Tic-Tac-Toe (Group Task) Documentation

Gehad Mohammed Atia
Esraa Salah Ahmed Mohammed
Hoda Sherif Issa Abd El-Aziz
Mohamed Metwaly Soliman
Mohammed Amir Fathy Elhofy

September 16, 2024

Contents

1 Introduction **2**

2 Dataset Collection **2**

3 Model Training **2**
3.1 YOLO Setup 2
3.2 Results and Model Accuracy 3

4 Game Implementation **4**
4.1 Grid Layout and Game Display 4
4.2 Real-Time Gesture Detection 4
4.3 Mapping Gestures to Grid 4
4.4 Game Logic 4
4.5 Visual Feedback 5

5 Testing and Results **5**
5.1 Testing Process 5
5.2 Results 5
5.3 Observations 5

6 Challenges and Solutions **6**
6.1 Limited Dataset Size 6
6.2 Model Selection 6
6.3 Threshold Adjustments 6

7 Conclusion **6**

1 Introduction

The objective of this project is to develop an interactive Tic-Tac-Toe game using computer vision techniques, specifically designed to be played through hand gestures. This project was inspired by the iconic Justice League, adding a playful twist to a classic game by incorporating gesture recognition as the primary mode of interaction.

Using a state-of-the-art object detection model, YOLO (You Only Look Once), players can make their moves on a 3x3 grid by performing specific hand gestures. A gesture places an "X", while places an "O", allowing for a hands-free and engaging gaming experience.

This project integrates game mechanics with real-time video processing, challenging us to design and implement a system capable of detecting hand gestures, translating them into meaningful game actions, and ensuring smooth user interaction. The entire process involves collecting and annotating custom gesture datasets, training the YOLO model for accurate detection, and developing a responsive game interface. Through this project, we demonstrate the use of computer vision to bridge the gap between human gestures and interactive gameplay, providing a unique way to experience the classic Tic-Tac-Toe game.

2 Dataset Collection

To train our YOLO model for detecting the required hand gestures (for "X" and for "O"), we began by collecting a diverse dataset of hand images. The dataset consists of 656 photos, featuring different hand sizes, skin tones, and various backgrounds to ensure robustness and accuracy in real-world scenarios.

For data diversity, we photographed hands from different individuals, including ourselves, our family members, and friends. This approach helped capture a wide range of hand features and environmental conditions, which is crucial for training a model capable of generalizing across varied inputs.

Once the photos were collected, we created a workspace on Roboflow, a tool designed for managing computer vision datasets. Using Roboflow's annotation features, we manually labeled the collected images, marking the specific gestures (and) that correspond to the "X" and "O" game pieces. After completing the annotation process, we exported the dataset in a format suitable for YOLO training by downloading the annotated dataset as a zip file. This annotated dataset was then used for model training.

3 Model Training

To develop the hand gesture recognition model, we used YOLOv8, a state-of-the-art object detection framework. Specifically, we utilized the YOLOv8 medium model (`yolov8m.pt`) due to its balance between speed and accuracy, making it well-suited for real-time gesture detection.

3.1 YOLO Setup

The training was performed using Google Colab with the YOLOv8 framework, which was installed using the `ultralytics` package. The model was mounted on Google Drive

to access the dataset and store training outputs. Below are the key steps we followed:

- **Model Loading**: We loaded the pre-trained YOLOv8 medium model:

```
model = YOLO("yolov8m.pt")
```

- **Dataset**: Our dataset was stored in Google Drive and contained annotated images of two specific hand gestures: for "X" and for "O". We used Roboflow for annotation and exported the dataset in a format compatible with YOLO. The dataset configuration was defined in a `data.yaml` file, which specifies the training and validation image paths and class labels.

- **Training Configuration**: The model was trained for 50 epochs using the following hyperparameters:
  - Epochs: 50
  - Image Size: 640x640 pixels
  - Batch Size: 16
  - Learning Rate (lr0): 0.01
  - Data Augmentation: Enabled to introduce variety in the training data and improve model generalization to new, unseen conditions.

  The training was initiated with:

```
results = model.train(
    data="/content/drive/MyDrive/task 13 new model/Task 13.1.v3i.yolov8/data
    epochs=50,
    imgsz=640,
    batch=16,
    lr0=0.01,
    augment=True
)
```

- **Training Hardware**: The training process was executed on Google Colab's GPU to accelerate computation, which significantly reduced the training time.

3.2 Results and Model Accuracy

After 50 epochs, the model achieved good accuracy in detecting both hand gestures. The training metrics, such as precision, recall, and mean Average Precision (mAP), indicated that the model performed well across both gesture classes. We observed the following metrics:

- Precision: 0.96708
- Recall: 0.931

These metrics demonstrated that the model was capable of accurately recognizing hand gestures in various real-time conditions, making it a suitable choice for the interactive Tic-Tac-Toe game.

4 Game Implementation

The Tic-Tac-Toe game was implemented using OpenCV for real-time video processing and gesture detection with a YOLOv8 model. The game allows players to place their moves ("X" or "O") on a 3x3 grid using specific hand gestures: for "X" and for "O".

4.1 Grid Layout and Game Display

A 3x3 grid is dynamically drawn on each frame of the video stream using OpenCV's drawing functions. The grid is divided into 9 equal boxes, each representing a possible move location. The game continuously displays the score for each player (Player 1 as "X" and Player 2 as "O"), and the number of draws. This information is displayed at the bottom of the screen.

4.2 Real-Time Gesture Detection

A pre-trained YOLOv8 model was loaded using the `ultralytics` library to detect the hand gestures in real-time. The model was trained to recognize two specific gestures: for "X" and for "O".

Each frame captured from the webcam is passed to the YOLO model for inference, detecting hand gestures with specific confidence thresholds:

- Confidence threshold for "X" gesture: 0.5
- Confidence threshold for "O" gesture: 0.7

The detected gesture and its location in the frame are extracted from the model's output using the `detect_gesture()` function, which identifies the hand gesture and calculates the center coordinates of the detected gesture.

4.3 Mapping Gestures to Grid

Once a gesture is detected, the system calculates the corresponding position on the Tic-Tac-Toe grid by mapping the detected gesture's center coordinates to one of the 9 boxes. The `get_grid_position()` function determines the grid cell (row and column) based on the location of the gesture in the frame.

The game updates the grid with the corresponding move ("X" or "O") using the `update_grid()` function, which ensures that moves are only placed in empty cells. After each move, the turn alternates between players, ensuring smooth interaction.

4.4 Game Logic

After each move, the game checks for a winner or a draw using the `check_winner_or_draw()` function. This function checks the rows, columns, and diagonals of the grid to determine if there is a winning combination or if all the cells are filled without a winner, resulting in a draw.

If a player wins or the game ends in a draw, the result is displayed on the screen, and the current score is updated accordingly using the `display_winner_and_score()` function.

4.5 Visual Feedback

Players receive instant visual feedback as their gestures are detected and translated into moves on the grid. When a move is made, the corresponding "X" or "O" symbol is drawn on the grid, providing clear feedback to both players.

After each round, the game pauses for 2 seconds, displaying the winner (or a draw), before resetting the grid for the next game.

5 Testing and Results

To evaluate the performance of our Tic-Tac-Toe game, we conducted several tests by running the Python script and interacting with the game in real-time using hand gestures. The main objective of the testing phase was to assess the accuracy of gesture detection and the overall smoothness of gameplay.

5.1 Testing Process

We tested the game by playing multiple rounds ourselves, performing the (for "X") and (for "O") gestures in front of the webcam. The game was tested in different environments with varying lighting conditions to evaluate the robustness of the YOLO model's gesture detection. Each test session involved moving our hands across different parts of the grid to simulate typical gameplay, ensuring that the gestures were accurately detected and mapped to the correct grid locations.

5.2 Results

The gesture detection system performed well during testing, with the camera accurately identifying both the and gestures. The YOLO model was able to correctly distinguish between the two gestures and translate them into the respective moves ("X" or "O") on the Tic-Tac-Toe grid.

The system correctly recognized the grid location where the gesture was performed, placing the move in the appropriate box. This allowed for seamless gameplay, where players could focus on the game without experiencing significant delays or misdetections.

Overall accuracy: In most cases, the gestures were recognized with high accuracy, and the game responded promptly to player actions.

5.3 Observations

- **Lighting conditions**: The game performed best in environments with sufficient lighting, where the model could clearly detect the hand gestures. In low-light conditions, we observed a slight reduction in detection accuracy, but the game was still playable.
- **Game responsiveness**: The game interface was responsive, with minimal lag between performing a gesture and the move appearing on the grid.

6 Challenges and Solutions

Throughout the development of the Tic-Tac-Toe game, we encountered several challenges that affected the accuracy and performance of the gesture detection system. Below are the key issues faced and the solutions we implemented to overcome them:

6.1 Limited Dataset Size

- **Challenge**: Initially, our dataset consisted of only 235 images, which resulted in poor model performance. The limited variety in hand gestures, backgrounds, and lighting conditions made it difficult for the model to generalize well and accurately detect gestures.

- **Solution**: To address this issue, we significantly expanded our dataset to 656 images, capturing a more diverse range of hand sizes, skin tones, and environments. This improvement in data diversity helped enhance the model's ability to detect gestures across different scenarios and led to a noticeable increase in detection accuracy.

6.2 Model Selection

- **Challenge**: At the beginning, we experimented with different YOLO models, including `yolov8l.pt` (YOLOv8 large) and `yolov8n.pt` (YOLOv8 nano). However, neither model provided satisfactory results in terms of detection accuracy and speed. The large model was too slow for real-time performance, while the nano model lacked sufficient precision.

- **Solution**: We switched to the YOLOv8 medium model (`yolov8m.pt`), which offered a better balance between accuracy and speed, making it more suitable for real-time gesture detection. This change significantly improved the overall performance of the system.

6.3 Threshold Adjustments

- **Challenge**: Initially, the detection thresholds were not optimized, leading to incorrect gesture recognition or missed detections during gameplay.

- **Solution**: We fine-tuned the confidence thresholds for both ("X") and ("O") gestures. By adjusting the confidence threshold for ("X") to 0.5 and ("O") to 0.7, we improved the model's ability to accurately distinguish between the two gestures. This adjustment further enhanced the detection accuracy and reduced misclassifications during the game.

7 Conclusion

In this project, we successfully developed an interactive Tic-Tac-Toe game powered by computer vision, allowing players to make their moves using hand gestures detected in real-time through YOLO object detection. By integrating gesture recognition with

traditional game mechanics, we provided a unique and engaging experience, transforming a classic game into an innovative, hands-free interaction.

The process involved various challenges, including expanding our dataset, selecting the appropriate model, and fine-tuning detection thresholds. Through iteration and improvements, we were able to overcome these obstacles, resulting in a robust system capable of accurately detecting gestures and ensuring seamless gameplay. The use of the YOLOv8 medium model, combined with a more diverse dataset and threshold adjustments, significantly enhanced the accuracy and reliability of the gesture detection.

This project not only demonstrated our ability to apply computer vision techniques in a practical setting but also emphasized the importance of continuous optimization and testing to achieve desired results. The final product met the objectives of the task, providing an entertaining and functional game with real-time gesture detection.

Looking forward, there are opportunities to further improve the system by enhancing its performance in low-light environments and potentially expanding the range of gestures recognized. These advancements could open the door to even more interactive and immersive gaming experiences.