CYSF - A Practical Weapon Detection System for Enhanced Security Using Artificial Intelligence - By Alveena Ashiq - Grade 8 - LOGBOOK

Logbook at a glance

Dec 23, 2024	I filled it out in the 'Basic Project Info' section on the CYSF platform.* My project is innovation-based. *My school CYSF coordinator invited me onto the platform.
Dec 24, 2024 - Dec 27, 2024	Did online research on gun violence and severity. Did some brainstorming to come up with an idea for proposing a solution to mitigate the issue.
Dec 28, 2024 - Dec 31, 2024	Did online research on machine learning and learned about supervised learning. Tried to find an open source dataset for training a custom model to detect weapons. Learned about YOLO and the library called Ultratlytics that supports YOLO for model training and inference.
Jan 1, 2025 - Jan 4, 2025	Started coding Python script for model training with the downloaded dataset from various sources. Tried different epoch settings and different model profiles and tested on both my laptop and the Raspberry PI to select the most suitable models to run on. The open dataset from Kaggle was chosen finally as the model trained with this dataset performed the best.
Jan 5, 2025 - Jan 8, 2025	Started coding an application to run the model for weapon detection. Ordered Raspberry PI board and the Raspberry PI camera from Amazon. Also ordered toy guns from halloweencostumes.ca and a toy knife from Amazon for testing and demonstration.
Jan 9, 2025 - Jan 12, 2025	Registered to Twilio for Email and SMS notification. Included the Email and SMS notification features in my application. Received Raspberry PI from Amazon. Prepared the Raspberry PI by installing the Raspberry PI camera and installed the required Python libraries such as OpenCV, Ultralytics.
Jan 13, 2025 - Jan 17, 2025	I had a lot of school assignments this week; therefore, I took a break from working on CYSF.

Jan 18, 2025 - Jan 21, 2025	I had a lot of school assignments this week; therefore, I took a break from working on CYSF.
Jan 22, 2025 - Jan 25, 2025	Modified the Python application written for laptop to run on Raspberry PI without a Graphical User Interface. Performed thorough testing of both the applications (on laptop and on Raspberry PI).
Jan 26, 2025 - Jan 29, 2025	Started working on the poster. Completed the 'Problem' section of my project.
Jan 30, 2025 - Feb 2, 2025	Completed the 'Method' section. Included some snapshots of my developed application, training dataset samples and the Raspberry PI prototype. The prototype prepared for the school science fair is temporary. If I get the opportunity, I am working on building an enclosure for my final demonstration at the CYSF science fair.
Feb 3, 2025 - Feb 4, 2025	Completed the Analysis, Conclusion, Citations and Acknowledgement sections. Worked on formatting the poster materials for printing.
Feb 5, 2025 - Feb 6, 2025	Printed poster pages from Staples and finished glueing all the sheets of paper onto my trifold.
February 7, 2025	Attended my school science fair 2025.
February 7, 2025	Got selected to attend CYSF 2025
Feb 8, 2025 - March 7, 2025	Took a break from CYSF
March 7, 2025 - March 14, 2025	Created a Plexiglass enclosure for my YOLOv8 Nano model, uploaded images onto CYSF platform
March 15, 2025 - March 18, 2025	Created slide show presentation for project and used it to create a 10 minute long video on my project. Uploaded video, slideshow and logbook onto platform. At this point, all CYSF platform elements complete.
March 19, 2025 - April 9, 2025	Presentation rehearsals, practiced speech

April 10, 2025 - April 12, 2025	Attended CYSF 2025

A detailed log of research done

Dec 23, 2023	I filled it out in the ‘Basic Project Info’ section on the CYSF platform.* My project is research/study based. *My school CYSF coordinator invited me onto the platform.

I filled these on the CYSF platform:

Name of Project: A Practical Weapon Detection System for Enhanced Security Using Artificial Intelligence

Grade: 8

Project type: Innovation

Language: English

Project Category: Mathematics and Computer Science

Project Topics: Computer Science, Engineering, Technology

Brief Description: A weapons detection system

Jan 26, 2025 - Jan 29, 2025	Started working on the poster. Completed the ‘Problem’ section of my project.

PROBLEM

Gun violence is a daily scourge that jeopardizes our most fundamental right, the right to life. Every day, more than 600 people die as a result of firearm violence, which is fueled in part by easy access to firearms, whether legal or illegal [1]. Other root causes of gun violence include but are not limited to [2]:

- Income inequality
- Poverty
- Underfunded public housing
- Under-resourced public services
- Underperforming schools
- Lack of opportunity and perceptions of hopelessness

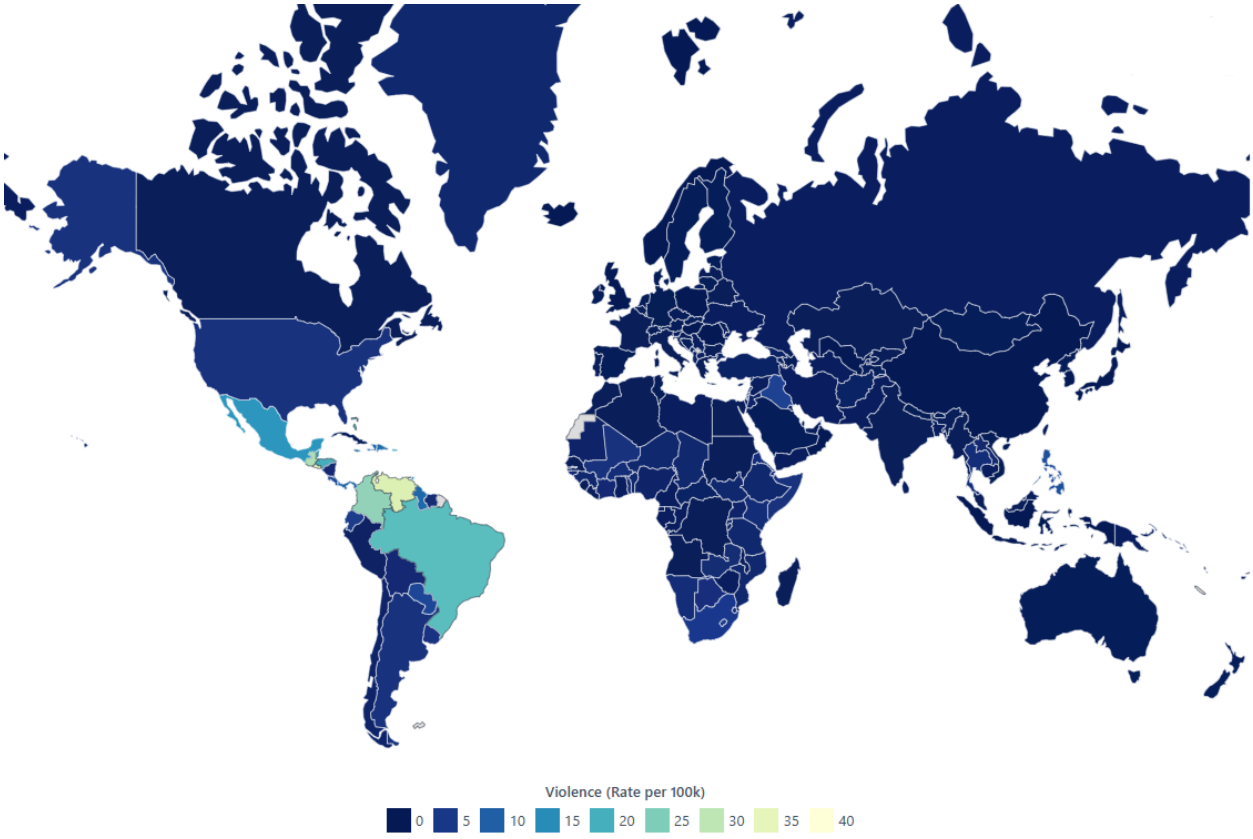


Figure 1: Over 250,000 people died from firearms in 2019, with the majority of deaths being homicides, especially in countries like Brazil and the United States. [3]

As we are in North America, this project primarily focuses on Canada and the United States. Both countries are currently working to prevent gun violence. For example, in Canada, in March 2022, the Government announced a new federal investment of $250 million through the Building Safer Communities Fund (BSCF). The fund will help municipalities and Indigenous communities prevent gun and gang violence by tackling its root causes [4].

However, even with these governmental precaution measures in place, North America is still experiencing gun violence daily. Therefore, the question is:

How can we make our cities more secure against gun and weapon violence?

Jan 30, 2025 - Feb 2, 2025	Completed the ‘Method’ section. Included some snapshots of my developed application, training dataset samples and the Raspberry PI prototype. The prototype prepared for the school science fair is temporary. I am working on building an enclosure for my final demonstration at the CYSF science fair if I get the opportunity.

METHOD

This project focuses on developing a machine learning [5] algorithm and creating an app that can accurately detect weapons using cameras and transmit notifications in real-time via email and SMS for immediate action against potential violence. Early detection of firearms and other weapons can prevent violence from happening and save hundreds of lives.

OBJECT DETECTION

The YOLO (You Only Look Once) model is used in this project to detect weapons. YOLO is a popular object detection [6] model known for its speed and accuracy. It was first introduced by Joseph Redmon in 2016 [7]. The foundation of YOLO is a neural network [8] that falls under deep learning [9], a subset of machine learning. A key step in developing a machine learning solution is training. Typically, pre-trained models are available as open source that are trained to detect common objects, such as people, cars, etc., for quickly deploying a machine-learning solution. However, no pre-trained model was found freely available to satisfy the requirements of this project after a comprehensive search over the internet. Therefore, this project's first step was to train the YOLO model to detect weapons. I used a dataset containing various types of guns and knives to train the model to detect weapons for demonstration.

DEVELOPMENT ENVIRONMENT AND THE PLATFORM

I chose Python as the programming language and used Visual Studio Code for development. As one of the most popular programming languages, Python offers several benefits, including [10]:

1. Python runs on major platforms like Windows, macOS, Linux, Raspbian, and more.
2. It provides extensive library support for data analysis and machine learning, including libraries such as NumPy, OpenCV, PyTorch, and others.

3. Python is ideal for rapidly validating a proof of concept or building a prototype.

Visual Studio Code offers a user-friendly interface for editing, debugging, and running the developed code [11]. In this work, I created and demonstrated I prototype on two different platforms:

1. A Windows-based workstation: This setup demonstrates the suitability of I system for a central monitoring station, where it can be connected to multiple IP security cameras to detect weapons.
2. An integrated Raspberry Pi-based [12] camera system: Equipped with weapon detection analytics, this edge device provides a cost-effective solution that does not require a central monitoring station for operation.

TRAINING: Supervised learning [13] is the learning approach suitable for this project. It is akin to classroom teaching, where the teacher shows labelled examples—such as apples and bananas—and the students learn to recognize these objects when they encounter them later. Other types of learning include unsupervised learning [14] and reinforcement learning [15].

In my approach, I used images of knives and guns, labelling them as weapons. Instead of capturing or collecting my own dataset, which is time-consuming and labour-intensive, I leveraged open-source images. These datasets are freely available on the internet from several platforms, including Kaggle [16], Roboflow [17], and various GitHub repositories. After careful consideration, I chose a dataset from Kaggle to train my YOLOv8 model. The comprehensive dataset contains 8,149 annotated images of knives and various types of guns [18].


武器大讲堂
是供坦克和车辆乘员使用的短枪管型


武器大讲堂
美国M249机枪（FN Mini）
Ultimax 100轻机枪


武器大讲堂
却要求设计成

Figure 2: Some labelled data of guns and knives

YOLOv8 MODELS

While YOLOv8 is compatible with multiple frameworks, it is primarily supported by the Ultralytics library [19], which simplifies implementation. Therefore, this work has used the Ultralytics Python library for model training and development.

YOLOv8 offers five model profiles, ranging from the lightweight 'Nano' (YOLOv8n) to the high-capacity 'Extra Large' (YOLOv8x) profile [20]. The choice of a suitable profile depends on the application requirements, available resources, and computational power. The key tradeoff among these profiles is accuracy versus speed—smaller models provide faster inference but lower accuracy. In contrast, larger models offer higher accuracy at the cost of slower inference speed and increased computational demands.

Given this tradeoff, I selected the 'Large' profile (YOLOv8l) for the Windows-based workstation application, as these systems typically have sufficient computational power to handle it. However, due to the minimal computational resources of the Raspberry Pi, I used the 'Nano' profile (YOLOv8n) for developing the integrated weapon detection system.

The pre-trained Nano [21] and Large [22] models do not include knife or gun detection by default. Therefore, I trained the models using a labelled dataset downloaded from Kaggle [18]. The figure below shows the code snippet used for training.

train.py > ...

```python
from pathlib import Path
from ultralytics import YOLO

# Load a COCO-pretrained YOLOv8l model
model = YOLO('./model/yolov8l.pt')

# Define the absolute path for the training dataset configuration file 'data.yaml'
data_path = str(Path('./Weapon.detector.v1i.yolov8/data.yaml').resolve())

# Train the model on the custom dataset
# parameters:
#    data: Path to dataset configuration file
#    epochs: Number of training epochs/iterations
#    workers: Number of worker threads for data loading
#    imgsz: Input image size
#    device: Device to run training on. 0 selects GPU
results = model.train(data=data_path, batch=8, epochs=100, workers=0, imgsz=640, device=0)
```

Figure 3: Python script for training YOLOv8m model with weapon detector dataset

The Ultralytics train() function for model training accepts several parameters. One key parameter is 'data,' which specifies the path to the configuration file (data.yml). This configuration file contains the locations of the training and validation images, the number of classes in the dataset and their names.

In my case, I have a single class called "weapon." However, this class encompasses a variety of objects, such as knives, guns, and rifles.

The "epochs" parameter defines the number of training iterations. I set this value to 100.

Figure 4 shows a code snippet from the developed application. The trained model is loaded to run predictions on images captured by the webcam.

```python

class VideoThread(QThread):
    change_pixmap_signal = pyqtSignal(QVariant, arguments=['data'])

    def __init__(self, channel):
        super().__init__()
        self.channel = channel
        self.model = YOLO('../YOLOv8l_WeaponModel.pt')

    def run(self):
        # capture from web cam1')
        vidcap = cv2.VideoCapture(self.channel)

        success, image_BGR = vidcap.read()

        while (success):
            now = datetime.now()
            success, image_BGR = vidcap.read()

            results = self.model.predict(image_BGR, conf = 0.7)

            # Visualize the results on the frame
            annotated_frame = results[0].plot()

            aDict = {"image": annotated_frame, "result": results[0], "datetime": now}

            self.change_pixmap_signal.emit(aDict)

        # shut down capture system
        vidcap.release()

```

Figure 4: A code snippet from the developed application

A snapshot of the developed application is shown in Figure 5. The application interface is designed using the Qt library for Python. Currently, the application supports only a single camera, but the concept can be easily scaled to accommodate multiple cameras. The interface includes a live view, a snapshot view of alarms, an alarm list, and an input configuration window where users can enter an email address and phone number to receive real-time notifications when a weapon is detected via email or SMS.

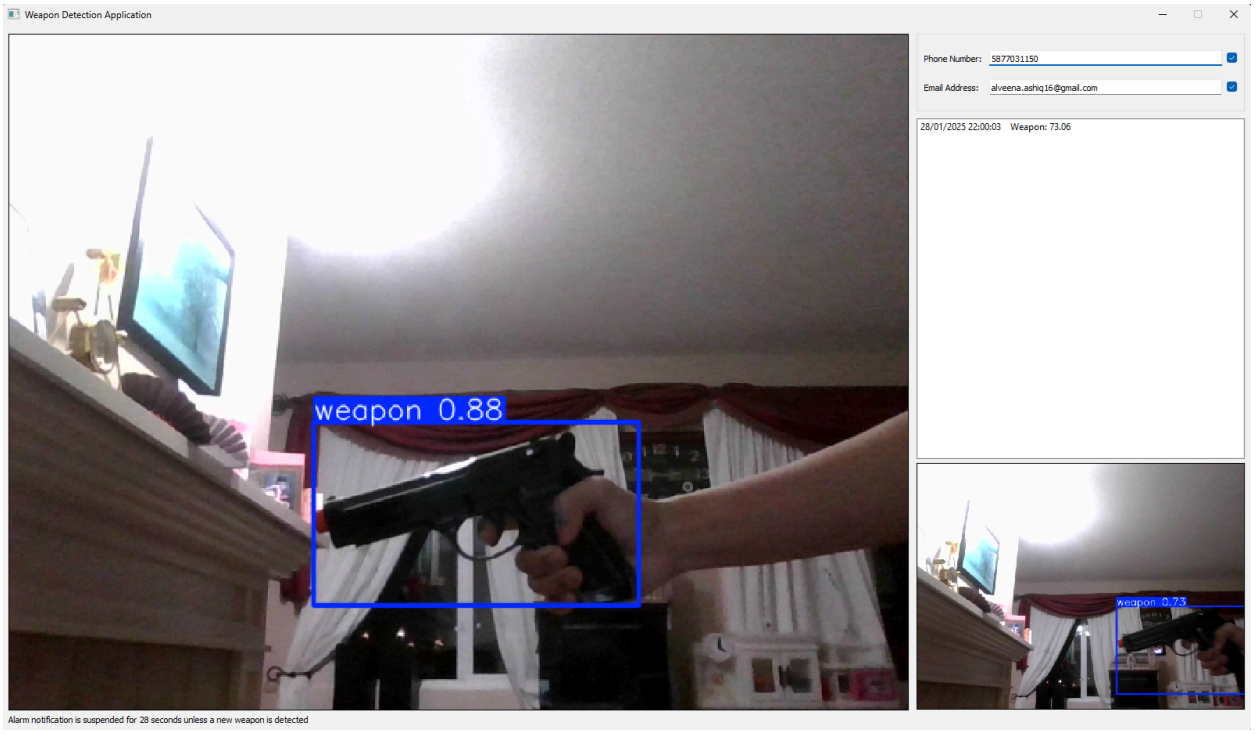
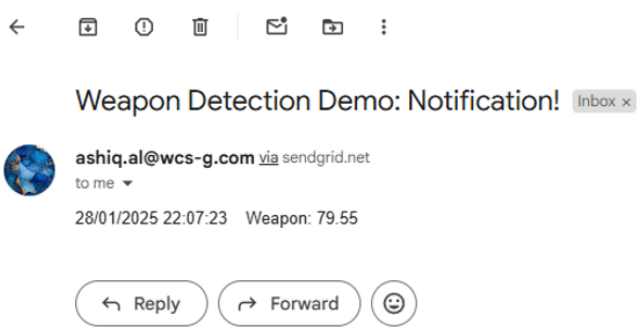
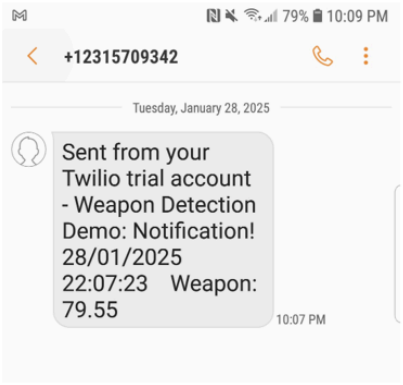


Figure 5: Weapon detection application interface

Figure 6 shows samples of email and SMS notifications. I used Twilio for these notifications, a platform that allows developers to create applications to support messaging over various media, such as email and SMS.



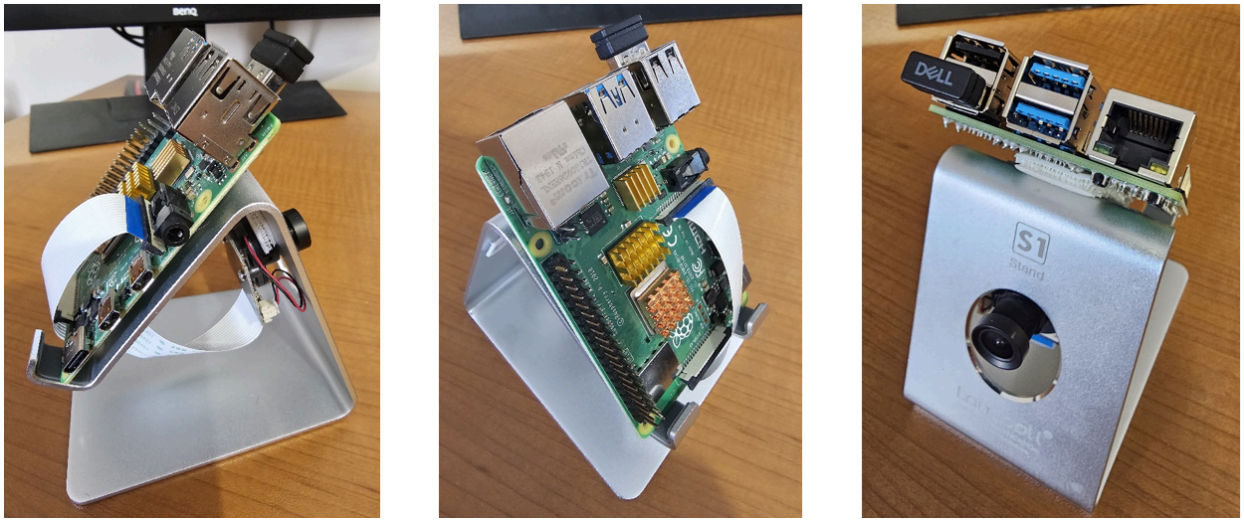
Email notification



SMS notification

Figure 6: Real-time weapon detection notification via Email and SMS

The developed program can also run on an edge device, such as a Raspberry Pi with an integrated camera, for standalone applications. Due to the limited computational power of the Raspberry Pi, I used the YOLOv8 'nano' model to train on the weapon dataset. Figure 7 show the Raspberry Pi 4 hardware board with a camera. This hardware can be enclosed in a box to create a cost-effective integrated camera device, which can function as a standalone edge device for real-time weapon detection.



Figures 7: Raspberry PI camera prototype with integrated weapon detection analytics

The software on Raspberry PI does not use a Graphical User Interface (GUI) to run as it is an edge device and will never be connected to a monitor for practical application. This approach will also help saving some CPU load so that the machine learning analytic's performance on this limited power device is not affected. The cost breakdown for the Raspberry Pi-based smart weapon detection camera is provided below:

Item#	Description	Cost
1	Raspberry PI 4 board	$95.00
2	Raspberry PI camera	$20.00
3	Plexiglass sheet for enclosure	$10.00
4	Mounting hardware	$5.00
	Total	**$130.00**

Table 1: Approximate cost breakdown for a Raspberry 4 PI based integrated camera for weapon detection

Feb 3, 2025 - Feb 4, 2025	Completed the Analysis, Conclusion, Citations and Acknowledgement sections. Worked on formatting the poster materials for printing.

PERFORMANCE ANALYSIS

PERFORMANCE ANALYSIS

Training for both "Large" and "Nano" YOLOv8 models was executed for 100 epochs with a batch size of 8. The figures below show the performance results achieved with both models

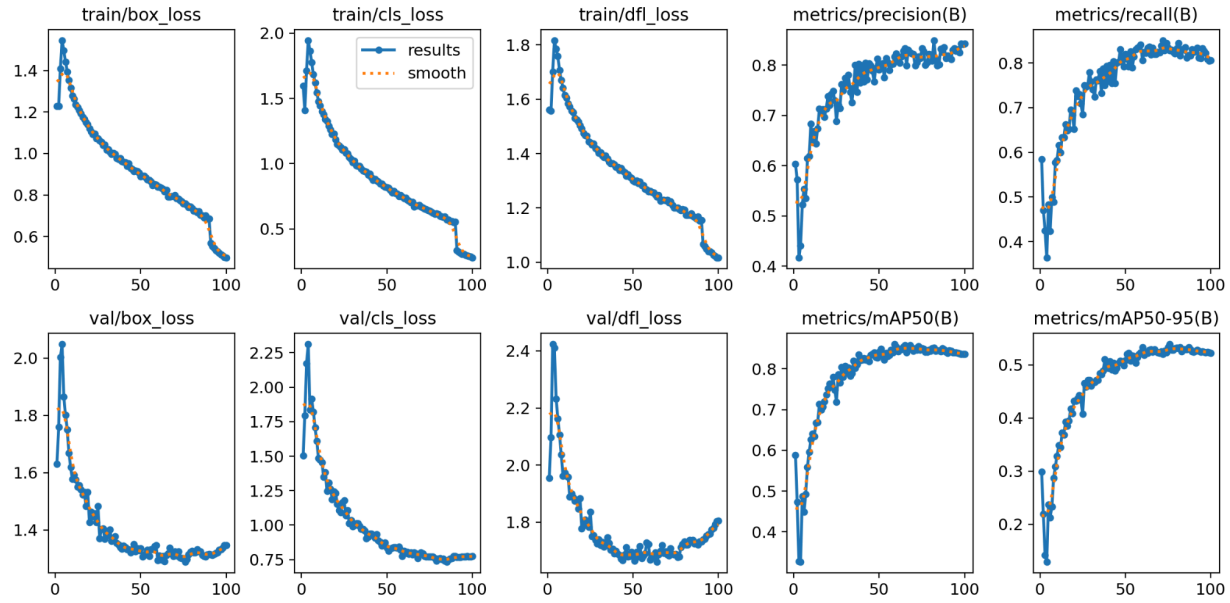


Figure 8: YOLOv8 Lange model training results from over 100 epochs using the weapon dataset

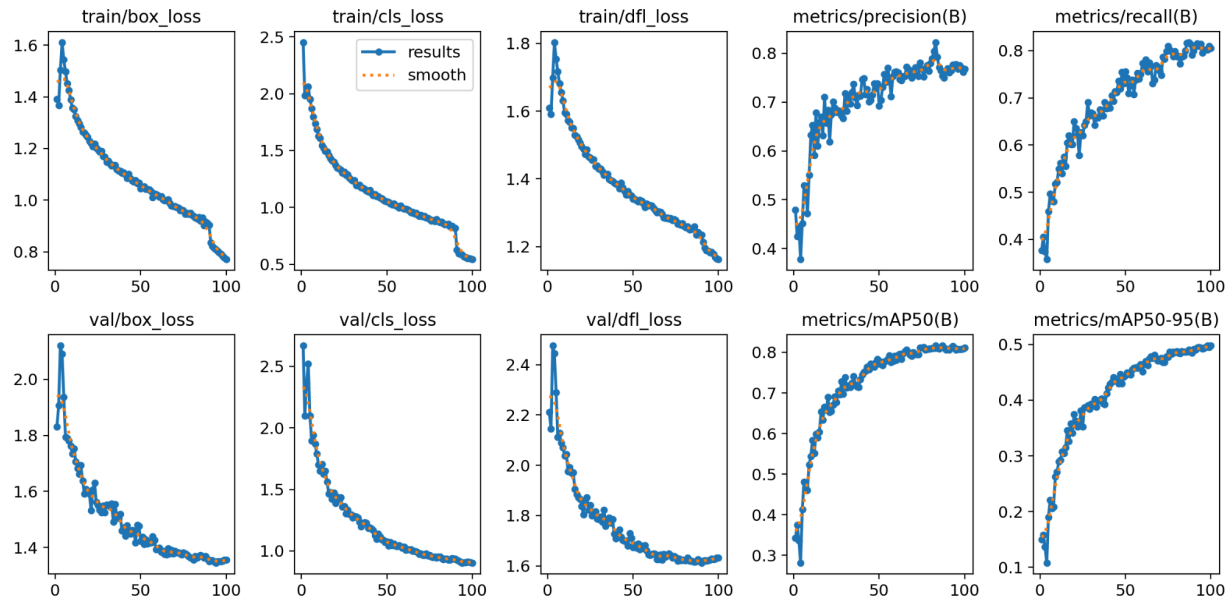


Figure 9: YOLOv8 Nano model training results from over 100 epochs using the weapon dataset

The figures illustrate different types of losses: "box" loss (which measures how well the predicted bounding box overlaps with the labelled data), classification loss "cls" (which indicates how accurately the model predicts the labelled classes), and distribution focal loss "dfl" (which represents how well the algorithm predicts the presence of an object) [23]. As the training process progresses through multiple epochs, the decreasing trend of these losses signifies the model's improving ability to detect weapons. Additionally, the increase in precision, recall, and mean average precision (mAP) further indicates the model's enhanced performance over time.

Precision measures how accurately the model identifies weapons [24]. In an ideal scenario, if the model correctly classifies every detected weapon without any false positives, its precision would be 100%. However, if the model frequently misclassifies non-weapons as weapons (false positives), its precision decreases.

On the other hand, recall measures how well the model recognizes a weapon when it appears in front of the camera [24]. If the model detects every weapon instance without missing any (false negatives), its recall would be 100%.

In practice, achieving 100% precision and recall is nearly impossible. There is usually a tradeoff between the two—improving precision often comes at the cost of recall, and

vice versa. After training for 100 epochs, the best models we adopted in our prediction application achieved the following precision and recall values:

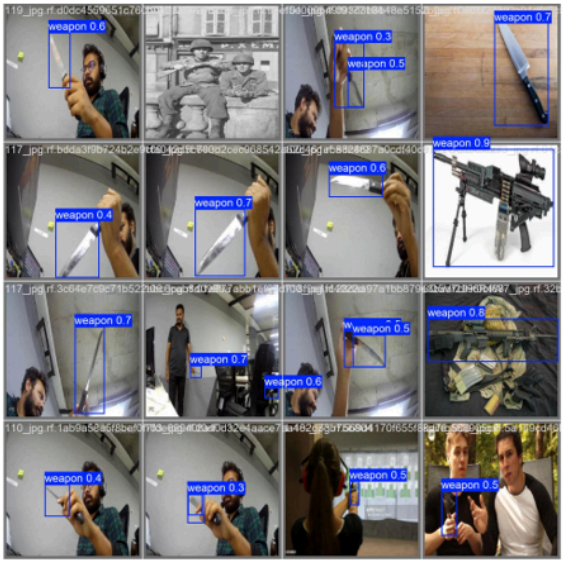
Model	Precision (%)	Recall (%)
YOLOv8l	82.6	82.8
YOLOv8n	76.8	80.4

Table 2: Precison and recall values of the trained YOLOv8l and YOLOv8n models

Figure 10 shows the models' validation performance for a batch of images:

A batch of labelled data for validation

YOLOv8l (large model) validation result



YOLOv8n (nano model) validation result

Figure 10: Model validation performance for a batch of labelled data

As the figure shows, both models perform well in detecting weapons. However, the nano model's confidence score, which reflects the model's certainty in its predictions, is slightly lower. Consequently, the nano model is more likely to miss weapons when they appear in front of the camera. This behaviour is expected and is discussed in the Method section.

CONCLUSION

In this project, I have demonstrated a practical application of machine learning to detect weapons in real time, helping to prevent violence before it occurs. Violence involving various types of weapons, such as knives, guns, and rifles, has increased significantly at an alarming rate over the past couple of decades worldwide.

To address this issue, I propose two weapon detection systems:

1. A central monitoring station-based application capable of analyzing streaming video from commercial IP cameras

2. A cost-effective, standalone edge camera device with built-in weapon detection analytics. The monitoring station application is demonstrated with a single camera feed but is easily scalable to support multiple cameras. Additionally, the developed system can notify users or authorities of any detected weapon activity in real-time via email or SMS, enabling immediate response and intervention.

CITATIONS

[1] International, Amnesty. “Gun Violence - Key Facts.” Amnesty.org, Amnesty International, 2024, www.amnesty.org/en/what-we-do/arms-control/gun-violence/.

[2] EFSGV. Efsgv.org Educational Fund to Stop Gun Violence the ROOT CAUSES of GUN VIOLENCE POLICYMAKERS MUST ADDRESS the SOCIAL and ECONOMIC INEQUALITIES THAT ARE the ROOT CAUSES of GUN VIOLENCE in IMPACTED COMMUNITIES of COLOR. Mar. 2020, efsgv.org/wp-content/uploads/2020/03/EFSGV-The-Root-Causes-of-Gun-Violence-March-2020.pdf.

[3] World Population Review. “Gun Deaths by Country 2020.” Worldpopulationreview.com, World Population Review, 2021, worldpopulationreview.com/country-rankings/gun-deaths-by-country.

[4] Canada, Public Safety. “Parliamentary Committee Notes: Complementary Firearms and Gun Violence Measures.” Securitepublique.gc.ca, 2023,

www.securitepublique.gc.ca/cnt/trnsprnc/brfng-mtrls/prlmntry-bndrs/20240220/14-en.aspx. Accessed 2 Feb. 2025.

[5] Wikipedia Contributors. “Machine Learning.” Wikipedia, Wikimedia Foundation, 29 Apr. 2019, en.wikipedia.org/wiki/Machine_learning.

[6] Wikipedia Contributors. “Object Detection.” Wikipedia, Wikimedia Foundation, 27 July 2019, en.wikipedia.org/wiki/Object_detection.

[7] Redmon, Joseph. “YOLO: Real-Time Object Detection.” Pjreddie.com, 2012, pjreddie.com/darknet/yolo/.

[8] “Neural Network (Machine Learning).” *Wikipedia*, 18 Feb. 2024, en.wikipedia.org/wiki/Neural_network_(machine_learning).

[9] Wikipedia Contributors. “Deep Learning.” *Wikipedia*, Wikimedia Foundation, 10 May 2019, en.wikipedia.org/wiki/Deep_learning.

[10] Wikipedia Contributors. “Python (Programming Language).” *Wikipedia*, Wikimedia Foundation, 4 May 2019, en.wikipedia.org/wiki/Python_(programming_language).

[11] Microsoft. “Visual Studio Code.” *Visualstudio.com*, Microsoft, 2024, code.visualstudio.com/.

[12] Ltd, Raspberry Pi (Trading). “Buy a Raspberry Pi 4 Model B.” *Raspberry Pi*, 2023, www.raspberrypi.com/products/raspberry-pi-4-model-b/.

[13] Wikipedia Contributors. “Supervised Learning.” *Wikipedia*, Wikimedia Foundation, 20 Mar. 2019, en.wikipedia.org/wiki/Supervised_learning.

[14] Wikipedia Contributors. “Unsupervised Learning.” *Wikipedia*, Wikimedia Foundation, 25 Mar. 2019, en.wikipedia.org/wiki/Unsupervised_learning.

[15] Wikipedia Contributors. “Reinforcement Learning.” *Wikipedia*, Wikimedia Foundation, 20 May 2019, en.wikipedia.org/wiki/Reinforcement_learning.

[16] Kaggle. “Kaggle: Your Home for Data Science.” *Kaggle.com*, 2024, www.kaggle.com/.

[17] “Roboflow: Go from Raw Images to a Trained Computer Vision Model in Minutes.” *Roboflow.ai*, roboflow.com/.

[18] “Weapon Detector Object Detection Dataset and Pre-Trained Model by Joo Assalim.” *Roboflow*, 2023,

universe.roboflow.com/joo-assalim-kbrtb/weapon-detector-pbxol. Accessed 2 Feb. 2025.

[19] “Ultralytics | Revolutionizing the World of Vision AI”, www.ultralytics.com.

[20] Ultralytics. “YOLOv8.” *Docs.ultralytics.com*, 12 Nov. 2023, docs.ultralytics.com/models/yolov8/.

[21]
(Download)

[22] https://github.com/ultralytics/assets/releases/download/v8.2.0/yolov8l.pt (Download)

[23] Ultralytics. “YOLO Performance Metrics.” *Docs.ultralytics.com*, 12 Nov. 2023, docs.ultralytics.com/guides/yolo-performance-metrics/.

[24] Wikipedia Contributors. “Precision and Recall.” *Wikipedia*, Wikimedia Foundation, 19 Apr. 2019, en.wikipedia.org/wiki/Precision_and_recall.

ACKNOWLEDGEMENT

I want to acknowledge my Science Fair coordinator, Ms. Heather Lai, for making Science Fair possible and my parents, specifically my dad, for helping me build and put everything together.

I also want to acknowledge anyone whose work I used and credited in this project.