Weapon Detection from CCTV Footage

Grace McPadden, Isabelle Perez

I. ABSTRACT

Gun violence remains a critical public safety issue in the United States. Although surveillance systems are often used to mitigate such threats, human limitations– including fatigue and alertness– significantly reduce their effectiveness. This project explores the use of deep learning to automate weapon detection in CCTV footage. Specifically, we compare the performance of three object detection models– YOLOv5, YOLOv8, and YOLOv11– for identifying firearms and knives in still images. The dataset, derived from a mock attack scenario, includes manually annotated frames from 3 camera angles. Models were trained and tested on Google Colab, and evaluated using metrics such as mean average precision (mAP@0.5), precision, recall, and loss components. Additionally, we explored hyperparameter tuning, finding that reducing the learning rate to 0.005 improved detection accuracy. Our findings indicate that YOLOv5 achieved the best performance and was most suitable for real-time deployment.

II. INTRODUCTION

GUN related deaths continue to pose a threat to public safety in the United States, with over 46,000 deaths reported in 2023 alone. In response to growing security concerns, surveillance technologies have become increasingly widespread; however, the effectiveness of these systems is hindered by human limitations. Research shows that continuous monitoring of CCTV footage leads to visual fatigue and a significant decrease in detection accuracy– leading to a failure rate of up to 83% after just 40 minutes of continuous monitoring [1]. These findings highlight a critical need for automated tools for threat detection. The development of such systems could revolutionize security by providing faster and more reliable responses to potential threats, ultimately saving lives and reducing the burden on security personnel.

In this project, we leverage deep learning to automate weapon detection in video surveillance. More specifically, we apply the You Only Look Once (YOLO) family of object detection models to identify firearms and knives in still images derived from CCTV footage. YOLO models are known for their real-time detection capabilities, offering a balance between speed and accuracy [6], making them ideal for deployment in dynamic and high-risk environments. This approach allows for immediate identification of threats, significantly reducing the response time compared to manual surveillance.

By addressing the limitations of manual surveillance and contributing to the growing field of computer vision for public safety, this work offers a scalable and reliable tool for enhancing threat response in high-risk environments. The automation of weapon detection has the potential to transform the way security is managed in public spaces, offering a proactive and efficient solution to mitigate gun violence and improve safety outcomes.

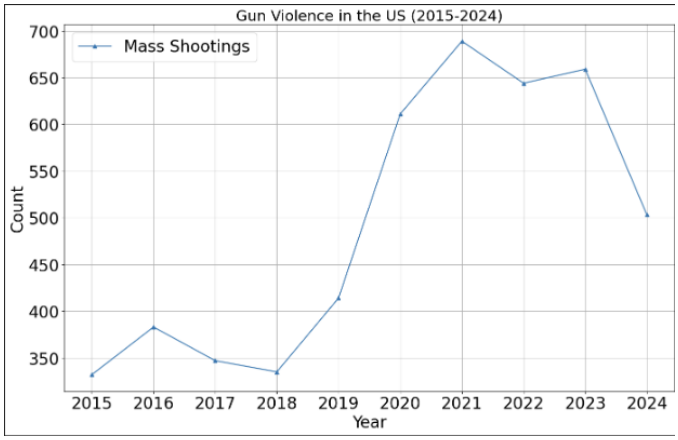

Fig. 1. Mass shootings per year since 2015. Data from the Gun Violence Archieve

III. BACKGROUND AND RELATED WORK

CNN based detectors are commonly used for real-time object identification. For the task of weapon detection, there are multiple methods including sliding window, two-stage, and one stage detectors. Sliding window methods are least efficient and most computationally expensive as they require sliding a region proposal across the image and evaluating each region. Two stage detectors (ex. R-CNN or Faster R-CNN) operate by first proposing 2,000 bounding box possibilities and then refining and selecting the most likely object bounding box. One stage detectors (ex. YOLO) segment objects in a single pass, offering both an advantage in speed and computational resources over both two-stage and sliding window object detectors.

Yadav et al. [1] conducted a comprehensive review of both classical and deep learning methods for weapon detection. This paper explores the accuracy and speed of both one-stage detectors using YOLO and two-stage detectors such as R-CNN. YOLOv3 performed the best out of the one stage detectors and R-CNN architecture performed the best out of the two-stage detectors. Overall, the one stage detectors were shown to have the best balance of speed and accuracy, making it the most suitable for real time weapon detection.

Bhatti et al. [2] compared several modern detection architectures, including SSD MobilNetV1, YOLOv3, Faster RCNN-Inception ResNetV2, and YOLOv4 for weapon detection algorithms. They found that out of both sliding window and region proposal methods, YOLOv4 performed the best with a mean average precision of 91.73%. This paper also discussed preprocessing techniques and dealing with objects that are small

or partially occluded.

Jain et al. [3] implemented single shot-multibox detector (SSD) and Faster R-CNN weapon detectors. They found that single shot detectors were able to nearly match the accuracy of R-CNN two-stage detectors while having a much higher speed for processing each frame. Considering both accuracy and speed, SSD models were shown to be better for real time segmentation of weapons.

Narejo et al. [4] developed a YOLOv3-based weapon detection model using 53 layers stacked over the original 53 layer network. This model was able to outperform Alexnet and Faster R-CNN based models for weapon detection tasks.

These studies show that YOLO-based models consistently outperform two-stage and sliding window methods for real time weapon detection by providing a balance between speed and accuracy Based on these finding, out project focuses on implementing and evaluating the most recent YOLO models (YOLOv5, YOLOv8, and YOLOv11) for the task of weapon detection in CCTV footage.

To better understand the advantages of YOLO-based models, it is important to briefly examine how the architecture functions. Introduced by Redmon et al. [6], YOLO is fundamentally different from traditional detection pipelines in that it treats object detection as a single regression problem, rather than relying on separate stages for region proposal and classification. This enables it to use a single convolutional network to predict bounding boxes and class probabilities in just one pass.

Specifically, the input image is divided into an SxS grid, with each cell is tasked with predicting a fixed number of bounding boxes and class probabilities for objects where the center falls within that cell. This eliminates the computational overhead associated with sliding window and region proposal networks. The original YOLO architecture consists of 24 convolutional layers used for feature extraction, followed by 2 fully connected layers responsible for making the final predicton. This structure enables YOLO models to balance accuracy with computational efficiency, making it most effective for real time predictions.

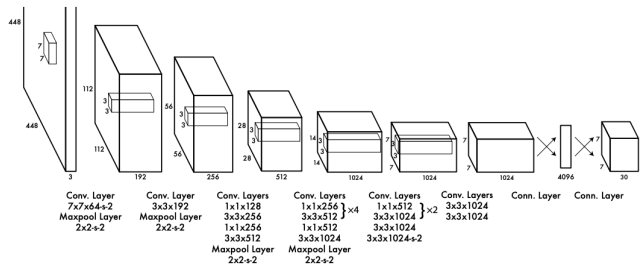

Fig. 2. Structure of YOLO network. Reproduced from [6].

IV. Methods

The data set used for this project was created for a study by Salazer González et al. [5]. It consists of over 5000 image frames from a mock attack recorded on CCTV cameras from three distinct angles. Each frame was manually annotated with bounding box coordinates and class labels for any visible weapons. The data includes three categories for handguns, short rifles, and knives. Additionally, the original resolution of these images is 1920x1080 pixels.

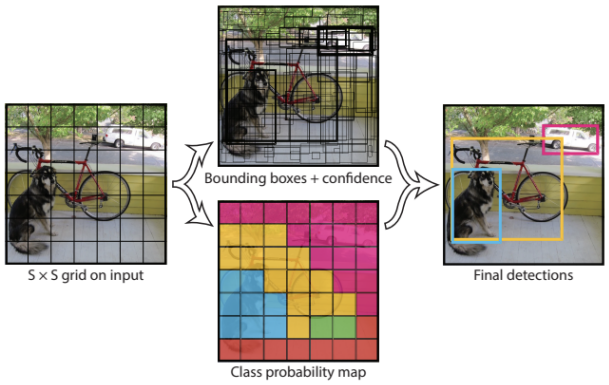

Fig. 3. YOLO architecture illustrating the division of an image into a grid and prediction of bounding boxes. Reproduced from [6].

As a part of preprocessing, the coordinates and type of weapons were extracted from the associated .yaml files, normalized relative to the image dimensions, and converted to .txt files in the YOLO format. Each line in the resulting file encodes the class ID and normalized coordinates for each bounding box. Data augmentation was not applied manually, as default data augmentation techniques are included in the YOLO pipeline. These augmentations include random horizontal flips, scaling, brightness adjustments, and image rotations. Such transformations improve generalization and help to reduce overfitting, which is especially important due to the limited number of frames and angles in the data set.

Fig. 4. Sample frame from mock attack dataset, Cam1

The architecture of the model family used, YOLO, consists of three major components. First is the backbone, a convolutional neural network (CNN) used to process the images, extracting features and using convolutional layers to learn hierarchical representations of the image. This is the followed by the neck, which aggregates and processes features from the backbone using a Feature Pyramid Network (FPN). Lastly is the detection head to output the actual predictions. This portion is used to predict bounding boxes, objectness scores,

and class probabilities for the classification.

We evaluated three model architectures in our project: YOLOv5s, YOLOv8s, and YOLOv11s. Hyperparameter tuning and data augmentation was also used to optimize the accuracy of the models. Training was conducted on Google Colab using an NVIDIA L4 GPU.

The results of the model training and evaluation were assessed using a variety of standard object detection metrics. These include bounding box loss (box_loss), classification loss (cls_loss), objectness loss (obj_loss), mean average precision at an Intersection over Union (IoU) threshold of 0.5 (mAP@0.5), and recall. Each of these metrics captures a different aspect of the model's performance, offering a comprehensive view of its detection capabilities.

Bounding box loss (box_loss) evaluates the model's ability to accurately localize objects within an image. In modern YOLO architectures, this is calculated using the Complete Intersection over Union (CIoU) loss, which extends traditional IoU by incorporating distance, overlap, and aspect ratio consistency. CIoU loss penalizes bounding boxes not only for low overlap but also for misalignment of centers and differences in aspect ratios. The loss is defined as follows:

$$L_{\text{box}} = 1 - \text{IoU} + \frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2} + \alpha v \tag{1}$$

where $\mathbf{b}$ and $\mathbf{b}^{gt}$ are the centers of the predicted and ground truth boxes respectively, ρ is the Euclidean distance between box centers, c is the diagonal length of the smallest enclosing box covering both, v measures aspect ratio similarity, and α is a positive trade-off parameter. This formulation encourages precise localization, better aspect ratio alignment, and robust overlap, improving detection performance especially in crowded or small-object scenarios.

Classification loss (cls_loss) measures the model's ability to correctly classify the object detected within each bounding box. In YOLO models, this is implemented using the Binary Cross-Entropy with Logits Loss function, which combines a sigmoid activation with binary cross-entropy in a numerically stable way. The classification loss is computed as:

$$\begin{aligned} L_{\text{cls}} = -\sum_{i=1}^{C} \big[& y_i \cdot \log \sigma(\hat{y}_i) \\ & + (1 - y_i) \cdot \log(1 - \sigma(\hat{y}_i)) \big] \end{aligned} \tag{2}$$

where $y_i \in \{0, 1\}$ is the ground truth label for class i, $\hat{y}_i$ is the predicted logit for class i, and $\sigma(\cdot)$ denotes the sigmoid function. Here, C is the total number of object classes in the dataset. This loss encourages the model to produce high confidence scores for the correct classes while minimizing incorrect predictions, contributing to more reliable object classification across a wide range of categories.

Lastly, objectness loss (obj_loss) evaluates the model's confidence in whether a given bounding box actually contains an object. This is calculated by comparing the predicted objectness score to the ground truth— 1 if an object is present in the grid cell and 0 if not. This calcuation uses binary cross-entropy loss function as follows:

$$L_{\text{obj}} = -\left[y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y}) \right] \tag{3}$$

where $y \in \{0, 1\}$ is the ground truth objectness label (1 if an object is present in the grid cell, 0 otherwise), and $\hat{y} \in [0, 1]$ is the predicted objectness score. This metric helps control the trade-off between detecting actual weapons and avoiding false positives in background regions.

Mean Average Precision at IoU 0.5 (mAP@0.5) is a standard detection performance metric that combines both precision and recall across all classes. It is computed by first calculating precision-recall curves for each class, then computing the average precision (AP) at a threshold of 0.5 IoU—meaning the predicted bounding box must overlap at least 50% with the ground truth to be considered correct. The mean of all AP values across classes gives the mAP score. A higher mAP@0.5 indicates strong overall detection accuracy. The formal equation is as follows:

$$\text{mAP@0.5} = \frac{1}{C} \sum_{i=1}^{C} \text{AP}_i(\text{IoU} \geq 0.5) \tag{4}$$

where C is the number of object classes, $\text{AP}_i(\text{IoU} \geq 0.5)$ and is the average precision for class at the IoU threshold of 0.5.

Recall measures the proportion of actual objects that the model successfully detects. It is calculated as the number of true positives divided by the total number of ground truth objects (true positives + false negatives). High recall indicates that the model is able to detect most weapons in the dataset, though it does not account for how many false positives it also returns.

Together, these metrics provide a multi-dimensional evaluation of the YOLO models' ability to detect, classify, and localize weapons accurately and efficiently. Tracking improvements across all of them allows for targeted tuning and deeper insight into where each model excels or needs refinement.

V. Results

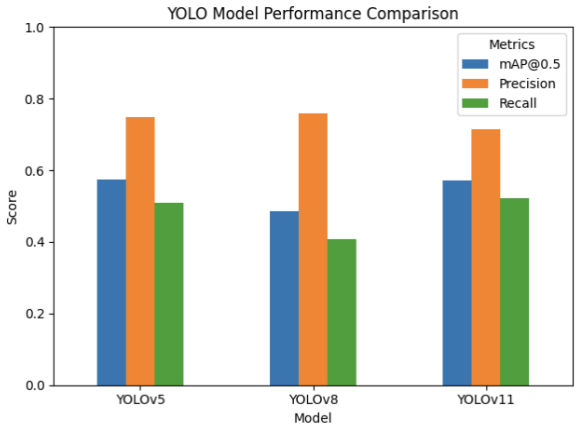

Fig. 5. Comparison of mAP, precision, and recall for YOLOv5, YOLOv8, and YOLOv11 on the validation set.

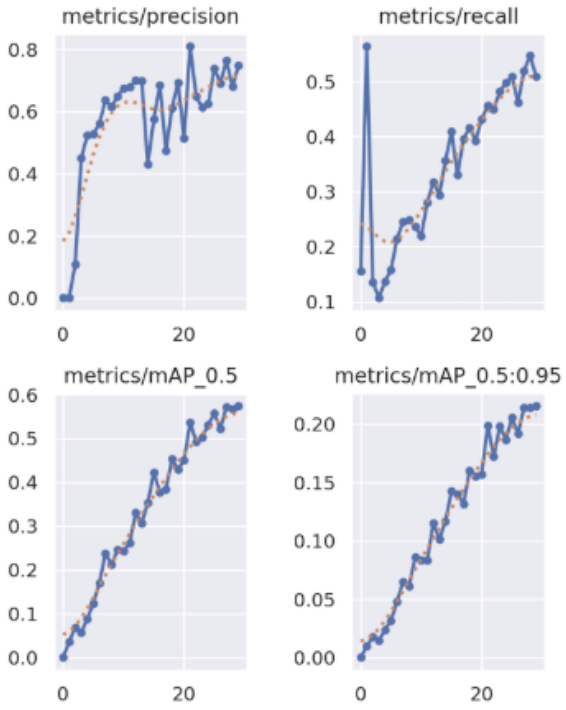

Fig. 6. Precision, recall, and mAP@0.5 vs. epochs for YOLOv5 before learning rate optimization

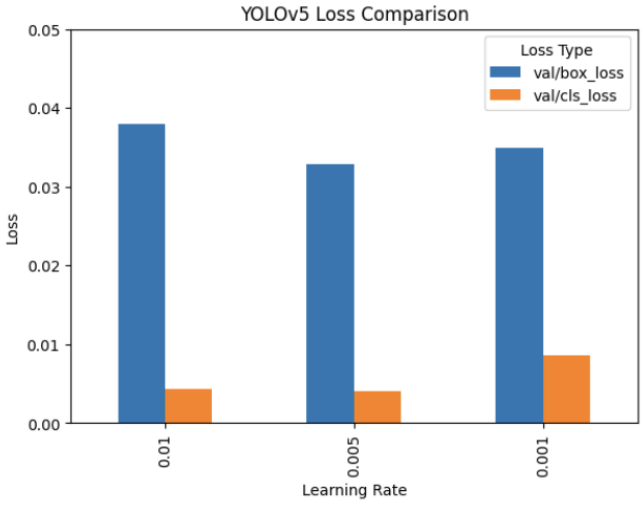

Fig. 7. YOLOv5 validation loss at different learning rates (0.01, 0.005, 0.001)

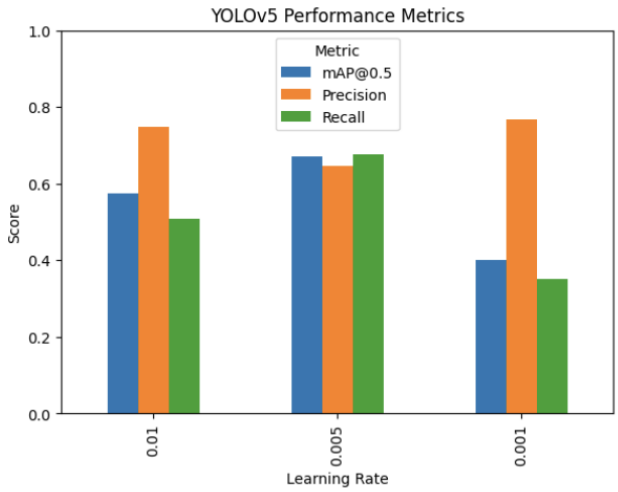

Fig. 8. YOLOv5 performance metrics at different learning rates (0.01, 0.005, 0.001)

The first part of our analysis was comparing three recent YOLO models– YOLOv5, YOLOv8, and YOLOv11. Each of these models were trained on our data set using 640p image resolution, 30 epochs, and a batch size of 16. These models were evaluated using three key metrics on the validation set– mean average precision (mAP@0.5), precision, and recall.

As shown in Figure 5, YOLOv5 and YOLOv11 achieved the highest mAP scores (0.5746 and 0.5709 respectively), while YOLOv8 lagged slightly behind at 0.4856. In terms of precision, YOLOv8 led with 0.7604, followed closely by YOLOv5 at 0.7522. YOLOv11 had the lowest precision at 0.7138. This indicates that YOLOv5 and YOLOv8 were more effective at minimizing false positives during detection. However, YOLOv11 achieved the highest recall (0.5211), suggesting stronger overall detection coverage. While all models demonstrated comparable overall performance, YOLOv5 was selected for further optimization and hyperparameter tuning due to its higher mAP and precision scores.

The hyperparameter we optimized for was the learning rate. As seen in Figure 6, the precision, recall, and mAP@0.5 showed rapid oscillations with the increasing number of epochs, which could indicate that the learning rate of the model is too high. We trained the model again, reducing the learning rate from its default of 0.01 to 0.005 and 0.001. As shown in figure 8, a learning rate of 0.005 had the highest mAP (0.6696) compared with a learning rate of 0.01 and 0.001 (0.5746 and 0.4013, respectively).

The YOLOv5 model with a tuned precision rate of 0.005 had an overall mAP@0.5 of 0.668 across all the classes. The Short_rifle class had the highest mAP of 0.750, handguns were second highest with an mAP of 0.680, and knives had the lowest mAP of 0.575. Knives had the smallest number of instances by far, with approximately 200 instances compared to approximately 600 for handguns and approximately 1200 for Handguns. Despite having more instances, handguns likely had a lower mAP due to their smaller size in the image frame.

This also likely contributed to the lower rate of recognition for knives.

As seen confusion matrix for the trail on YOLOv5 with a learning rate of 0.005 (figure 9), most of the errors came from mistaking the background for the object. The highest number of these errors came from incorrectly labeling the background as a handgun. For future work, it would be helpful to have more objects in the knives and handgun category.

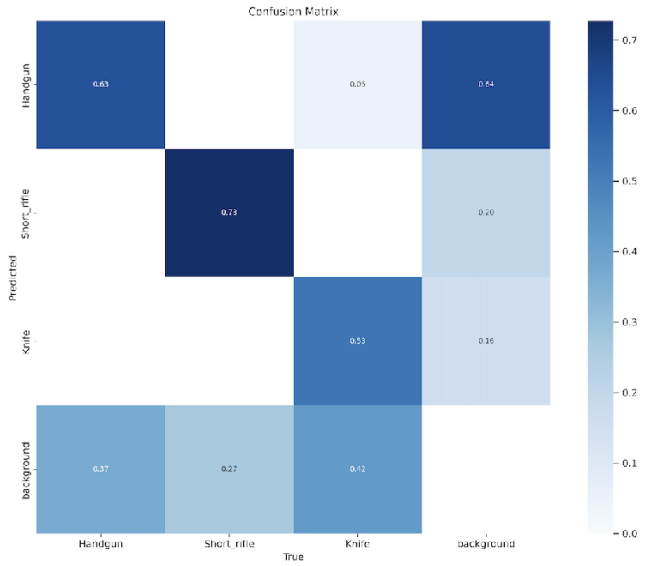

Fig. 9. YOLOv5 confusion matrix, learning rate = 0.005

VI. Conclusion

As gun safety continues to be a major concern in the United States, it's critical that CCTV footage is properly monitored to help prevent gun-related death. Studies show that human monitoring can be ineffective, therefore automatic identification via machine algorithms have the potential to make a significant impact. The YOLOv5, YOLOv8, and YOLOv11 model architectures have been shown to be quite robust and effective in weapon detection. In this analysis, we use these techniques to model images from mock attack data, optimizing the learning rate for better performance. Results showed that for this particular dataset, the YOLOv5 is most effective, especially in minimizing false positives. Overall, the results were quite promising and suggest that these object detection models could greatly improve weapon detection from CCTV footage.

For future improvements to the model, it would be helpful to have a larger and more diverse dataset with a variety of camera angles, lighting, and background noise. This would enhance the model's ability to generalize to real-world surveillance scenarios. Additional hyperparameter tuning beyond the learning rate—such as weight decay, augmentation settings, or freezing layers—could also improve model performance. Another key area for improvement is enhancing the model's ability to handle small and distant objects, which can be difficult to detect accurately. One possible solution could be to incorporate multi-scale feature extraction. Lastly, a next step would be testing the integration of these models into real-time systems, evaluating their performance on live CCTV footage to determine their effectiveness in dynamic environments and under varying operational conditions.

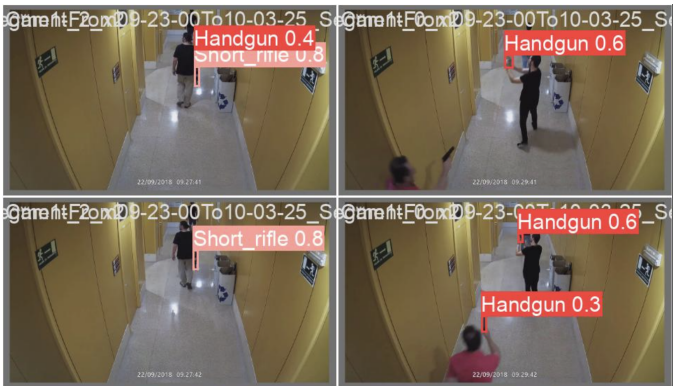

Fig. 10. Detection examples, from validation set.

Appendix A
Appendix A: Code and Reproducibility

Full Colab Notebook: https://tinyurl.com/3ked6huy

References

[1] V. Yadav, P. et al., "A comprehensive study towards high-level approaches for weapon detection using classical machine learning and deep learning methods," *IEEE Xplore*, 2020.
[2] M. T. Bhatti, et al., "Weapon detection in real-time CCTV videos using deep learning," *IEEE Access*
[3] R. Jain, et al., "Weapon detection using artificial intelligence and deep learning for security applications," *International Journal of Computer Vision*
[4] A. Narejo, et al., "Weapon detection using YOLO v3 for smart surveillance system," *IEEE Transactions on Security and Surveillance*
[5] DeepKnowledge-US, "US real-time gun detection in CCTV: An open problem dataset," DeepKnowledge, 2023. [Online]. Available: https://deepknowledge-us.github.io/US-Real-time-gun-detection-in-CCTV-An-open-problem-dataset/
[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *arXiv preprint arXiv:1506.02640*, 2016.
[7] Gun Violence Archive, "Gun Violence Archive 2023," [Online]. Available: https://www.gunviolencearchive.org/. [Accessed: May 6, 2025].