Configuration Manual

MSc Research Project
Cyber Security

Kartikey Sharma
Student ID: 22128824

School of Computing
National College of Ireland

Supervisor: Jawad Salahuddin

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Kartikey Sharma
Student ID:	22128824
Programme:	MSc Cyber Security **Year:** 2023
Module:	MSc Research Project
Lecturer:	Jawad Salahuddin
Submission Due Date:	14/12/23
Project Title:	A novel approach for privacy preserving cheat detection in E-Sports using cloud-based computer vision techniques
Word Count:	802…… **Page Count:** 8…….……

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: *Kartikey Sharma*

Date: 14/12/23

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Kartikey Sharma
Student ID: 22128824

1 Introduction

In the research project we explored the idea of using a computer vision model specifically trained to detect cheating techniques in competitive video games while utilizing minimum user data and resources. The AI model was then deployed on a cloud service to test its viability and increase in performance relative to local deployment. This configuration manual serves as a step-by-step guide to recreate all the environment both locally, and on the cloud, that was used to carry out the research.

2 Configurations

Below are the software artefacts used to carry out the research followed by the minimum required hardware to be able to train and run inference on the model locally.

Table 1: Software requirements

Software	**Version**
Python	3.11
Ultralytics package	8.0.202
YOLO model	V8n
AssaultCube game	1.3.0.2
PyTorch	0.16.0+cu118
OpenCV Python	4.8.1.78
Roboflow	1.1.9
MinGW	0.6.0-beta-20130904-1
OBS Studio	30.0.2

Table 2: Hardware requirements

Hardware (Local deployment)	**Minimum requirement**
Operating System	Windows 10 (64 bit)
CPU	Intel Core i5 2500k or AMD FX-6300
Memory	4 GB of RAM
Graphics	GTX 1060
Storage	10 GB of free space on the hard disk

2.1 Dependencies and Compilers

The command shown is a one-click solution that calls the pip utility to install the required python modules (Intel, no date; Meta AI, no date; *Ultralytics | Revolutionizing the World of Vision AI*, no date; *ultralytics/ultralytics: NEW - YOLOv8 🚀 in PyTorch > ONNX > OpenVINO > CoreML > TFLite*, no date).

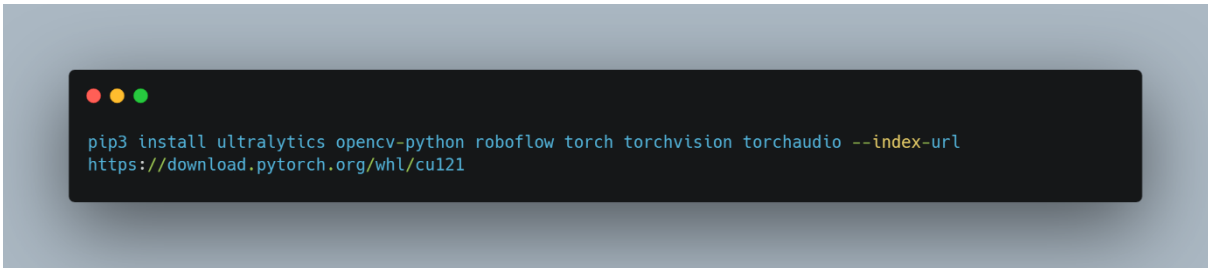

Figure 1: Python pip module to install dependencies

Further we must install MinGW[1] which is a C/C++ compiler for Microsoft Windows. This is needed to compile the ESP cheat written in C++ (*MinGW-w64*, 2007).

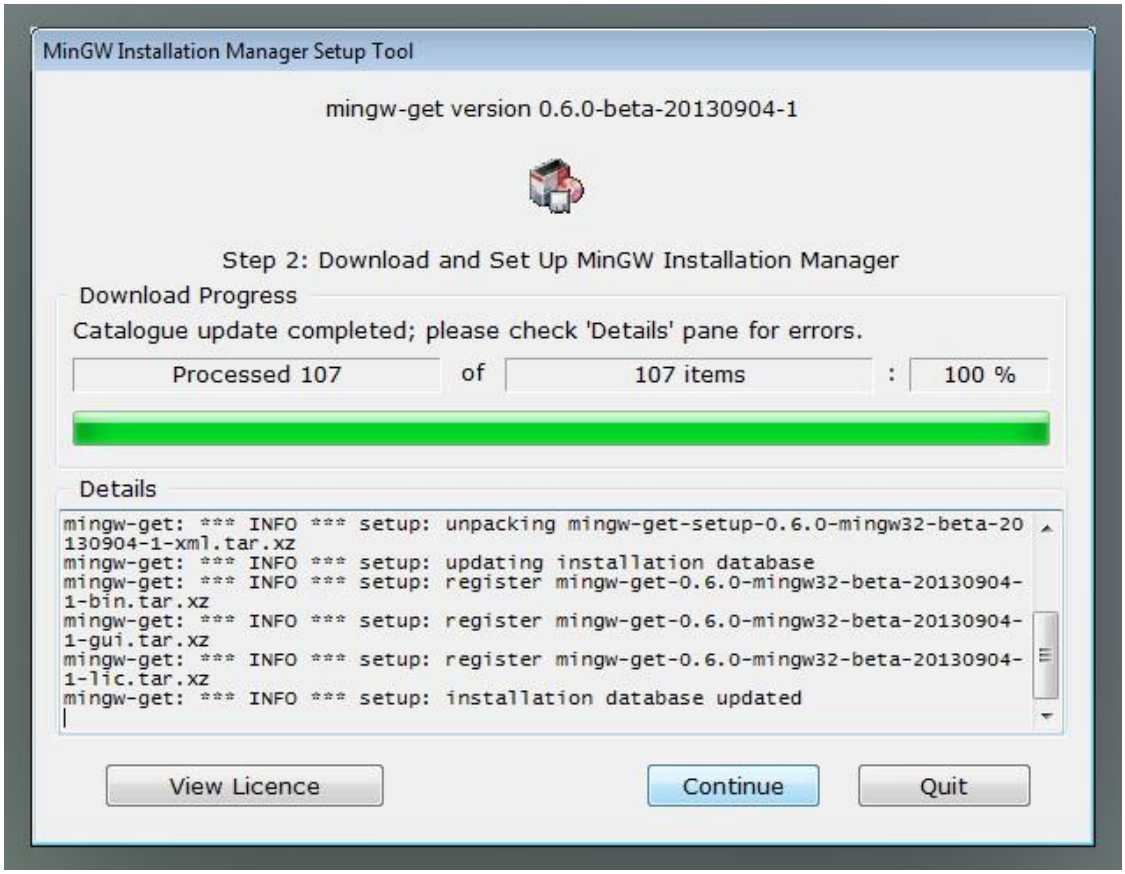

Figure 2: MinGW installation

[1] MinGW Sourceforge: https://sourceforge.net/projects/mingw/

2.2 AssaultCube Setup

Download and install AssaultCube from https://assault.cubers.net/ as this is the open-source game used in the research (Rabid Viper Productions, 2006).

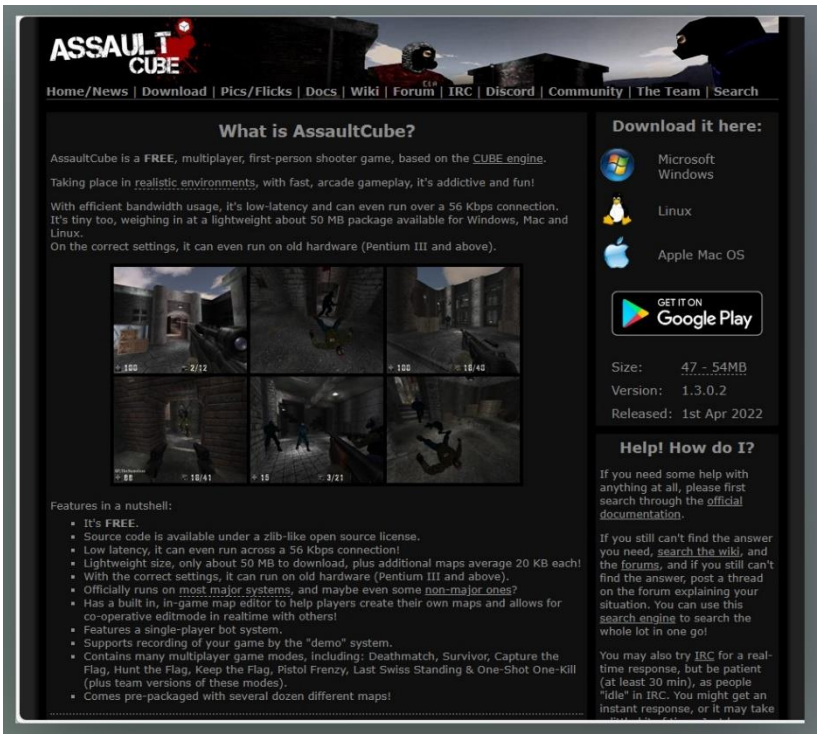

Figure 3: AssaultCube official website

2.3 Compiling the ESP Cheat

Using a terminal, after successful installation of MinGW, navigate to the root directory of the ESP cheat and execute the following command which will output an executable named ESP.exe to the present working directory.

Figure 4: Compiling the C++ code

2.4 OBS Studio Setup

OBS Studio is an open-source screen recording and streaming solution that was used to record AssaultCube in two different scenarios where the ESP cheat was on and off. This tool proved

to be a very efficient tool to generate the dataset required to train the AI model (*Open Broadcaster Software | OBS*, no date).

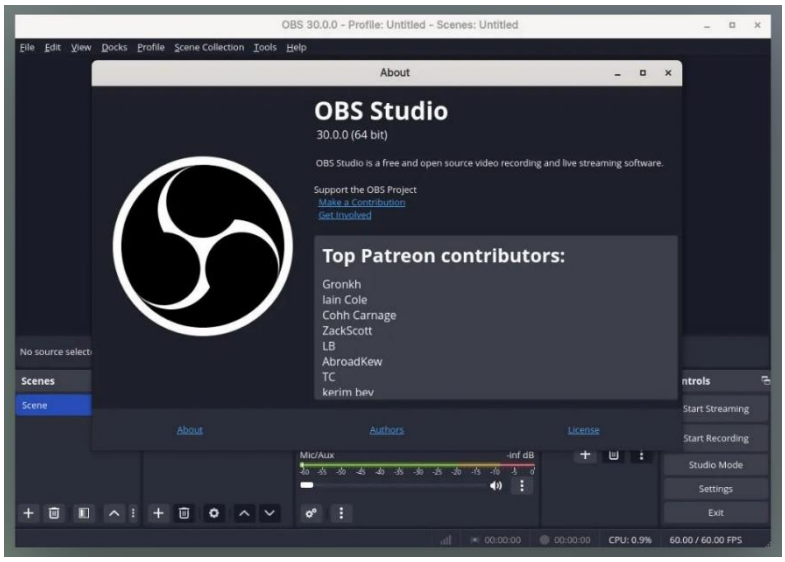

Figure 5: OBS Studio splash screen

2.5 Roboflow Setup

Roboflow was used to organize and optimize the dataset for training. After extracting frames from the recorded AssaultCube sessions in OBS Studio, we use roboflow to perform annotation and clearly define our classes for detection. The features available for free proved sufficient for carrying out this research (*Roboflow: Give your software the power to see objects in images and video*, no date).

After signing up for Roboflow, create a new project.

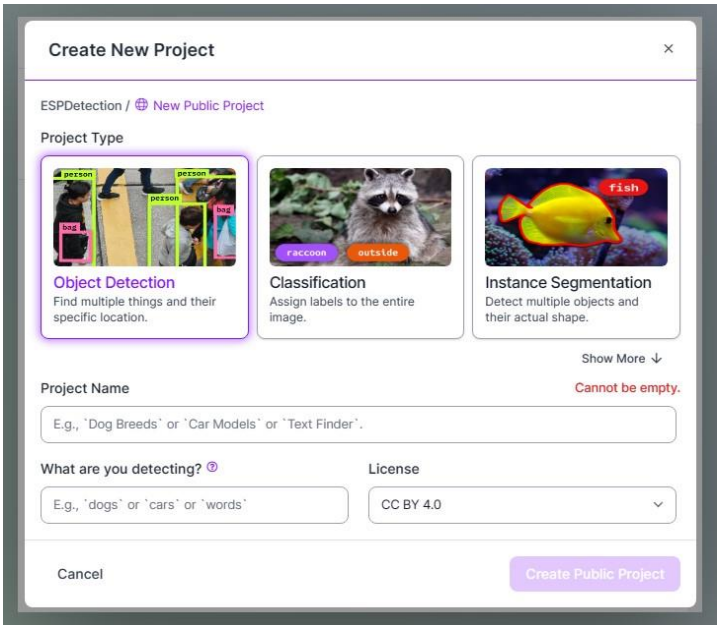

Figure 6: Create new project

Upload the collected images that must be used for training the model.

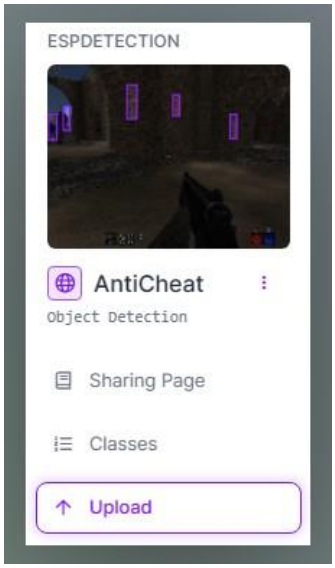

Figure 7: Upload dataset

Each frame must be annotated with the desired classifications.

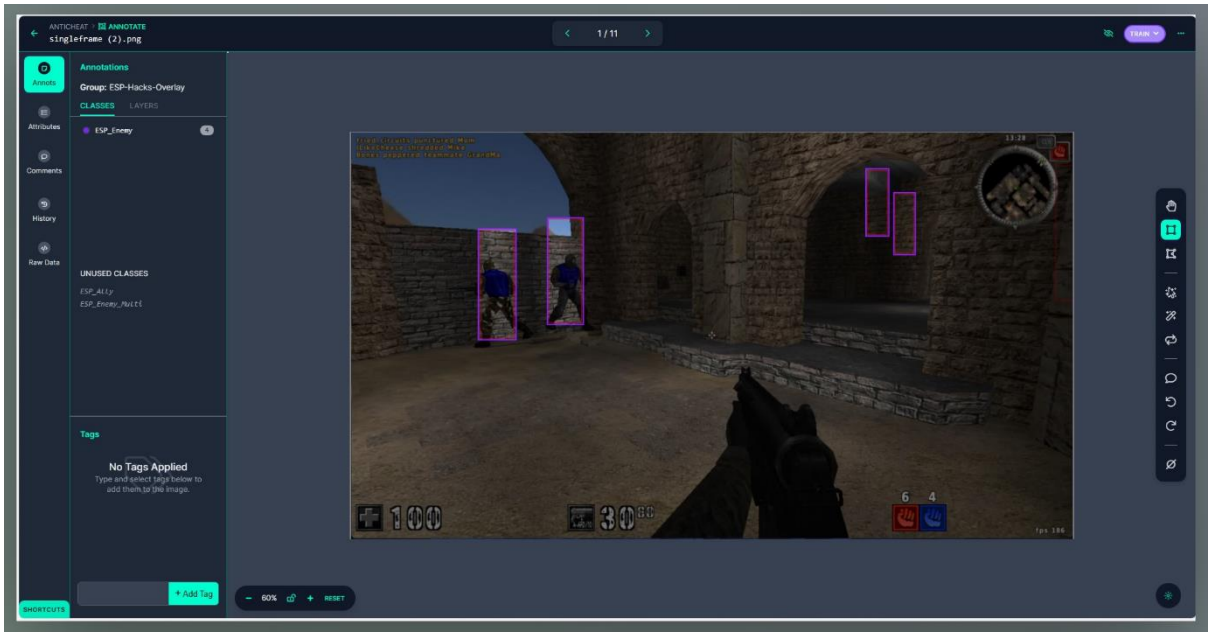

Figure 8: Frame annotation

Generate your personal API key by navigating from Account to Settings and then workspaces. This is essential to download the optimized dataset directly through the python script written for this research and initiate training of the model.

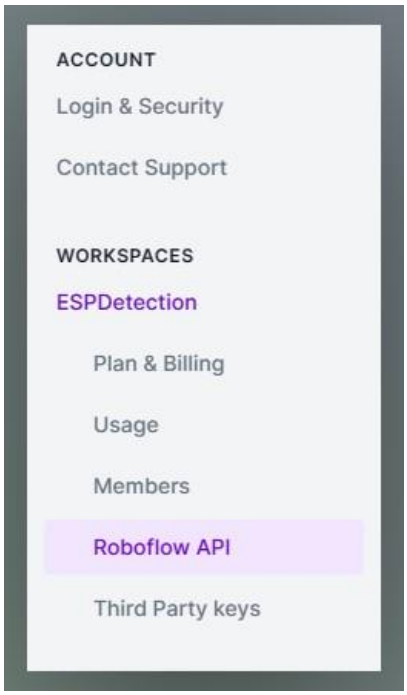

Figure 9: Generate API key

3 Model Training and Detection

Proceed to execute the "train AC.py" file available in the project artefact to download and initiate training of the model.

```
from roboflow import Roboflow
from ultralytics import YOLO
import torch

print(torch.cuda.is_available())
device = 'cuda' if torch.cuda.is_available() else 'cpu'
print(f'Using device: {device}')

rf = Roboflow(api_key="********************")
project = rf.workspace("espdetection-fndjp").project("anticheat-jcyqg")
dataset = project.version(1).download("yolov8")

model = YOLO('yolov8n.pt')

results = model.train(data='AntiCheat-1/data.yaml', epochs=100, imgsz=640)
```

Figure 10: Sample code for dataset download and training locally

This code can also be used on a cloud platform supporting python runtime or jupyter notebooks.

3.1 Launching AssaultCube and Enabling Cheat

With AssaultCube successfully installed and our ESP cheat compiled into an executable, we run the game and start a singleplayer match with bots.
To do that, we select “Singleplayer” when the game launches, and select “Bot Team Deathmatch” followed by selection of the difficulty, and finally how many bots to add to the game.

Figure 11: AssaultCube match launched with bots

With the game launched, run the ESP.exe compiled in the section 2.3 to enable the cheat and record the screen playing the game.

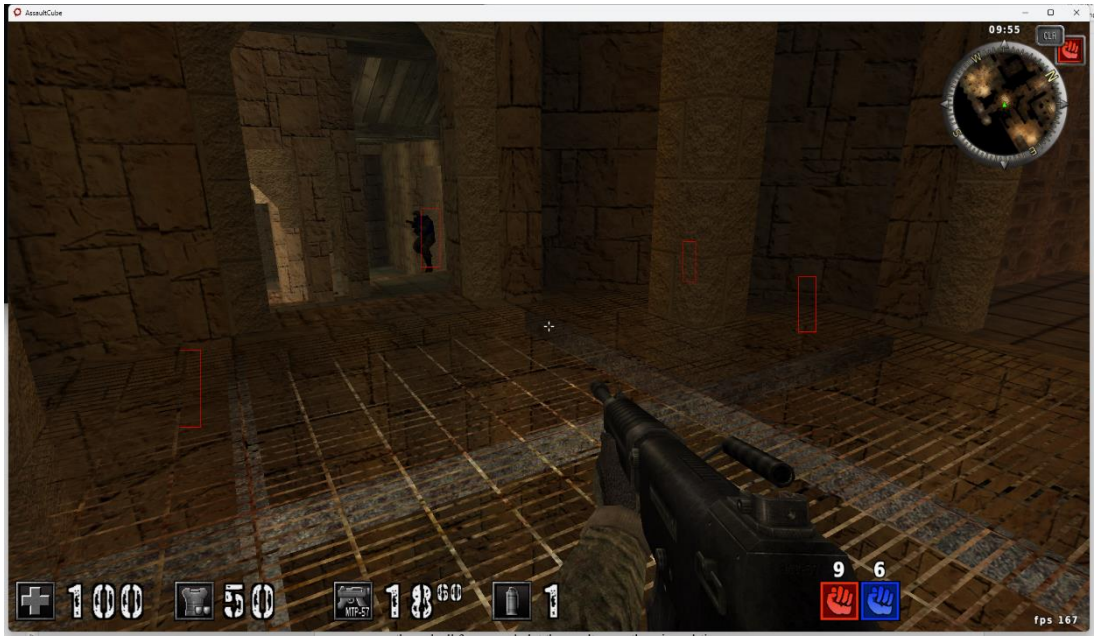

Figure 12: AssaultCube with ESP cheat enabled

3.2 Detection Testing

Finally, use the trained model checkpoint over the recorded AssaultCube session with ESP cheat turned on to test its output.

Simply execute the “detectVideo.py” python script included in the project artefact to iterate through all frames and plot the results over them in real-time. To run the script on a cloud platform instead, we run the “detectVideo_Cloud.py” script instead which is optimized for execution on a cloud service.

Figure 13: Detection testing

```
0: 384x640 2 ESP_Enemys, 7.2ms
Speed: 1.0ms preprocess, 7.2ms inference, 3.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 ESP_Enemys, 7.0ms
Speed: 2.1ms preprocess, 7.0ms inference, 2.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 ESP_Enemys, 7.0ms
Speed: 3.0ms preprocess, 7.0ms inference, 1.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 ESP_Enemys, 7.9ms
Speed: 1.1ms preprocess, 7.9ms inference, 1.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 2 ESP_Enemys, 8.1ms
Speed: 1.0ms preprocess, 8.1ms inference, 2.0ms postprocess per image at shape (1, 3, 384, 640)

0: 384x640 4 ESP_Enemys, 7.0ms
Speed: 2.0ms preprocess, 7.0ms inference, 2.0ms postprocess per image at shape (1, 3, 384, 640)
```

Figure 14: Console output of the detection script

References

MinGW-w64 (2007). Available at: https://www.mingw-w64.org/ (Accessed: December 13, 2023).

Open Broadcaster Software | OBS (no date). Available at: https://obsproject.com/ (Accessed: December 13, 2023).

Intel (no date) *OpenCV - Open Computer Vision Library, 2000*. Available at: https://opencv.org/ (Accessed: December 13, 2023).

Meta AI (no date) *PyTorch, 2016*. Available at: https://pytorch.org/ (Accessed: November 13, 2023).

Rabid Viper Productions (2006) *AssaultCube*. Available at: https://assault.cubers.net/ (Accessed: November 11, 2023).

Roboflow: Give your software the power to see objects in images and video (no date). Available at: https://roboflow.com/ (Accessed: November 12, 2023).

Ultralytics | Revolutionizing the World of Vision AI (no date). Available at: https://www.ultralytics.com/ (Accessed: November 17, 2023).

ultralytics/ultralytics: NEW - YOLOv8 🚀 in PyTorch > ONNX > OpenVINO > CoreML > TFLite (no date). Available at: https://github.com/ultralytics/ultralytics (Accessed: November 17, 2023).