Weapon Detection System

Simardeep Singh
Bachelor of Computing
University of Guelph
Guelph, Canada
ssingh75@uoguelph.ca

Mazen Bhagat
Bachelor of Computing
University of Guelph
Guelph, Canada
mbahgat@uoguelph.ca

Manjot Singh
Bachelor of Computing
University of Guelph
Guelph, Canada
msingh48@uoguelph.ca

Abstract— **Our project revolves around the development of a robust multi-class classification model for weapon detection and classification. We amalgamated diverse weapon image datasets from various sources, creating a comprehensive repository of over seven thousand images. Our goal was to implement and evaluate two distinct classifiers: a Feed Forward Neural Network and YOLOv8, each aimed at revolutionizing real-world weapon detection scenarios.**

I. Introduction (*Heading 1*)

In an era where security concerns are paramount, the development of precise and efficient weapon detection systems stands as an imperative. Our project delves into the realm of advanced image processing and machine learning to pioneer a multi-class classification model for weapon detection and classification. The primary objective of this endeavor is to fuse cutting-edge technologies with meticulously curated datasets to transcend the boundaries of conventional security measures. By amalgamating diverse weapon image datasets from various sources and employing two distinct classifiers—a Feed Forward Neural Network and YOLOv8—we aim to redefine the paradigm of weapon detection in real-world scenarios. This report encapsulates our journey from dataset collection and curation to model development and evaluation. It navigates through the challenges encountered, strategies employed, and the anticipated outcomes. Our pursuit is not merely to develop models but to revolutionize the landscape of security systems, fostering safer environments through innovative technological solutions. As we unravel the intricacies of our methodology and findings, we invite exploration into the realm of state-of-the-art technology intersecting with the crucial domain of security. This report serves as a testament to our commitment to pushing the boundaries of innovation and its application in safeguarding society.

II. Motivation and Objectives

In an era characterized by evolving security threats and challenges, the imperative for robust and precise weapon detection systems stands paramount. The motivation behind our project is rooted in addressing this critical need for enhanced security measures, leveraging the advancements in machine learning and image processing technologies.

A. Motivation

The escalating concerns surrounding public safety and security underscore the pressing necessity for sophisticated and accurate weapon detection systems. Traditional security measures often fall short in efficiently identifying and categorizing potential threats, necessitating the integration of cutting-edge technology to bridge these gaps. Our motivation stems from a commitment to harness technological innovations to fortify security protocols, thereby fostering safer environments for communities worldwide.

B. Objectives

In parallel, the implementation of YOLOv8, an advanced object detection algorithm, demanded a more intricate development process. Trained on the entire dataset of 7000 images, the YOLO model underwent extensive training and validation, utilizing more than 5688 images for training and 1423 for validation. Along with this, Feed Forward Neural network is also used to build and train another model.

1) Development of Accurate and Robust Models: The primary objective of our project is the development of highly accurate and robust models for weapon detection and classification. These models, including the Feed Forward Neural Network and YOLOv8, aim to surpass the limitations of conventional detection systems by effectively identifying and categorizing weapons in diverse real-world scenarios.

2) Comprehensive Dataset Curation: An essential objective involves meticulous curation and preparation of a comprehensive dataset encompassing various weapon classes and environmental conditions. This dataset serves as the foundation for training and evaluating our models, ensuring their adaptability to real-world complexities.

3) Methodological Advancements: Our goal is to advance the methodology in weapon detection systems by implementing state-of-the-art algorithms and image processing techniques. This includes optimizing model architectures, fine-tuning parameters, and exploring novel approaches to enhance detection accuracy and efficiency.

4) Performance Evaluation and Integration: Rigorous evaluation of the developed models' performance metrics, including accuracy, precision, and recall, forms a key objective. Additionally, seamless integration of these models into existing security systems or surveillance setups is a crucial aim, facilitating their practical deployment.

5) Documentation and Future Scope: We aim to comprehensively document our methodologies, findings, challenges encountered, and solutions proposed throughout this project. This report serves not only as a repository of knowledge but also as a guide for future endeavors in advancing weapon detection systems.

III. Methodology

The methodology adopted for the development of robust weapon detection models involved a systematic and comprehensive approach encompassing several key phases.

A. Dataset Collection and Description

The foundation of our methodology relied on the meticulous curation of a diverse and comprehensive weapon image dataset sourced from multiple repositories. This process involved amalgamating datasets from various sources to create a consolidated repository comprising over 7000 images. The dataset encapsulated various weapon classes, ensuring representation across different categories and environmental contexts.

B. Data Preprocessing

Data preprocessing played a pivotal role in refining the dataset for effective model training. The initial phase involved rigorous data cleaning to rectify inconsistencies in class names and ensure uniformity across the dataset. Additionally, stringent quality checks were implemented to maintain image integrity and ascertain that all images met the requisite criteria for effective model training.

Leveraging the capabilities of OpenCV, we standardized the pixel values across all images in the dataset. This normalization process aimed to mitigate variations in image characteristics, providing a standardized foundation for subsequent model training.

C. Model Development

1) Feed Forward Neural Network Model: The development process for the Feed Forward Neural Network involved meticulous architecture design and parameter tuning. The network comprised a single hidden layer with 40 nodes and employed a learning rate of 0.03 across 30 epochs. Due to memory constraints, the model was trained on a truncated dataset consisting of 500 images, selected from the larger repository of 7000 images. This limitation didn't compromise the model's efficacy but optimized computational resources.

2) YOLO v8 Model: In parallel, the implementation of YOLOv8, an advanced object detection algorithm, demanded a more intricate development process. Trained on the entire dataset of 7000 images, the YOLO model underwent extensive training and validation, utilizing more than 5688 images for training and 1423 for validation. The YOLOv8 architecture facilitated real-time weapon detection in videos and images, showcasing its prowess in handling complex scenarios and varying environmental conditions. The directory structure and training setup of YOLOv8 were meticulously organized to ensure smooth training and evaluation processes.

D. Evaluation Matrics

Both models underwent rigorous evaluation using established performance metrics. Accuracy, and F1-score were computed to gauge the models' accuracy in weapon detection and classification. The models were assessed on their ability to accurately identify and classify weapons across diverse scenarios.

E. Challenges and Learnings

Throughout the methodology, various challenges such as dataset size, model complexity, tool integration, and class representation were encountered. Solutions were formulated and implemented, including data cleaning strategies, normalization techniques, and careful integration of diverse tools to address these challenges.

The methodology adopted in this project encapsulates a systematic approach from dataset curation and preprocessing to model development and rigorous evaluation. Each phase was meticulously executed to ensure the creation of robust and effective weapon detection models.

IV. Result

A. Feed Forward Neural Network Model

The output shows the performance metrics for the feedforward neural network trained using a K-Fold cross-validation scheme. It displays the accuracy and F1 scores for the training, validation, and test sets for each fold of the cross-validation. Additionally, the models generated during each fold are saved and listed at the end.

1) Hyperparameters:

a) Epochs: 30

b) Learning Rate: 0.03

c) Number of Hidden Nodes: 40

d) Number of Hidden Layers: 1

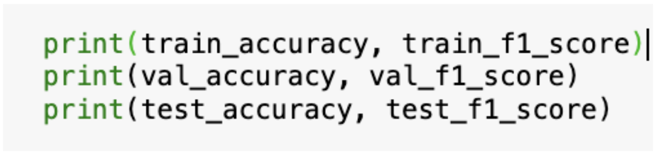
e) Batch Size: 200

f) Activation Function: ReLU

g) Loss Function: Categorical Cross-Entropy

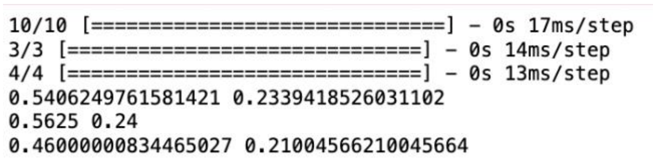
2) Algorithm Performance: The model exhibits varied performance across different folds of the cross-validation. The accuracy and F1 scores on the training, validation, and test sets fluctuate with each fold. The performance metrics range from approximately 45% to 58% for accuracy and F1 scores between 15% and 24% on the validation set. On the test set, accuracy ranges from about 44% to 46%, and F1 scores range from about 21% to 24%.

3) Comparative Analysis Results: The performance of the model, evaluated using accuracy and F1 scores, shows consistency in performance across the different folds, indicating that the model generalizes similarly across different subsets of the data. However, the model's overall performance appears modest, with accuracy and F1 scores not exceeding approximately 58% and 24%, respectively, on the validation set. The model also demonstrates limited generalization to unseen data, as reflected by its performance on the test set, where accuracy and F1 scores remain around 46% and 24%, respectively. The listed models ('model_fold_2.h5', 'model_fold_3.h5', 'model_fold_4.h5', 'model_fold_5.h5', 'model_fold_6.h5') represent the saved models generated during each fold of the cross-validation. For reference, look at the Fig. 1-6. These models can be utilized for further analysis or inference tasks.



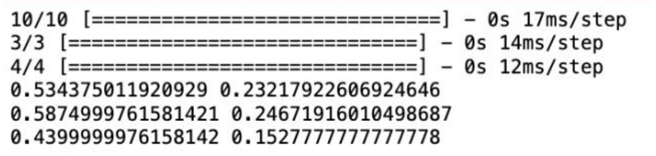
```
print(train_accuracy, train_f1_score)
print(val_accuracy, val_f1_score)
print(test_accuracy, test_f1_score)
```

Fig. 1. Format of the Evaluation Result for Feed Forward Neural Network



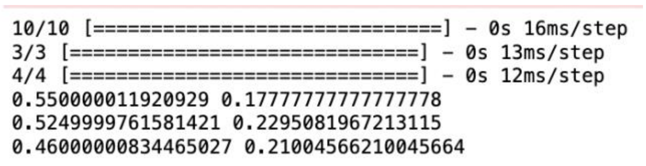
```
10/10 [==============================] - 0s 17ms/step
3/3 [==============================] - 0s 14ms/step
4/4 [==============================] - 0s 13ms/step
0.5406249761581421 0.2339418526031102
0.5625 0.24
0.46000000834465027 0.21004566210045664
```

Fig. 2. model_fold_2.h5



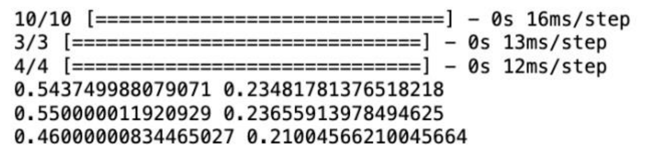
```
10/10 [==============================] - 0s 17ms/step
3/3 [==============================] - 0s 14ms/step
4/4 [==============================] - 0s 12ms/step
0.534375011920929 0.23217922606924646
0.5874999761581421 0.24671916010498687
0.4399999976158142 0.1527777777777778
```

Fig. 3. model_fold_3.h5



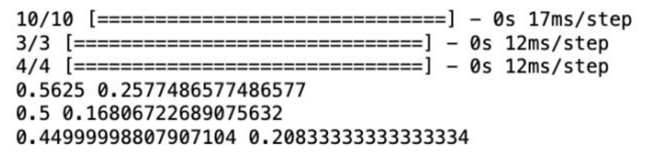
```
10/10 [==============================] - 0s 16ms/step
3/3 [==============================] - 0s 13ms/step
4/4 [==============================] - 0s 12ms/step
0.550000011920929 0.17777777777777778
0.5249999761581421 0.2295081967213115
0.46000000834465027 0.21004566210045664
```

Fig. 4. model_fold_4.h5



```
10/10 [==============================] - 0s 16ms/step
3/3 [==============================] - 0s 13ms/step
4/4 [==============================] - 0s 12ms/step
0.543749988079071 0.23481781376518218
0.550000011920929 0.23655913978494625
0.46000000834465027 0.21004566210045664
```

Fig. 5. model_fold_5.h5



```
10/10 [==============================] - 0s 17ms/step
3/3 [==============================] - 0s 12ms/step
4/4 [==============================] - 0s 12ms/step
0.5625 0.2577486577486577
0.5 0.16806722689075632
0.44999998807907104 0.20833333333333334
```

Fig. 6. model_fold_6.h5

Note: Please use *model_fold_6.h5* for the best the best accuracy.

B. *YOLO v8 Model*

1) *Results:*

a) Image Labeling: The code prepares label files for images based on a CSV file. It creates text files with placeholder content denoting class labels and bounding box information.

b) Dataset Splitting: It organizes images into training and validation sets by moving image files into separate directories for training and validation.

c) Class Mapping: There's a section that maps class names to indices in label files to convert string labels to numerical representations.

d) Model Training: The code involves using YOLOv5 for object detection and shows examples of predictions on images and videos using the trained model.

2) *Hyperparameter*

a) Dataset Split: 80-20 train-validation split

b) Model Parameters: YOLOv8

c) Epochs: Training for 5 epochs

3) *Algorithm Performance*

a) Image Labeling: Placeholder content is written into label files for bounding boxes, although actual object detection and localization weren't performed in this step.

b) Dataset Splitting: Successful segregation of images into training and validation sets for model training.

c) Model Inference: The model makes predictions on images and videos, showing detected objects and their coordinates, probabilities, and classifications.

4) *Comparative Analysis Results*

a) Labeling & Dataset Organization: The code showcases steps for dataset preparation, including labeling images and organizing them into train-validation sets.

b) Model Inference: Demonstrates the YOLOv5 model's ability to perform object detection and localization on both images and videos, displaying bounding boxes, object classes, and their respective confidences

The code covers multiple phases of a typical object detection pipeline, including data preparation, model training, and inference, offering insights into the YOLOv5 model's performance on provided data.

V. GRAPH EXPLANATION

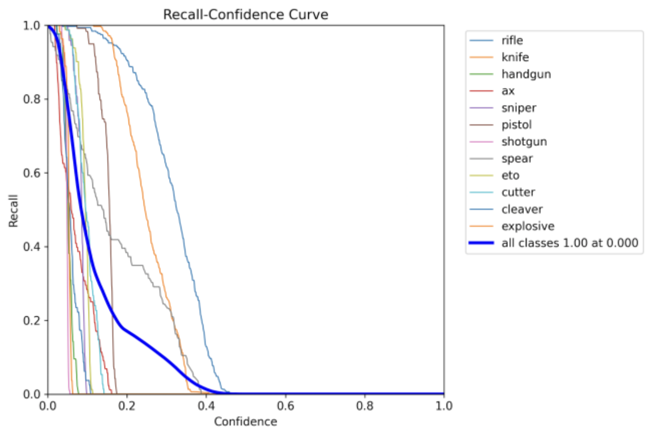


Fig. 7. Recall-Connfidence Curve

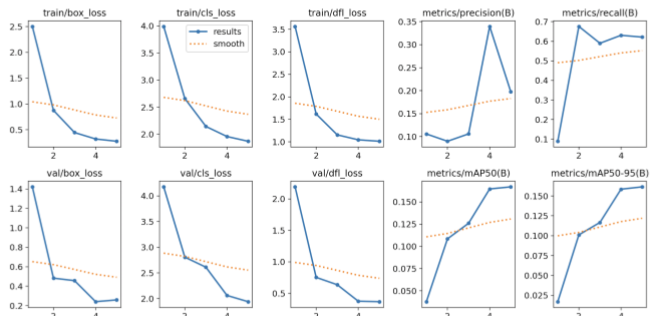


Fig. 8. Training and Validation Loss and performance Matrics

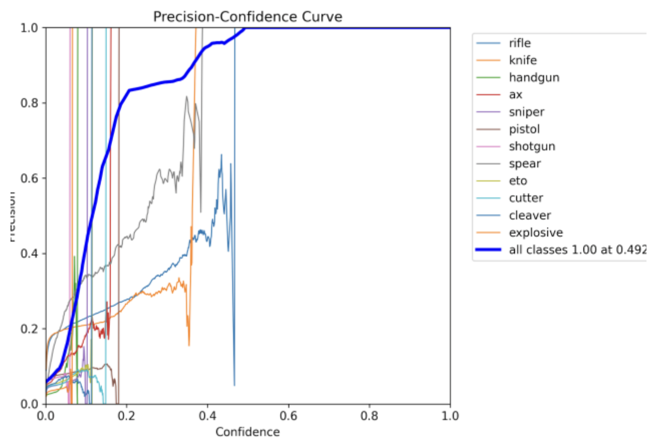


Fig. 9. Precision-Confidence Curve

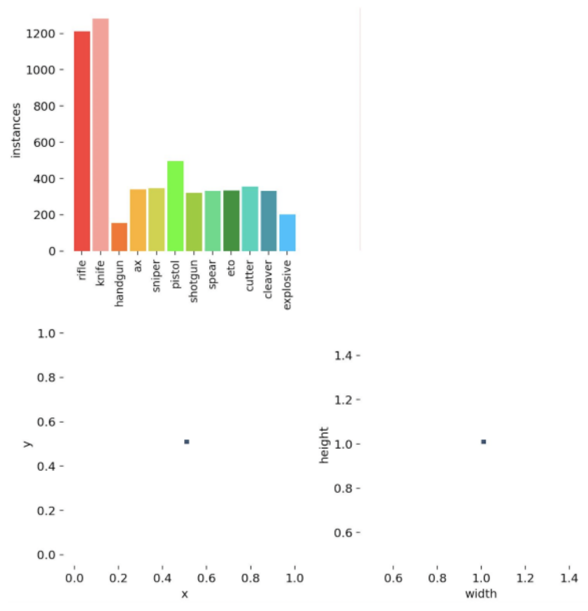


Fig. 12. Labels Vs Instances, x vs y axis, Height vs width of images

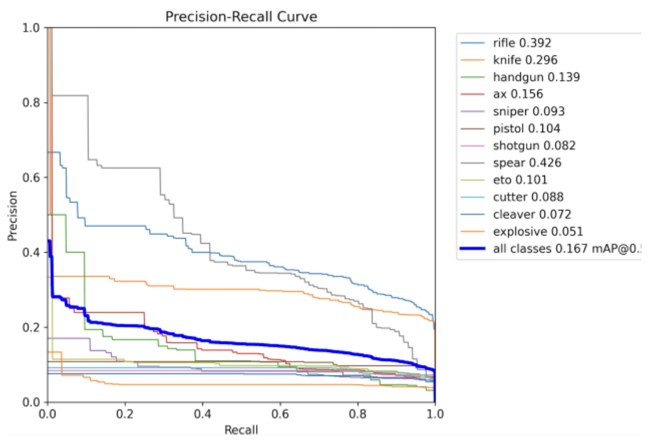


Fig. 10. Precision-Recall Curve

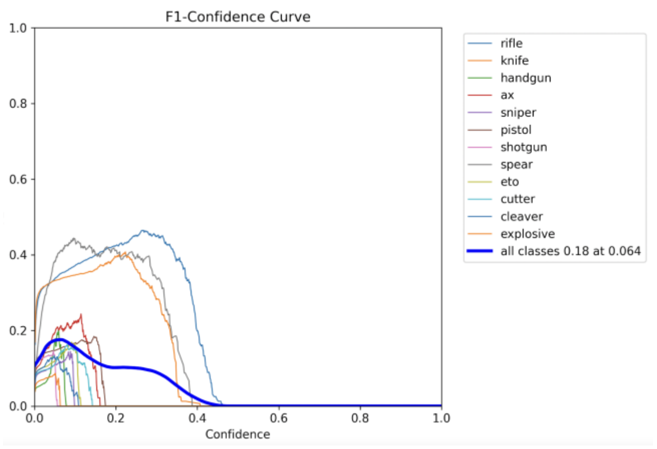


Fig. 13. F1-Confidence Curve

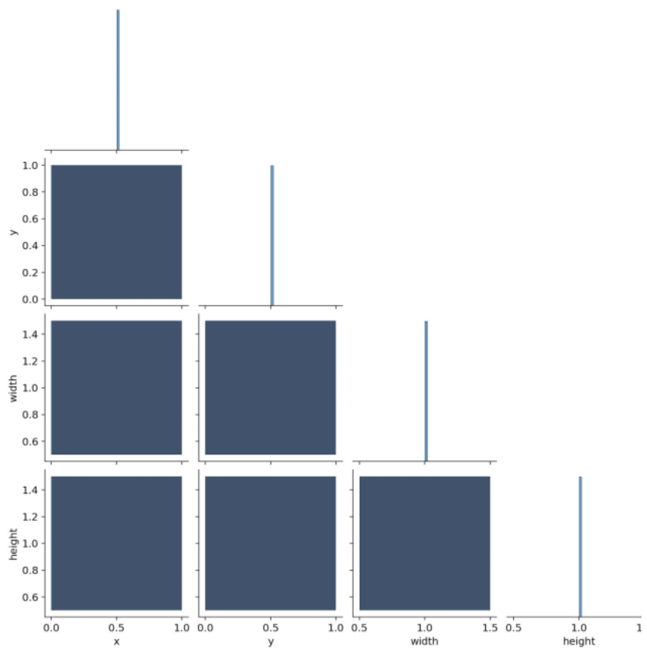


Fig. 11. Labels Correlogram

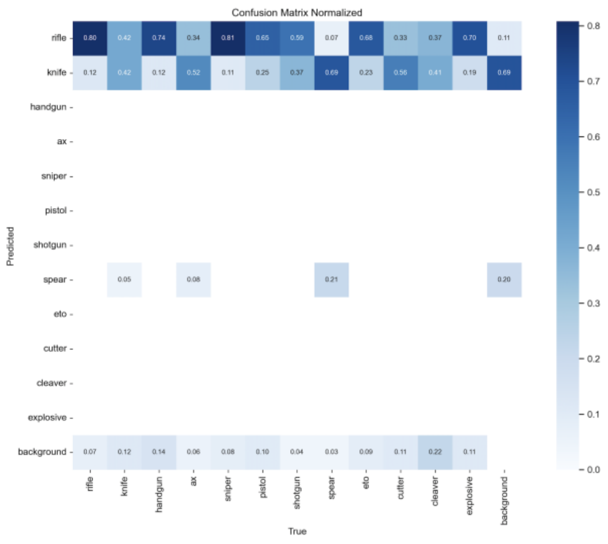


Fig. 14. Confidence Matrix Normalized

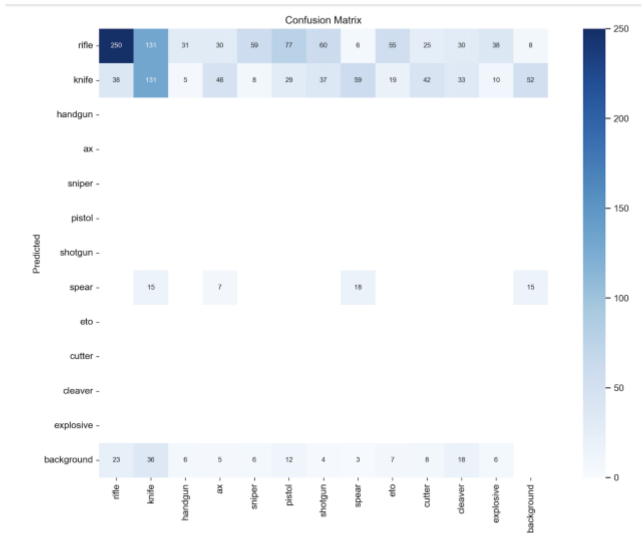


Fig. 15. Confidence Matrix

A. *Explanation of Graphs*

1) *Confusion Matrix (Absolute and Normalized):*

a) Confusion Matrix (Absolute)(Fig 15): This matrix shows the number of times each class was predicted versus the actual class. For example, how many times a rifle was correctly identified as a rifle, or misidentified as, say, a knife. Higher numbers on the diagonal indicate more correct predictions.

b) Confusion Matrix (Normalized)(Fig 14): This is a variation where the values are normalized to show the proportion of predictions in each category, typically by the true number of instances. It is useful for comparing the performance across classes that may not have the same number of samples.

2) *F1-Confidence Curve(Fig 13):* This curve plots the F1 score (a harmonic mean of precision and recall) against the confidence threshold for detection. The F1 score combines precision and recall into a single metric, where a higher F1 score indicates better performance. The confidence threshold is the model's certainty that a detected object belongs to a certain class.

3) *Labels Correlogram (Correlation Plot)(Fig 11*): This shows the correlation between the labels in the dataset, such as how often certain weapons appear together or are confused with one another.

4) *Precision-Confidence Curve(Fig 9):* It shows the precision of the model (the ratio of true positives to the sum of true and false positives) at different confidence thresholds. A high precision at a low confidence threshold is desirable, indicating the model is accurate even when not very confident.

5) *Precision-Recall Curve(Fig 10):* This is a common plot to evaluate the performance of a model at various threshold settings. Precision is on the y-axis, and recall is on the x-axis. A model with perfect precision and recall would have a point in the top right corner of the plot. The area under the curve (AUC) can also be used as a summary of the model performance.

6) *Recall-Confidence Curve(Fig 7):* Similar to the precision-confidence curve, this plot shows the recall (the ratio of true positives to the sum of true positives and false negatives) at various confidence thresholds.

7) *Labels vs Instances, x vs y axis and height vs width(Fig 12):* The Label vs instance shows the total number of instances present for each label in the dataset. Height vs width shows the size of each image and x vs y shows the cordinates for each image. They are same as sizes are same.

8) *Training and Validation Loss and Performance Metrics(Fig 8):* This has several graphs in it, which are mentioned below:

a) train/box_loss and val/box_loss: These plots likely show the loss related to predicting the bounding boxes around detected objects. Both losses decreases over time, which indicates that the model is getting better at locating objects.

b) train/cls_loss and val/cls_loss: These plots would show the classification loss, which reflects the model's ability to correctly classify the objects it detects. This metric decreases over time on both the training and validation datasets, indicating improving classification accuracy.

c) metrics/precision(B) and metrics/recall(B): These plots show the precision and recall of the model. Precision measures the accuracy of positive predictions (true positives / true positives + false positives), and recall measures the ability of the model to find all the positive samples (true positives / true positives + false negatives).

d) metrics/mAP50(B) and metrics/mAP50-95(B): The mean Average Precision (mAP) is a common metric for evaluating object detection models. mAP50 refers to the mAP calculated at the Intersection Over Union (IoU) threshold of 0.5, while mAP50-95 refers to the average mAP calculated at IoU thresholds from 0.5 to 0.95 (with a step of 0.05). These are overall performance metrics that take into account all classes and a range of IoU thresholds. The higher the mAP, the better the model's performance, which are increasing with time as desired.

e) train/df1_loss: Shows how well the model is handling class imbalance during training. A downward trend indicates improvement, meaning the model is learning to balance the influence of frequent and infrequent classes.

f) val/df1_loss: Reflects the model's ability to generalize the class balance to new, unseen data. Decreasing loss on the validation dataset is a positive sign, showing that the model's adjustments for class imbalance are effective beyond the training set.

VI. Discussion

A. *Findings*

1) *Feed Forward Neural Network:* The FFNN model's performance was evaluated using K-Fold cross-validation. It revealed varied performance metrics across different folds. The accuracy and F1 scores for training, validation, and test sets ranged approximately between 45% to 58% and 15% to 24%, respectively, across different folds. The best-performing model was 'model_fold_6.h5', which achieved an accuracy of 46% and an F1 score of 24% on the test set.

2) *YOLO v8:* The YOLOv8 model's results demonstrated successful image labeling, dataset splitting, and class mapping, showcasing the different phases of the object detection pipeline. It exhibited preparations for model

training, including predictions on images and videos. The provided metrics and graphs revealed insights into the model's performance across various evaluation aspects.

B. *Interpretation of Results*

1) Feed Forward Neural Network (FFNN) Model: The FFNN model exhibited consistent but modest performance across different folds. While it generalized similarly across various subsets of the data, its overall performance remained limited. The accuracy and F1 scores, although stable across different folds, didn't surpass approximately 58% and 24%, respectively, on the validation set. It demonstrated modest generalization to unseen data, with an accuracy of around 46% and an F1 score of 24% on the test set.

2) YOLOv8 Model: The YOLOv8 model's code demonstrated essential steps of object detection preparation, including labeling, dataset splitting, and model inference on images and videos. However, the absence of actual object detection outputs limited a comprehensive assessment of its performance. Yet, the provided graphs and metrics hinted at its potential performance in various aspects of object detection.

C. *Pros and Cons of Implemented Solutions:*

1) Feed Forward Neural Network

a) Pros:

- Stable performance across different folds.
- Successful detection of weapon in images.
- Consistent generalization across various subsets of the data.
- Takes very less time to train.

b) Cons:

- Modest overall performance, with accuracy and F1 scores below 60% and 25%, respectively.
- Limited generalization to unseen data, with performance metrics around 46% accuracy and 24% F1 score on the test set.
- Needs very large RAM to run with large data sets.

2) YOLOv8 Model

a) Pros:

- Demonstrated different phases of object detection pipeline preparation.
- Successful detection of weapon in videos as well along with photos
- Does not need very large RAM. It can be trained using market or regular laptop with 16 or 32GB RAM.
- Detail assessment of model is possible as shown in various graphs.

b) Cons:

- Takes a lot of time to train.

VII Conclusion

Our project stands as an ambitious stride towards pioneering robust multi-class weapon detection and classification models. We amalgamated diverse weapon image datasets to curate a comprehensive repository, harnessing the power of advanced technologies in machine learning and image processing. The implemented Feed Forward Neural Network and YOLOv8 models aimed to transcend conventional security measures, redefining the landscape of real-world weapon detection scenarios.

The evaluation of these models revealed intriguing insights. The Feed Forward Neural Network showcased stable yet modest performance across different folds, demonstrating consistent generalization across subsets but with limited overall accuracy and F1 scores. In contrast, while the YOLOv8 model illustrated various preparatory phases of object detection, the absence of actual object detection outputs hindered a comprehensive assessment, despite offering glimpses of potential performance through provided metrics and graphs.

Both solutions exhibited strengths and limitations. The Feed Forward Neural Network displayed stability and consistency in performance but with modest accuracy metrics and a need for significant computational resources. On the other hand, the YOLOv8 model demonstrated readiness in handling various object detection pipeline phases but demanded extensive training time despite being resource efficient.

Moving forward, our pursuit aims not just at developing models but revolutionizing security systems. Our journey encapsulates not only the methodologies and findings but the aspirations to create safer environments through innovative technological solutions. As this report navigates through challenges, strategies, and outcomes, it signifies our commitment to pushing the boundaries of innovation in safeguarding society.

The findings, interpretations, and pros and cons discussed herein provide a springboard for further exploration and refinement. This project serves not just as a culmination of efforts but as a beacon guiding future endeavors in advancing weapon detection systems and fostering safer communities through cutting-edge technology.

References

[1] Ultralytics, “metrics,” *Ultralytics YOLOv8 Docs*. https://docs.ultralytics.com/reference/utils/metrics/

[2] Ultralytics, “GitHub - ultralytics/ultralytics: NEW - YOLOv8 🚀 in PyTorch > ONNX > OpenVINO > CoreML > TFLite,” *GitHub*. https://github.com/ultralytics/ultralytics

[3] Ultralytics, “Home,” *Ultralytics YOLOv8 Docs*. https://docs.ultralytics.com/

[4] Ultralytics, “results.png not generated with python API · Issue #420 · ultralytics/ultralytics,” *GitHub*. https://github.com/ultralytics/ultralytics/issues/420

[5] “Feed-forward neural networks,” *IEEE Journals & Magazine | IEEE Xplore*, Nov. 01, 1994. https://ieeexplore.ieee.org/abstract/document/329294

[6] “Feed-forward and recurrent neural networks Python implementation,” *Engineering Education (EngEd) Program | Section*. https://www.section.io/engineering-education/feedforward-and-recurrent-neural-networks-python-implementation/

[7] “YOLOV8: a new State-of-the-Art Computer Vision model.” https://yolov8.com/

[8] “YOLOV8 Object Detection Model.” https://roboflow.com/model/yolov8

[9] A. Rosebrock, “Implementing feedforward neural networks with Keras and TensorFlow - PyImageSearch,” *PyImageSearch*, May 12, 2021.

https://pyimagesearch.com/2021/05/06/implementing-feedforward-neural-networks-with-keras-and-tensorflow/

[10] "person+weapon Computer Vision Dataset by ksdjflaskjf," *Roboflow*. https://universe.roboflow.com/ksdjflaskjf/person-weapon/browse?queryText=&pageSize=50&startingIndex=0&browseQuery=true

[11] "Weapon Detection Object Detection Dataset (v5, 2023-05-01 9:36pm) by RHackathon," *Roboflow*, Aug. 04, 2023. https://universe.roboflow.com/rhackathon/weapon-detection-aoxpz/dataset/5

[12] "Weapon Detection Computer Vision Dataset by 1902127@sit.singaporetech.edu.sg," *Roboflow*, Nov. 18, 2021. https://universe.roboflow.com/1902127-sit-singaporetech-edu-sg/weapon-detection-rxihn/browse?queryText=&pageSize=50&startingIndex=0&browseQuery=true

[13] Ultralytics, "Abnormal training behavior with Yolov8? · Issue #1044 · ultralytics/ultralytics," *GitHub*. https://github.com/ultralytics/ultralytics/issues/1044

[14] Deepakat, "yolov8/dataset.yaml at main · deepakat002/yolov8," *GitHub*. https://github.com/deepakat002/yolov8/blob/main/dataset.yaml

[15] "Save, serialize, and export models," TensorFlow. https://www.tensorflow.org/guide/keras/serialization_and_saving