Smart-Watcher: An AI-Powered IoT Monitoring System for Small-Medium Scale Premises

Ahmed Abdelmoamen Ahmed
Department of Computer Science
Prairie View A&M University
Prairie View, TX, USA
Email: amahmed@pvamu.edu

Bernard Nyarko
Department of Computer Science
Prairie View A&M University
Prairie View, TX, USA

***Abstract*—With recent advances in both Artificial Intelligence (AI) and Internet of Things (IoT) capabilities, it is more possible than ever to implement surveillance systems that can automatically identify people who might represent a potential security threat to the public in real time. Imagine a surveillance camera system that can detect various on-body weapons, suspicious objects, and traffic. This system could transform surveillance cameras from passive sentries into active observers, which would help prevent a possible mass shooting in a school, stadium, or mall. In this paper, we tried to realize such systems by implementing Smart-Watcher, an AI-powered threat detector for intelligent surveillance cameras. The developed system can be deployed locally on the surveillance cameras at the network edge. Deploying AI-enabled surveillance applications at the edge enables the initial analysis of the captured images on-site, reducing communication overheads and enabling swift security actions. We developed a mobile app that alerts system users about any detected suspicious objects in an image and video captured by several cameras at the network edge. Also, Smart-Watcher can generate a high-quality segmentation mask for each object instance in the photo, along with the confidence percentage. Smart-Watcher can recognize eight object classes, including baseball bats, birds, cats, dogs, guns, hammers, knives, and human faces. Smart-Watcher was evaluated using various performance metrics such as classification time and accuracy.**

***Index Terms*—IoT; AI; Machine Learning (ML); Mobile Computing; Communication; Edge Computing; Monitoring System.**

I. Introduction

Artificial intelligence (AI) has emerged as a powerful tool in various fields, with Machine Learning (ML) playing a crucial role in driving innovation [1], [2]. The Convolutional Neural Network (CNN) model [3] has emerged as the ML model of choice in image and video recognition applications, as it excels in evaluating pixel data, making it well-suited for such applications. The rising demand for reliable and automated security and public safety mechanisms has increased the need to explore the usage of AI in this domain [4].

Imagine an innovative threat detection system that can automatically glimpse security perils in video streams without manual intervention. Such a system would significantly boost traditional camera systems' monitoring and surveillance capabilities. This paper presents Smart-Watcher, an AI-powered Internet of Things (IoT) monitoring system for small-medium scale premises. Smart-Watcher leverages the powers of the CNN model to perform descriptive and generative tasks, making it highly efficient in detecting potential security threats.

Smart-Watcher aims to minimize communication overheads between the network edge surveillance devices (i.e., cameras) and the centralized video processing servers hosted in the cloud core. The developed system is organized into two parts: executing on the edge's monitoring cameras and on core video processing servers. Smart-Watcher uses motion sensors to detect any motion in the monitored environment. When an action is seen, these sensors send a signal to a Raspberry Pi device hosted locally, instructing it to start capturing images for such activity, thus reducing the overheads of continuously capturing videos when there is no motion in the scene.

An imagery dataset of more than 2,000 images of various threatening objects was constructed, including baseball bats, guns, hammers, knives, birds, cats, dogs, and human faces. Figure 1 shows examples of various types of the eight object categories from our imagery dataset. Most of the images in our dataset are in their natural environments because object detection is highly dependent on contextual information. The pet images are included in our dataset because they may trigger false alarms. Smart-Watcher could smartly analyze the whole scene as a security threat scenario because it can handle multi-class classification problems, where it can classify the input image as belonging to one or more of the eight object classes.

II. Related Work

The growing global human population and the increasing need for security and safety have led to a surge in demand for innovative surveillance systems [1]. As a result, the field of information security has become a multibillion-dollar industry [5]. The recent advancements in AI and IoT capabilities have enabled the realization of intelligent surveillance systems capable of real-time threat identification [6].

For instance, Hawk-eye [1] leverages AI-powered threat detection algorithms to enhance the capabilities of surveillance cameras for proactive threat prevention. Hawk-eye is an example of an AI-powered public surveillance system that offers increased coverage and intelligence for threat detection, thereby contributing to public safety [4].

In the area of real-time object tracking and detection, Kumar et al. [7] introduced a system that utilizes multiple cameras for

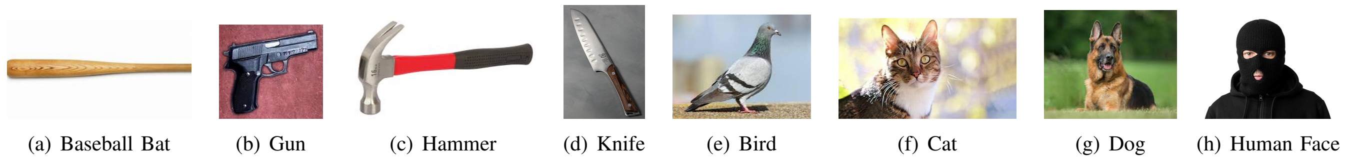


Fig. 1. Sample Examples from our Dataset.

real-time object tracking to enhance security and situational awareness. In the health sector, Smart-Monitor [6] is a patient monitoring system for IoT-based healthcare systems using deep learning for e-healthcare. The developed method can monitor the surveyed patients' physiological signals.

AI and IoT have been used in agriculture to develop intelligent greenhouse monitoring systems. In [8], the authors introduced an automated smart greenhouse based on an adaptive Neuro-Fuzzy Inference System (ANFIS), which is used to boost crop production inside the house. Using various IoT sensors, the developed system can monitor and control several ecological conditions inside the greenhouse, such as temperature, humidity, and soil moisture.

In summary, most of the existing video surveillance approaches work on either recorded videos [5] or offloading the video data into a centralized server in the cloud [9], which is inappropriate for real-time threat detection. Furthermore, none of the current video surveillance approaches can be deployed on the security camera due to its limited computational capabilities, which precludes taking advantage of minimizing the communication delays in such mission-critical security tasks.

III. System Design

Figure 2 shows the distributed architecture of Smart-Watcher, which is arranged with components executing on IoT devices at the camera side and on centralized servers on the cloud side. This architecture orchestrates the seamless interaction between the various hardware devices and software technologies used in the system, culminating in a comprehensive intelligent surveillance solution.

The hardware components of the camera side are shown in layer 1. Layer 2 describes the ML models used in Smart-Watcher, which includes the Mask R-CNN model [10]. It also shows the Intermediate Representation (IR) model, which runs on a Neural Compute Stick 2 (NCS2) device at the camera side. Layer 3 illustrates the user interface of Smart-Watcher running on the cloud server. As shown in layer 4, a mobile app is developed so that users can detect suspicious objects in an image and video captured by several cameras at the edge.

The Mask R-CNN model is used to construct segmentation masks on different Regions of Interest (RoI) of the threatening objects in an image. The model extends the Faster R-CNN model [3] by adding a new module for predicting segmentation masks on each RoI and the existing modules for feature extractions, RoI classification, and bounding box regression.

We trained a Mask R-CNN model with four convolutional layers, one input layer, and one output layer. $I = [i_1, i_2, .., i_r]$ and $O = [o_1, o_2, ..., o_h]$ represent the input and output vectors,

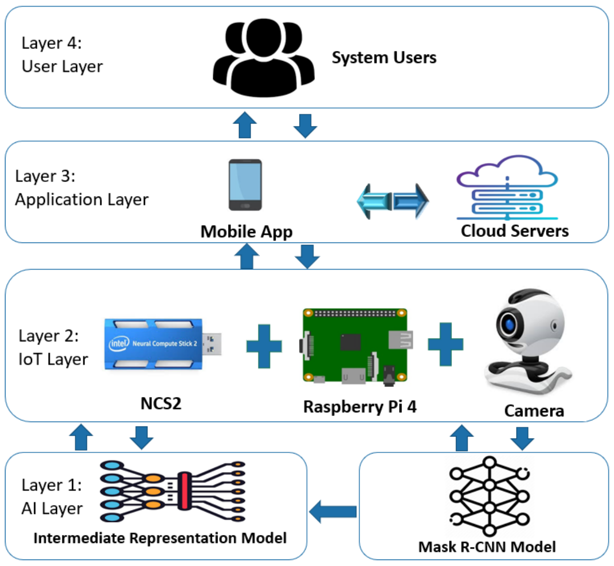


Fig. 2. System Architecture

respectively, where r means the number of elements in the input feature set and h is the number of classes. The network's main objective is to learn a compressed dataset representation. In other words, it tries to approximately learn the identity function F, which is defined as:

$$F_{W,B}(I) \simeq I \tag{1}$$

where W and B are the network weights and biases vectors.

We further customized the Mask R-CNN model by implementing object classification using focal loss and optimized the position of the bounding box using a regression method. The focal loss, Υ, is defined as:

$$\Upsilon = -\sigma_\tau (1 - p_\tau)^\gamma \; log(p_t) \tag{2}$$

where γ is the modulating factor that balances the training dataset, σ_τ is a factor used to balance the number of true positive and false negatives samples, p_τ is the estimated probability of the ground truth, and $log(p_\tau)$ is the cross entropy for the model's binary classification.

The regression method, for optimizing the position of the bounding box, predicts four parameters $((P_{x1}; P_{y1}) ; (P_{x2}; P_{y2}))$, representing the offset coordinates $((x1, y1) , (x2, y2))$ between anchor box A and the ground-truth box Y. Their ground-truth offsets $((T_{x1}; T_{y1}) ; (T_{x2}; T_{y2}))$ is expressed as:

$$T_x = \frac{(Y_x - A_x)}{A_w} \quad ; \quad T_y = \frac{(Y_y - A_y)}{A_h} \tag{3}$$

where Y is the ground-truth box and A is the anchor box. The width and height of the bounding box are represented by w and h. The regression loss, Θ, is defined as:

$$\Theta = \sum_{j\in\{x1;y1;x2;y2\}} S_L(P_j - Y_j) \tag{4}$$

where S_L is a smoothing function defined over x dimension.

Before moving the trained Mask R-CNN model to the NCS2 device, we had to convert it into an optimized Intermediate Representation (IR) model based on the trained network topology, weights and biases values. We used the Intel OpenVINO toolkit [11] to generate the IR model, which is the only format that the inference engine at NCS2 accepts and understands. The conversion process involved removing the convolution and pooling layers that are not relevant to the inference engine at the stick. In particular, OpenVINO splits the trained model into two types of files: XML and Bin extension. The XML files contain the network topology, while the BIN files contain the weights and biases binary data.

IV. Implementation

A. *Smart-Watcher Training*

Although standard object detection datasets (e.g., Microsoft COCO [12]) exhibit volume and various examples, they are unsuitable for thread detection as they annotate object categories, not including guns and other threatening objects. Therefore, we collected more than labeled 2k images for training the Mask R-CNN model from different sources such as Kaggle [13] and Google Web Scraper [14]. Our dataset is divided into three parts: training, validation and testing. The number of images in each phase is determined based on the fine-tuned hyperparameters and structure of the ML model.

For training the Mask-RCNN model, we used the LabelImg tool [15] to annotate 250 images for each object class. LabelImg is a graphical image annotation software tool used to label and define object bounding boxes within the images meticulously. The output annotations are saved in XML files using the PASCALVOC format. Figure 3 illustrates an annotated image with multiple human faces. We exported the spatial dimensions of each region in the input image in JSON format, which is then fed to the Mask-RCNN model as a one-dimensional array of the mask.

Given the limited-resource constraints of IoT devices in the network edge (i.e., Raspberry Pi 4), we decided to customize the Mask-RCNN model to generate a lightweight object detection model suitable to be executed on IoT devices. This decision aligned with the TensorFlow 2 Detection Model Zoo collection used in this work, specifically designed to operate on IoT devices. The integration of Intel's NCS2 further elevated the system's performance of the object detection process.

The training images must have the same size before feeding them as input to the model. Our model was trained with colored (RGB) images with resized dimensions of 400×400 pixels. We set the batch size and number of epochs to be 50 images and 30 epochs, respectively. The model training was carried out using a server computer equipped with a 4.50GHz

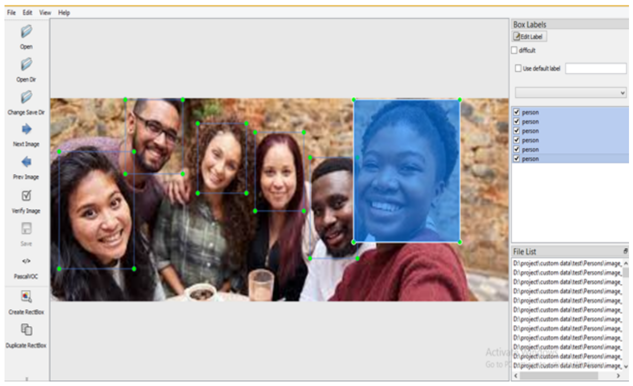


Fig. 3. Annotated Image used in Training the Mask R-CNN Model

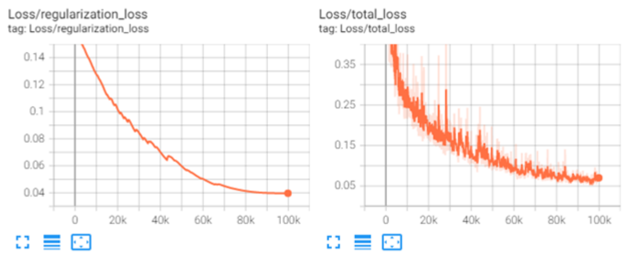


Fig. 4. The Regularization and Total Loss of the CNN Model

Intel Core™ i7-16MB CPU processor, 16GB of RAM, and CUDA GPU capability. The training phase took approximately 30 hours to run 30 epochs. We took a snapshot of the trained weights every 5 epochs to monitor the progress. The training error and loss are calculated using this equation:

$$M = \frac{1}{n}\sum_{i=1}^{n}(y_i - x_i)^2 \tag{5}$$

where M is the mean square error of the model, y is the value calculated by the model, and x is the actual value. M represents the error in object detection and the construction of a mask over the wrong object.

We measured these performance metrics during the training process: regularization loss (see Figure 4), total loss (see Figure 4), learning rate (see Figure 5), classification loss, and localization loss. The regularization loss metric indicates our model's ability to generalize. The classification and localization losses determine the capability of Mask R-CNN to classify a target object and the model's regression ability to create the bounding rectangular box around that object, respectively.

B. *Physical Implementation*

Figure 6 shows the physical prototype implementation of Smart-Watcher on the camera side, where we used a Raspberry Pi 4 device, NCS2, Logitech C920 webcams, PIR motion sensors, buzzers, push buttons, and LED lights; in addition to Python Face recognition software, and TensorFlow Custom Object Detection toolkit.

The LEDs have three colors (i.e., red, yellow, and green), indicating the different states of the system. The green LED

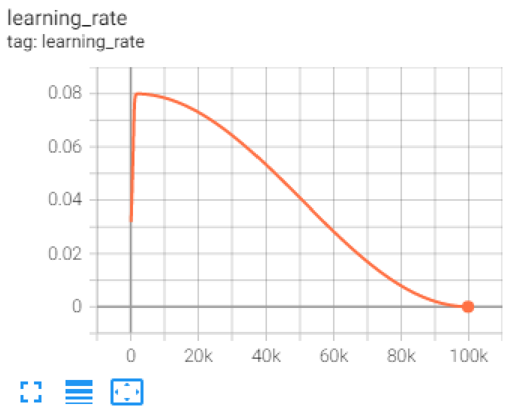


Fig. 5. The Learning Rate the CNN Model

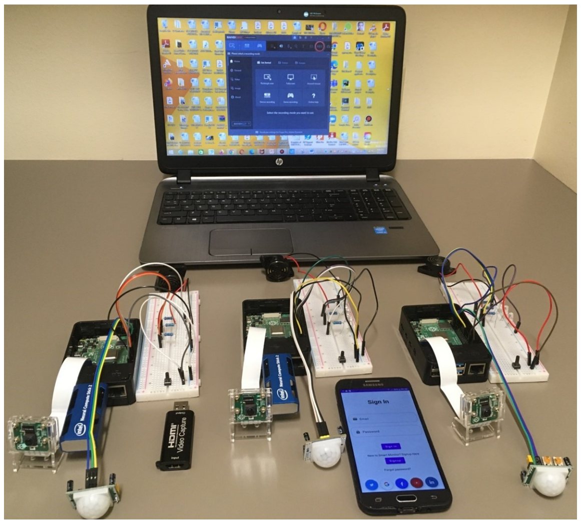


Fig. 6. The Physical Implementation of Smart-Watcher

indicates that there is no invasion of the monitored premises. The LED turns yellow when the motion sensor is triggered with a trained human face. If an unknown human face is detected, the buzzer is turned on, and the LED turns red.

C. User Interface

The user interface was developed as a responsive and self-contained mobile app application to enhance user experience using the system. The app was built using the Android Development Environment ADT bundle (64-bit). The app uses different technologies and tools, including the Android SDK and XML, to create the front-end interface. We used Python's Flask Web Framework as a middleware between the app and the database server on the cloud and SQLite for the database.

Figure 7 shows a snapshot of the camera view selector page where users can view live video streams from different cameras simultaneously. The mobile app allows users to calibrate their faces by capturing photos of their faces from different angles, which are considered `familiar faces`. When the

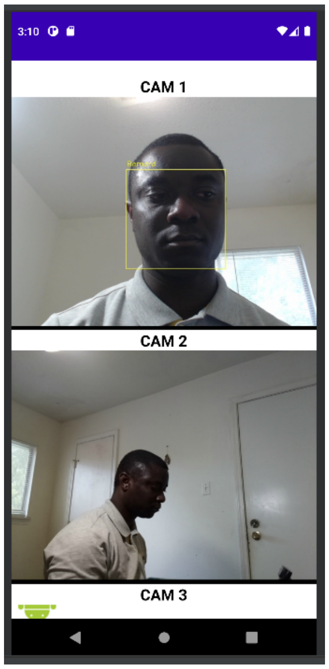


Fig. 7. A Screenshot of the Smart-Watcher Mobile App

system detects one of those familiar faces, the LED turns yellow, and the alarm is not triggered.

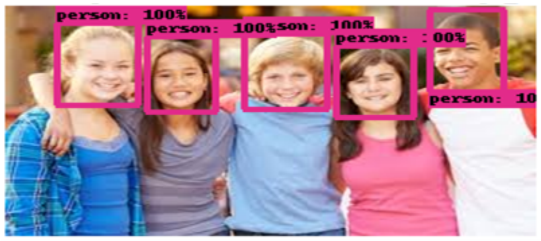
To run the mobile app on top of the Mask R-CNN model, we had to wrap the model, implemented on TensorFlow, as a REST API using the Flask web framework. In other words, the communication between TensorFlow and Flask is coordinated through that REST API. When a photo is captured, Flask uses the `POST` method to send the image from the mobile app to TensorFlow via an HTTP header.

V. Experimental Evaluation

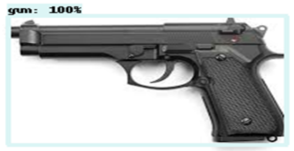
Figure 8 illustrates an example of the inference result of the Mask R-CNN model. The figure shows that the model created segmentation masks around human faces, a gun, and pets with high confidence percentages. Class prediction, mask construction, and displaying results took around 1.61 seconds. A significant portion of that time is consumed to identify the regions of interest and create the segmentation masks.

To provide timely alerts to users, Smart-Watcher sends them email and text message alerts when an unknown human face is detected. Figure 9 shows a sample email alert with attached scene images. This enables users to assess the threat level, guiding them to take the appropriate action.

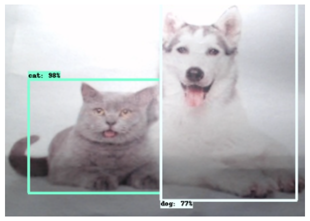
Table I shows the Mask R-CNN model's average classification accuracy and prediction time across the eight classes. The Mask R-CNN model achieved an overall average classification accuracy of 96.3%. The average prediction time of the Mask R-CNN was measured to be 1.61s (seconds). We noted that the prediction accuracy of human faces and guns were 99.2% and 98.2%, respectively. With the fact that machine guns are



(a) Human Faces

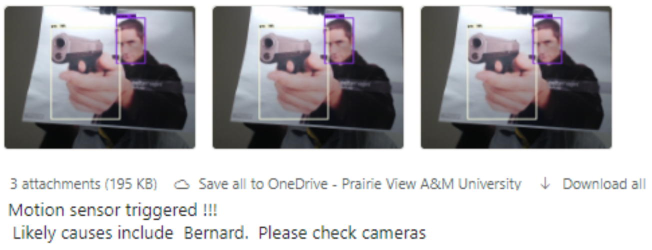


(b) Gun



(c) Pets

Fig. 8. The Inference Results of the Mask R-CNN Model



3 attachments (195 KB) Save all to OneDrive - Prairie View A&M University Download all

Motion sensor triggered !!!

Likely causes include Bernard. Please check cameras

Fig. 9. A Snapshot of the Motion Detection Trigger Email Alert

the most used weapon in the deadliest mass shootings in the United States [16], Smart-Watcher can be a helpful tool to mitigate mass shootings if deployed on the various surveillance cameras at public places such as schools, malls, parks, etc.

VI. Conclusions

This paper presented the design and implementation of Smart-Watcher. This AI-driven system empowers surveillance cameras to detect potential security threats in real-time, offering protection beyond passive monitoring. Smart-Watcher could preemptively prevent catastrophic incidents like mass shootings by identifying on-body weapons, suspicious objects, and unusual activities. The system implementation used the Intel NCS2 device to boost the processing capabilities of a Raspberry Pi 4 device for deploying the Mask R-CNN model locally on the surveillance camera without cloud computing dependence. A user-friendly interface was developed on top of the Mask R-CNN model to allow users to interact with the system conveniently. We carried out a sets of experiments to evaluate the performance and classification accuracy of our system, paying particular attention to the prediction time.

TABLE I
The Average Classification Accuracy and Prediction Time of the Mask R-CNN Model

Class	No. of Testing Images	Accuracy	Prediction Time
Baseball Bat	100	97.6%	1.54s
Gun	235	98.2%	1.61s
Hammer	104	98.2%	1.59s
Knife	123	93.2%	1.45s
Bird	170	95.6%	1.63s
Cat	271	94.3%	1.57s
Dog	236	94.1%	1.88s
Human Face	459	99.2%	1.67s

Acknowledgments

This research work is supported in part by the National Science Foundation (NSF) under grants # 2011330, 2200377 and 2302469. Any opinions, findings, and conclusions expressed in this paper are those of the authors and do not necessarily reflect NSF's views.

References

[1] A. A. Ahmed and M. Echi, "Hawk-eye: An ai-powered threat detector for intelligent surveillance cameras," *IEEE Access*, vol. 9, pp. 63 283–63 293, 2021.
[2] A. A. Ahmed and S. Ahmed, "A real-time car towing management system using ml-powered automatic number plate recognition," *Algorithms*, vol. 14, no. 11, 2021.
[3] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.
[4] A. A. Ahmed and N. Okoroafor, "An ml-powered risk assessment system for predicting prospective mass shooting," *Computers*, vol. 12, no. 2, 2023.
[5] G. F. Shidik, E. Noersasongko, A. Nugraha, P. N. Andono, J. Jumanto, and E. J. Kusuma, "A systematic review of intelligence video surveillance: Trends, techniques, frameworks, and datasets," *IEEE Access*, vol. 7, no. 1, pp. 457–473, 2019.
[6] P. R. Jeyaraj and E. R. S. Nadar, "Smart-monitor: Patient monitoring system for iot-based healthcare system using deep learning," *IETE Journal of Research*, vol. 68, no. 2, pp. 1435–1442, 2022.
[7] K. S. Kumar, S. Prasad, P. K. Saroj, and R. Tripathi, "Multiple cameras using real time object tracking for surveillance and security system," in *2010 3rd International Conference on Emerging Trends in Engineering and Technology*, 2010, pp. 213–218.
[8] S. Soheli, N. Jahan, M. Hossain, A. Adhikary, A. Khan, and M. Wahiduzzaman, "Smart greenhouse monitoring system using internet of things and artificial intelligence," *Wireless Personal Communications*, vol. 124, no. 4, pp. 3603–3634, 2022.
[9] U. Navalgund and K. Priyadharshini, "Crime intention detection system using deep learning," in *Proceedings of the IEEE International Conference on Circuits and Systems in Digital Enterprise Technology*, ser. ICCSDET '18, 2018, pp. 1–6.
[10] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, ser. ICCV '17, 2017, pp. 2980–2988.
[11] "Openvino: A toolkit for optimizing deep learning models," accessed December 5, 2023. [Online]. Available: https://software.intel.com/content/www/us/en/develop/tools/openvino-toolkit.html
[12] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *Computer Vision*, vol. 1405.0312, no. 1, 2014.
[13] "Kaggle: Machine learning and data science community," accessed December 5, 2023. [Online]. Available: https://www.kaggle.com/
[14] "Google web scraper," accessed December 5, 2023. [Online]. Available: https://chrome.google.com/webstore/detail/web-scraper/jnhgnonknehpejjnehehllkliplmbmhn?hl=en
[15] "Labelimg," accessed December 5, 2023. [Online]. Available: https://viso.ai/computer-vision/labelimg-for-image-annotation/
[16] "Gun violence archive," accessed December 5, 2023. [Online]. Available: https://www.gunviolencearchive.org/