performance

Model	**HaGRIDv2** Accuracy	**Open-Closed-1k** Accuracy
ResNet18	92.37	81.34
ResNet18 (Scratch)	91.92	80.78
ResNet50	91.53	78.92
ResNet50 (Scratch)	90.53	76.04
MediaPipe Hands (Missing = Wrong)	94.99	84.22
MediaPipe Hands (Missing = Ignored)	**99.87**	**99.80**

Table 3: Hand classification performance across open/closed hand datasets with two classes. ResNet18 (Scratch) and ResNet50 (Scratch) refer to training from random initialization rather than from pretrained ResNet18-ImageNet-1k and ResNet50-ImageNet-1k weights, respectively. The ResNet models were evaluated by classifying cropped hand bounding boxes from the Open/Closed Hand test dataset. MediaPipe, which is not intended to be used at extremely close ranges, was instead evaluated on full images from the Open/Closed Hand test dataset containing a single bounding box (4,831 of 6,000 images in HaGRIDv2 and all 1,077 images in Open-Closed-1k). MediaPipe Hands (Missing = Wrong) treats cases without hand detections as incorrect predictions, whereas MediaPipe Hands (Missing = Ignored) excludes such cases from the evaluation.

Model	**HaGRIDv2**		**Open-Closed-1k**	
	mAP@50	mAP@50–95	mAP@50	mAP@50–95
YOLOv8n-Open-Closed-Hand	**99.46**	**88.46**	63.80	35.51
YOLOv8n-Open-Closed-Hand (Scratch)	99.43	87.26	58.87	34.02
MediaPipe Hands	52.86	12.92	47.12	16.59
MediaPipe Hands (1.2x)	79.43	43.41	**79.03**	**41.34**

Table 4: Hand detection performance across open/closed hand datasets with two classes.

decreases on Open-Closed-1k for all methods, reflecting the added difficulty of localizing and classifying hand poses on a left-out, realistic test set. MediaPipe Hands exhibits lower mAP under standard evaluation due to the same bounding box definition mismatch observed in single-class detection. Expanding bounding boxes produced by MediaPipe Hands by a factor of 1.2 improves IoU alignment with the ground-truth annotations, yielding substantial gains on both datasets and the best performance on Open-Closed-1k (mAP@50 of 79.03 and mAP@50–95 of 41.34).

7 Discussion

Through this project, we gained practical insight into the trade-offs involved in deploying computer vision models for real-time interactive systems. Although we explored a range of hand detection and classification approaches, including custom-trained YOLO models and standalone image classifiers, our experiments ultimately demonstrated that MediaPipe Hands was the most suitable choice for our application. While MediaPipe does not achieve the highest detection mAP under standard benchmark evaluations—largely due to differences in bounding box conventions—it consistently delivered the best qualitative performance in gameplay scenarios and the strongest results on the left-out Open-Closed-1k test set for pose classification.

One of the primary challenges faced was acquiring suitable training data for our models. Given the design of our game pipeline, we required datasets with instances of open and closed hand poses in a variety of conditions and viewpoints, with labeled bounding boxes around hands. As few existing datasets are directly designed for our task, existing datasets were repurposed, such as subsets of American Sign Language and general hand gesture datasets, manually reclassifying images to align with our open- and closed-hand classes. In cases where bounding box annotations were missing, we used MediaPipe to automatically generate hand bounding boxes as part of our preprocessing pipeline.

Another major challenge we encountered was translating hand motion into intuitive and reliable game controls. Our initial proposal envisioned directly mapping hand velocity and direction to racket and ball physics. However, the constraints of the existing tennis simulation, combined with the limited range of motion allowed by a webcam-based, stationary setup, made such a direct mapping impractical. Real-world tennis involves large, continuous body movements and fine-grained control over racket orientation, whereas our system operates within a constrained field of view and relies on simplified gestures. Therefore, we revised our interaction design to have two discrete control modes, using closed-hand gestures for lateral player movement and open-hand gestures for swings.

Our work also has several limitations. First, the system assumes a single visible hand and does not explicitly handle occlusions, multiple users, or multiple hand instances (even those that do not belong to the user). Additionally, our evaluation focused on accuracy and our own preliminary qualitative gameplay experience rather than formal user studies, leaving open questions about how learnable, fatiguing, or playable the game is long-term.

Despite these limitations, this project demonstrates that accessible, webcam-based computer vision can support engaging and intuitive game interaction. Our results highlight the practical advantages of using lightweight, landmark-based models such as MediaPipe for real-time interaction, particularly in applications where low latency is more critical than fine-grained pose estimation. Future extensions of this project could explore integrating multiple-hand interaction mechanics, selective recognition of a player's hand when others are visible, or further integration between real-world velocity and direction with game physics to increase realism and expressiveness. More broadly, this project underscores the potential of computer vision–driven interaction to lower barriers to participation in virtual sports, games, and interactive experiences.

References

[1] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[2] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.

[3] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[4] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[5] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[6] Fan Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang, and Matthias Grundmann. Mediapipe hands: On-device real-time hand tracking. *arXiv preprint arXiv:2006.10214*, 2020.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[9] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[10] Supreeth Narasimhaswamy, Zhengwei Wei, Yang Wang, Justin Zhang, and Minh Hoai. Contextual attention for hand detection in the wild. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. `https://openaccess.thecvf.com/content_ICCV_2019/papers/Narasimhaswamy_Contextual_Attention_for_Hand_Detection_in_the_Wild_ICCV_2019_paper.pdf`.

[11] Alexander Kapitanov, Karina Kvanchiani, Alexander Nagaev, Roman Kraynov, and Andrei Makhliarchuk. Hagrid–hand gesture recognition image dataset. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 4572–4581, 2024.

[12] Anton Nuzhdin, Alexander Nagaev, Alexander Sautin, Alexander Kapitanov, and Karina Kvanchiani. Hagridv2: 1m images for static and dynamic hand gesture recognition. *arXiv*, 2024. `https://arxiv.org/abs/2412.01508`.

[13] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision – ECCV 2014*, pages 740–755. Springer International Publishing, 2014. `https://arxiv.org/abs/1405.0312`.

[14] Esha Uboweja, David Tian, Qifei Wang, Yi-Chun Kuo, Joe Zou, Lu Wang, George Sung, and Matthias Grundmann. On-device real-time custom hand gesture recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 4273–4277, October 2023. `https://arxiv.org/abs/2006.10214`.

8 Appendix

8.1 Instructions for Running Project Code

Our modified MinimumTennis game is available as a Windows build of the Unity Project.[8] We are only able to provide a Windows build, because there are some security issues in making a MacOS build from a Windows device.

To build and play the simulation on your device, follow these steps:

1. Unzip the folder, navigate and run MinimumTennis.exe (There may be a pop-up with "Windows protected your PC." To run the simulation, press "More Info" and "Run Anyway".)
2. Run `mediapipe_revised.py` from the code repository in our submission.[9] This interprets webcam motion into readable commands for the simulation.
   (a) Follow the README instructions to set up a virtual environment and download the required packages.
   (b) Note: The MediaPipe pop-up will lag if there is not an active game, as it will try to send messages to the Unity web socket, which is inactive unless a match is running.
3. Select "Normal Mode." Continue with the default settings for character sprites and "Game Settings."
4. The game will begin automatically with the player's automatic serve.
   (a) Note: Because of compatibility issues with the Unity project that we used for the simulation, the "restart" button will not work. If you want to play another match, close and reopen the application to play again.

[8] `https://github.com/ChenJieNi2004/MinimumTennis/releases/tag/v1.0`
[9] `https://github.com/rohanphanse/CPSC5800-Final`

8.2 Game Demo

A demo of our game can be found at this link: `https://youtu.be/DDZb2j8T5_Y`