Use of Custom Videogame Dataset and YOLO Model for Accurate Handgun Detection in Real-Time Video Security Applications

Diego Bazan, Raul Casanova and Willy Ugarte[a]
Universidad Peruana de Ciencias Aplicadas, Lima, Peru

Keywords: Video Surveillance Systems, Criminal Activities, Human Limitations, Custom Pistol Video-Game Dataset, YOLOV7, Real Time Detection, Artificial Vision, Machine Learning.

Abstract: Research has shown the ineffectiveness of video surveillance operators in detecting crimes through security cameras, which is a challenge due to their physical limitations. On the other hand, it was shown that computer vision, although promising, faces difficulties in real-time crime detection due to the large amount of data needed to build reliable models. This study presents three key innovations: a gun dataset extracted from the Grand Theft Auto V game, a computer vision model trained on this data, and a video surveillance application that employs the model for automatic gun crime detection. The main challenge was to collect images representing various scenarios and angles to reinforce the computer vision model. The video editor of the Grand Theft Auto V game was used to obtain the necessary images. These images were used to train the model, which was implemented in a desktop application. The results were very promising, as the model demonstrated high accuracy in detecting gun crime in real time. The video surveillance application based on this model was able to automatically identify and alert about criminal situations on security cameras.

1 INTRODUCTION

Video surveillance system operators often perform proactive recognition of criminal situations that could occur in multiple locations simultaneously during a full workday. However, this approach leads to the oversight of real danger situations, which occur sporadically. In a 2002 study[1] , Security Oz Magazine found that after 12 minutes of continuous monitoring, operators missed 45% of on-screen activity, and after 22 minutes, they missed 95%. This study also suggests that operators are expected to review a large number of monitors, which causes their performance in detecting incidents decrease as the number of monitors increases.

On a social level, the safety of citizens and the prevention of criminal acts is of utmost importance, and the use of video surveillance systems is a key tool in achieving this goal. Nevertheless, the human limitations of the operators to detect criminal activities make their surveillance work insufficient. Therefore, there is an urgent need to develop technological solutions that improve the performance of operators in these systems. This is where computer vision and machine learning techniques come into play, offering the possibility of partially automating the detection of criminal activity, thereby reducing the burden on human operators and increasing the efficiency of surveillance systems.

Addressing the challenge of real-time crime detection using computer vision techniques is not an easy task due to several reasons. One of the main difficulties is the high variability and complexity of the real-world scenarios that need to be analyzed. This complexity arises from various factors, such as changes in illumination, occlusions, and variations in the appearance of objects and people due to camera viewpoint, posture, clothing, and accessories. Furthermore, the dynamic and unpredictable nature of criminal events requires fast and accurate detection and tracking of targets across cameras, while minimizing false alarms and processing time.

There are a couple solutions to this problem. For example, the authors (Bhatti et al., 2021) use a model that combines deep learning and object detection al-

[a] https://orcid.org/0000-0002-7510-618X

[1] Wood, J. and Clarke, T. (2006) Practical guidelines for CCTV ergonomics, In: R.N. Pikaar, E.A.P. Koningsveld and P.J.M. Settels (eds.) Proceedings of IEA 2006 Congress, International Ergonomics Association 16th Triennial Congress; 9 – 14 July. Maastricht, The Netherlands: Elsevier, CD-ROM.

Bazan, D., Casanova, R. and Ugarte, W.
Use of Custom Videogame Dataset and YOLO Model for Accurate Handgun Detection in Real-Time Video Security Applications.
DOI: 10.5220/0012716500003690
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 26th International Conference on Enterprise Information Systems (ICEIS 2024) - Volume 1*, pages 520-529
ISBN: 978-989-758-692-7; ISSN: 2184-4992
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

gorithms for weapons detection. They achieved high accuracy in detecting firearms in real-world scenarios and can trigger alarms to alert security personnel in real-time, but they used a custom dataset of real scenarios unlike our approach in which we use a custom dataset from the Grand Theft Auto V video game in order to increase the robustness of our model by training many more detection angles and scenarios. In the other hand, (Ashraf et al., 2022) proposes a solution with CNN and YOLO-V5s, providing high accuracy and robustness to varying conditions. However, doesn't implement the model in a desktop application for video surveillance like our approach to alert when detection occurs.

This paper presents an innovative approach to crime detection through real-time identification of pistols and revolvers using a YoloV7 object detection model trained with a customized dataset based on images from the popular video game Grand Theft Auto V (GTAV). For this purpose, a set of images covering a wide variety of situations and scenarios has been carefully selected, and a manual labeling process has been carried out to obtain an accurate and detailed annotation of the weapons.

In addition, a desktop application has been developed that uses the detection model to alert the user to the presence of pistols and revolvers in real time, allowing for an immediate and effective response to potential dangerous situations. However, it is important to note the limitations of our approach.

First, real-time detection requires a powerful GPU, which may be an impediment to its implementation in some systems.

Second, detection will be performed from a single camera, which limits the monitoring capability in environments with multiple access points.

Third, it is worth mentioning that our model will only detect pistols and revolvers, so other weapons will not be identified.

Finally, the desktop application will only alert security operators using it, which could limit its effectiveness in emergency situations where other entities must also be notified. These limitations are important and should be considered when evaluating the feasibility and effectiveness of the proposed approach.

Our main contributions are:

- We have built a custom dataset from the game Grand Theft Auto V (GTAV) with images of guns and revolvers held at different angles and environments.
- We have trained a computer vision model based on YOLOV7 capable of reliably detecting guns and revolvers from security camera videos with high accuracy.
- We have developed a desktop application capable of reporting, visually and audibly, about the detections made by the model to partially automate the crime detection process by operators.

This paper is divided into the following sections: First, we will analyze works related to armed crime detection through computer vision in Section 2. Then, we will discuss the relevant concept and describe our contribution in more detail in Section 3. In addition, we will explain the procedure and experiments performed in this work in Section 4. Finally, we will discuss the conclusions of our project and indicate recommendations for future work in Section 5.

2 RELATED WORKS

The field of computer vision has been studied for several years in an attempt to discover new techniques that enable computers to understand images by categorizing the objects within them. This variety of techniques has been developed to achieve accurate results in less time and with minimal use of computational resources. In this section, we will mention related works where similar techniques to our proposal are used.

In (Nakkach et al., 2022), the authors propose an abnormal behavior detector based on deep learning through the use of convolutional neural networks (CNN) to alert these behaviors. They use convolutional layers to reduce the size of incoming images and, thus, reduce the use of computational resources. Similarly, they propose an architecture for an smart surveillance system which is responsible for alerting an operator as well as receiving feedback from the operator to update the model. Instead, we use YOLO in its version 7 to achieve high-confidence and fast detection of handguns in use, in order to alert about a possible crime where pistols or revolvers are being used through a video surveillance system only when these types of weapons are used.

In (Bhatti et al., 2021), the authors propose a real-time firearm detection system based on the YOLO object detection model version 4. They tested different computer vision models, such as SSD-MobileNet-v1, YOLOv3, FasterRCNNInceptionResNetv2, and YOLOv4, on three different data sets. After these tests, the dataset containing real-world situations was used to train the YOLOv4 model, as it demonstrated better results. Instead, our work focuses on utilizing the latest version of YOLO to exploit its efficiency in computational resource consumption while maintaining the speed and reliability of detections. Additionally, we train this model with a custom dataset from the Grand Theft Auto V video game created by us to facilitate the collection of gun and revolver images from larger angles and environments.

In (Sung and Park, 2021), the authors built an intelligent surveillance system that allows for active monitoring of security cameras without the need for an operator in charge. They developed this system based on a deep learning model called Fast R-CNN, which has the ability to conduct fast training and testing with good final detections. Instead, our automatic video surveillance detection system uses YOLO as its object detection model, which, unlike traditional region-based models such as Fast R-CNN, examines the entire image and creates bounding boxes with their corresponding probabilities, resulting in a much faster detection model with the same or better accuracy.

In (Ashraf et al., 2022), the authors developed a firearm detection model using YOLO-V5 and CNN algorithms. The model was trained on a dataset consisting of images with and without guns, and its efficacy was evaluated using various metrics such as precision, recall, and F1 score. The model achieved promising results in firearm detection for both images and videos, but it was not tested in a real-time environment, which may have had an impact on how well it worked in actual surveillance scenarios. In contrast to this approach, our system provides an intuitive user interface that enables operators to view detections and receive audiovisual notifications in real-time.

In (Olmos et al., 2018), the authors built an automatic gun detection system in videos based on a Convolutional Neural Network (CNN) model and two detection approaches: the sliding window and the region proposal approach. Various datasets have been used to train the model, and its performance was assessed using a newly created metric called Alarm Activation per Interval (AApI). With this method, videos with guns may be quickly and effectively identified and alerted. However, there may be issues with false positives (Due to the different environmental conditions commonly seen in security cameras, in which the datasets images were taken), and its reaction time is insufficient for real-time applications. Our solution could address these issues by reducing the number of false positives as the model has been trained with images of guns from angles commonly seen in security cameras, and by providing adequate performance for a real-time environment with the YoloV7 model.

3 CONTRIBUTION

3.1 Preliminary Concepts

In this section, the main concepts used in our work are presented. We aim to use deep learning methods, specific to the field of computer vision, in order to propose an application capable of alerting through object detection (handguns) to automate video surveillance.

Neural Networks (Khemani et al., 2024): It consists of a layer of input neurons, one or two layers of hidden neurons and a final layer of output neurons through which information travels to produce resulting values. It is inspired by the sophisticated functionality of the human brain where hundreds of millions of interconnected neurons process information in parallel. Such networks help computers make intelligent decisions without the need for constant human assistance. This is because, after some training, they can learn to find relationships between input and output data that are much more complex than linear computations.

In Figure 1, there is an illustration of a deep neural network, which includes an input layer and an output layer, with multiple hidden layers in between. The strength of the connections between neurons is represented by weight values, and nodes with higher weights have a greater impact on the overall functioning of the network.

Definition 1 (Convolutional Neural Networks (CNN) (Zhou et al., 2024))**.** *They were born in response to the problem that conventional neural networks had when they were used for the recognition of very large images, due to the need to use more and more input data proportional to the pixels of the image.*

This type of networks, also known as CNNs, solved this problem by adding to their architecture layers known as Convolutional Layers, in which filters of small dimensions are used to move along the image, in order to extract important features and reduce its size. These features will be used to train a conventional neural network to obtain the expected

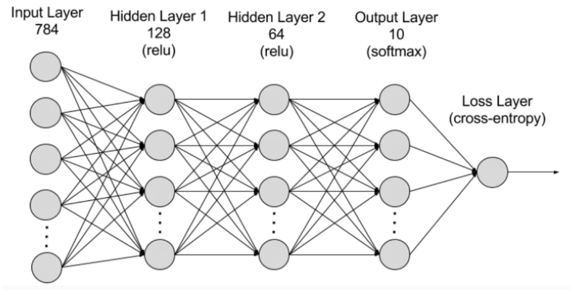


Figure 1: Deep Neural Network (Khemani et al., 2024).

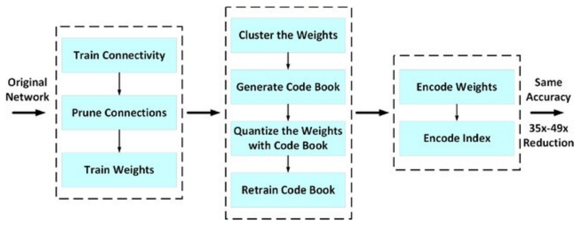


Figure 2: Deep Neural Network (Zhou et al., 2024).

classification results.

We can see an example of a Convolutional Network Arquitecture in Figure 2.

You Only Look Once (Tian et al., 2019): The algorithm called YOLO is an algorithm based on Convolutional Neural Networks, totally oriented to the detection of objects in real time. This is because it has a low computational cost. This algorithm follows a process to perform the detection, first, a matrix of cells is created on the image, to capture objects that exist within them. From the objects that have been detected in the cells, predictions are made to know the size of the object within the image, in order to achieve a complete framing in a bounding box. In Figure 3 we can see the YOLOv3 architecture.

Mean average precision metric (mAP) (b13, 2009): This metric is commonly used to measure accuracy in object detection within the field of computer vision. The process begins with object predictions based on bounding boxes and class labels. For each box, an overlap between the predicted box and the ground truth box is measured using the intersection-over-union (IoU) metric, which is calculated using the formula shown in Figure 4 (a) For each detection task, precision is calculated using the IoU value above a given threshold.

For example, if the threshold is set to 0.5 and the predicted value is 0.7, it will be classified as a True Positive (TP). Otherwise, if it is less than 0.5, it will be classified as a False Positive (FP). The calculation of this precision can be seen in Figure 4 (b). From this point on, with the TP and FP values, the model's precision can be calculated for each class, and the mAP

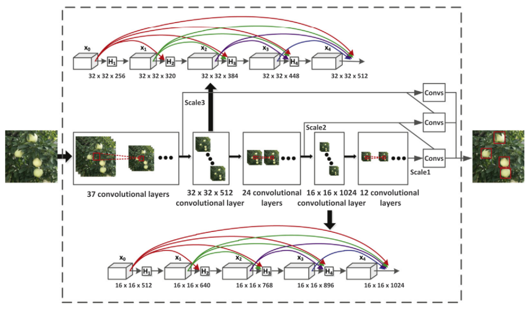
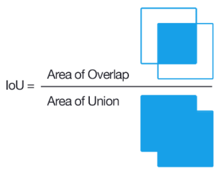


Figure 3: Deep Neural Network (Tian et al., 2019).



$$\text{Precision} = \frac{\textit{True Positive}}{\textit{True Positive} + \textit{False Positive}}$$

$$= \frac{\textit{True Positive}}{\textit{Total Predicted Positive}}$$

(a) (b)

Figure 4: IOU and Precision formulas (b13, 2009).

is the average of all these precisions, which gives an idea of how accurate each model is for object detection in images.

3.2 Method

In this section, the main contributions proposed in this project will be detailed.

3.2.1 Dataset

To choose the dataset with which the model would be trained, a prior review of existing datasets was necessary. The initial dataset, named ”Pistol dataset”, was sourced from the University of Granada[2], chosen for its notable collection of 2,986 images and 3,448 firearm labels. However, upon preliminary review, it was evident that certain images did not depict firearms used in contemporary contexts like the one we can see in Figure 5 (b), including cartoons, as we can see in Figure 5 (a), that were incongruent with the intended purposes of the study. Consequently, an initial version of the dataset was curated for model training.

Following the first training iteration, a decision was made to further refine the dataset by excluding images that deviated from typical camera angles observed in security footage as we can see in Figure 5 (c). Additionally, all standalone firearm images (without being held by a person), like in Figure 5 (d), were removed, as the presence of a grip is essential for criminal activities. This dataset refinement necessitated re-labeling, ensuring that the firearm images included a visible portion of the hand holding the weapon. Subsequently, the model was retrained using this updated dataset.

Upon analyzing the results, it was observed that the model struggled to accurately detect firearms in certain scenarios with varying perspectives and lighting conditions, yielding numerous false positives. To address these limitations, a new dataset had to be created to encompass a broader range of scenarios, in-

[2]F. Herrera Triguero, S. Tabi, A. Castillo Lamas, F. Pérez Hernández, and R. Olmos Pimentel, Weapons detection for security and video surveillance, https://sci2s.ugr.es/weapons-detection (accessed Jun. 24, 2023).

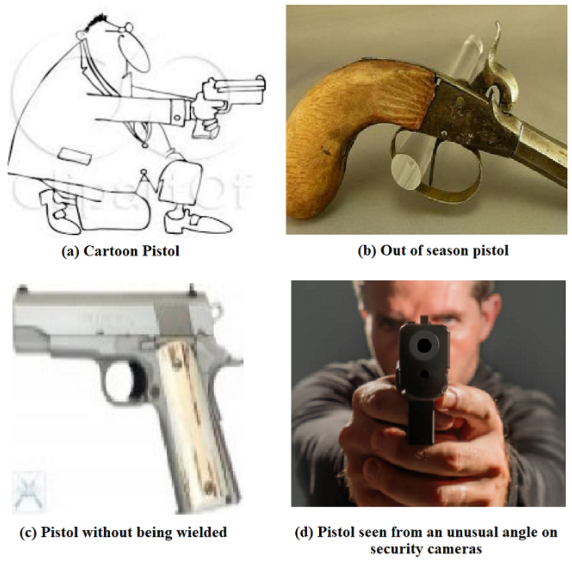

(a) Cartoon Pistol
(b) Out of season pistol
(c) Pistol without being wielded
(d) Pistol seen from an unusual angle on security cameras

Figure 5: ”Pistol dataset” images.

cluding diverse perspectives, firearm types, and lighting conditions. Given the limited availability of real-world photographic environments, a creative and unconventional approach was adopted.

The video game Grand Theft Auto V (GTA V) was chosen as the foundation for constructing this new dataset. Due to its realistic graphics, provided unparalleled freedom to replicate firearm actions and movements, capturing them from different camera angles. In the other hand, the in-game editor proved to be an invaluable tool, simplifying a significant portion of the dataset creation process.

The new dataset comprised four image categories, categorized according to the distance between the firearm and the camera, ranging from closest to furthest. Furthermore, factors such as daytime and nighttime lighting conditions, simulated security camera filters adding noise to the images, and a variety of pistols and revolvers were taken into consideration (all this variety of categories and images can be seen in Figure 6). After collecting the necessary images from the video game, a standardized aspect ratio of 1:1 was applied to ensure compatibility with model training requirements. Subsequently, all 2,300 images were meticulously labeled, resulting in an equal number of corresponding firearm labels.

To further enhance the dataset's diversity, a data augmentation technique employing horizontal mirroring was applied to all images. This approach introduced new perspectives and helped mitigate the risk of model overfitting. Finally, the updated dataset was utilized to retrain the firearm detection model, yielding the final version of the model.

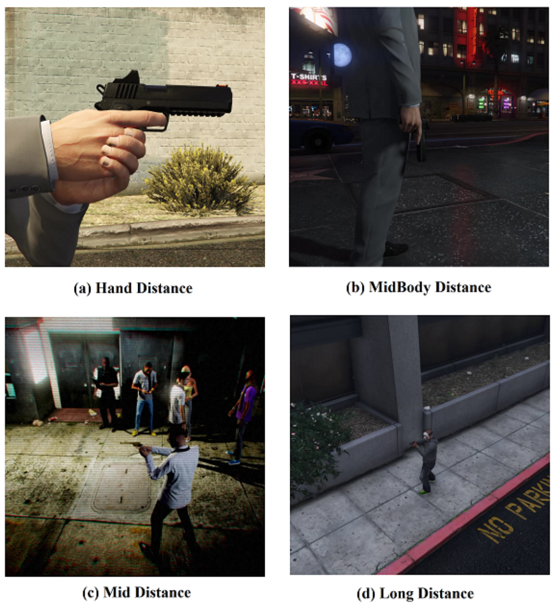


Figure 6: ”GTA V Dataset” Categories.

3.2.2 Fine-Tunning Model

To create the detection model, the Fine-Tuning technique was used by taking the weights obtained from the pre-trained YOLOv7 model. This model was chosen due to its high performance in similar scenarios where real-time response is required.

In the training process, most of the hyperparameters remained the same as those used in pre-training with the MS COCO dataset. However, some values were changed, such as the training batch size, set to 32, and the number of training epochs. This last value varied for the 3 training sessions as they were halted when a loss trend was observed in certain metrics.

3.2.3 Desktop Application

Our latest contribution is the desktop application that will obtain images from a video surveillance camera and process it to identify if a pistol or revolver was detected, in order to alert the operator. In the Figure 7, we will detail the process of how the application receives an image to detect if a crime is occurring with a gun in hand.

- First, the application is able to receive an image from a video source of the user's choice, because it is able to recognize all video devices connected to the computer.
- Second, the application will take care of processing the received image in order to adapt it to the input parameters (640x640 px) that the customized model is able to receive.

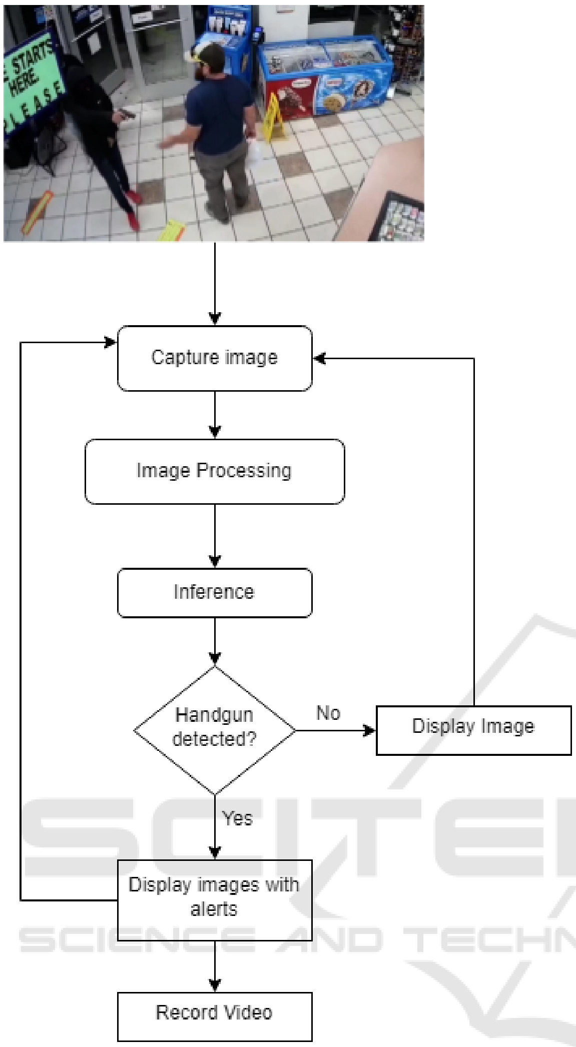
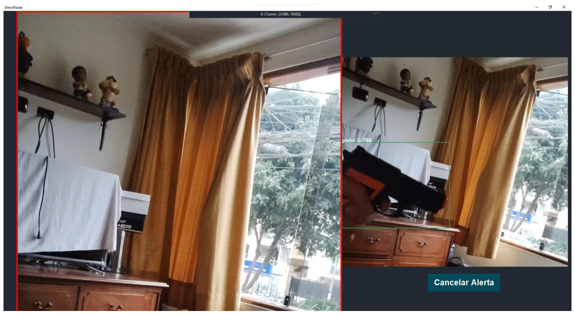


Figure 7: Application flow diagram.

- Third, the application is responsible for sending the processed image to the model to perform the corresponding inference and, in this way, determine the location of possible weapon detections.
- Fourth, when the results are obtained, the application will validate if it is necessary to alert the operator. This it decides based on the detection of a weapon or not.
- In case it has found a weapon, as can be seen in Figure 8a, it displays the real-time image it receives from the camera along with the freeze frame of the first weapon detection with an alert sound, which can be muted at any time. In addition, it records a video of about 10 seconds and, once that time has elapsed, the recording is saved in mp4 format and the freeze frame is no longer displayed.
- On the other hand, if the application did not detect anything, it only shows the image in real time and repeats the flow, as shown in Figure 8b.



(a) Gun detection.

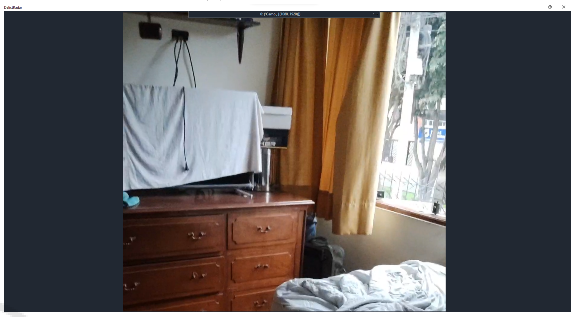

(b) Nothing detected.

Figure 8: Comparison of detection of a gun.

This is how the application can alert the operator and give him the facilities to detect crimes with weapons, in addition to recording these events so that the videos can serve as evidence of what happened.

4 EXPERIMENTS

This section explores the experiments conducted using the YOLO model trained with various datasets to identify pistols and revolvers. Additionally, it examines the tests performed with the desktop application for real-time detections, focusing on the utilization of a toy gun as the subject of study.

4.1 Experimental Protocol

This subsection provides an overview of the experimental environment's configuration, including the local hardware setup and the applications employed during the experiments. The following information outlines the specific details concerning the hardware configuration and software applications used in the experimental setup.

The model experiments have been executed on a computer with an AMD Ryzen 5600X CPU, an NVIDIA RTX 3060 12gb GPU and 16gb of RAM at

Table 1: Inference and frame processing times in seconds according to the computational resource used.

Resource	Inference (seconds)	Frame (seconds)
RTX 2070 super	.03	.05
RTX 3060	.03	.06
GTX 1050	.09	.11
CPU	.30	.40

3600Mhz. The model used for transfer learning is the latest version of the pretrained YOLOv7 model. In addition, to perform the desktop application experiments we also use a desktop computer with an AMD Ryzen 5600X CPU, an NVIDIA RTX 2070 Super 8gb GPU and 16gb of RAM at 3600Mhz, and a laptop with an Intel Core I5 7200U CPU, an NVIDIA GTX 1050 4gb GPU and 8gb of RAM at 2400Mhz.

As for the desktop application, it has been developed in Python 3.10, and the same version is required for deployment. In addition, the experiments were performed consuming 3 different computational resources, as shown in Table 1, and the performance of each of them was measured through computational time in seconds for both inference, as well as for the entire processing of a single frame.

The source code can be accessed through this link: https://github.com/DelictRadar/DesktopApp

4.2 Results

In this subsection, the experiments conducted and the results obtained for both the detection model and the desktop application in general will be detailed.

4.2.1 Model Validation with Metrics

It is necessary to evaluate the performance of the classification model to validate its accuracy and performance in detecting weapons in different scenarios. To complete this task, two different metrics were used, which are listed below:

- Mean Average Precision (mAP): This metric is a measure used to evaluate the performance of classification models used in detection problems with multiple classes or categories. There are 2 variants for calculating mAP.
  - Firstly, there is mAP 0.5, which uses a detection threshold of 0.5. This means that only detections with a confidence score or probability of belonging to a class equal to or greater than 0.5 are considered correct. Any detection below that threshold is considered incorrect.
  - Secondly, mAP 0.5-0.95 is calculated using a range of detection thresholds ranging from 0.5 to 0.95, with increments of 0.05. This implies that the performance of the model is evaluated considering a wider range of confidence thresholds.

  For our comparisons, we will use this latter variant as it provides a more comprehensive measure. Figure 9 shows 3 trainings with different datasets, number of epochs, and the evolution of mAP throughout them. The model with the highest mAP value is considered the best.
- Function Loss: It is a measure that quantifies the discrepancy between the predictions made by a model and the actual values of the training data. If the model is suffering from overfitting, it is likely that the loss function value on the training data will continue to decrease significantly, while the loss function on the validation data starts to increase or plateau at a higher level. In our context, one of the main goals is to minimize this metric during training so that the model can make accurate predictions on new data. We use 2 variants of this metric to make it easier to recognize erroneous behavior in the model.
  - Function Object Loss: This variation is related to the model's ability to correctly detect the presence of objects in an image. If the model has overfitting, it may be learning patterns and specific details from the training data that are not representative of the general population. Figure 11 shows this variation evolution.
  - Function Box Loss: This variation is related to the accuracy in predicting the coordinates of the bounding boxes around the detected objects. In an overfitting scenario, the model may memorize the exact details of the locations and sizes of the bounding boxes in the training data, which can result in lower accuracy in localizing the bounding boxes in the validation data. Figure 10 shows the metric variation evolution trough epochs.

4.2.2 Desktop Application Validation with Metrics

- Execution time: It is the time taken by the processor to solve the instructions of a program. This metric is calculated by subtracting the time obtained before and after executing the process steps. In the particular case of this work, this metric was used to validate the integration between the model and the desktop application in terms of its ability to produce real-time detections.

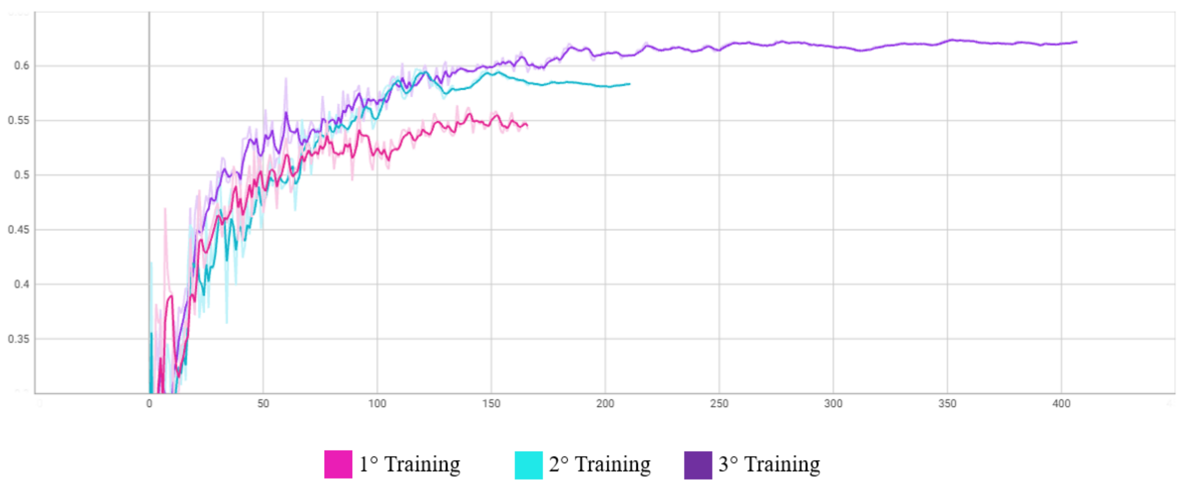


Figure 9: maP metric through epochs in 3 different trainings.

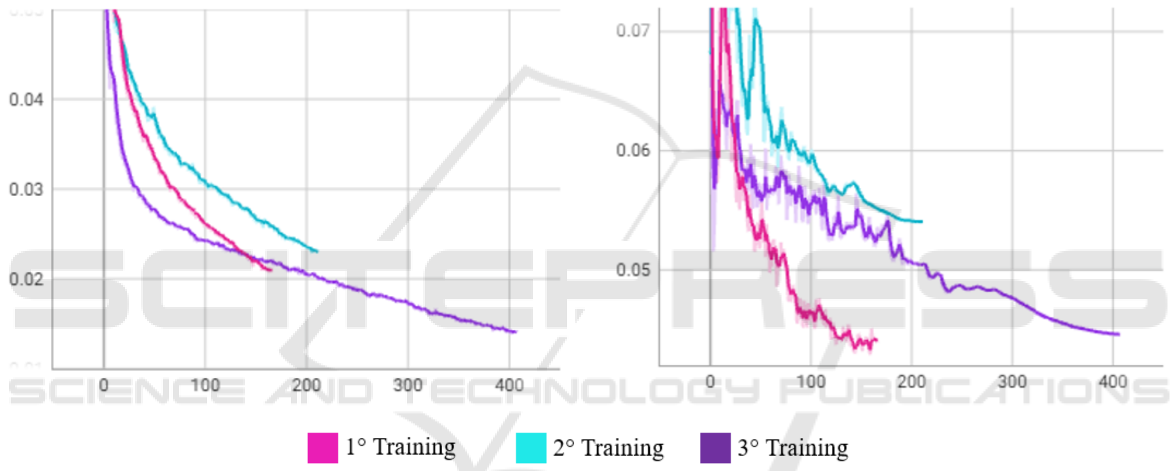


Figure 10: Function Box Loss Training vs Validation.

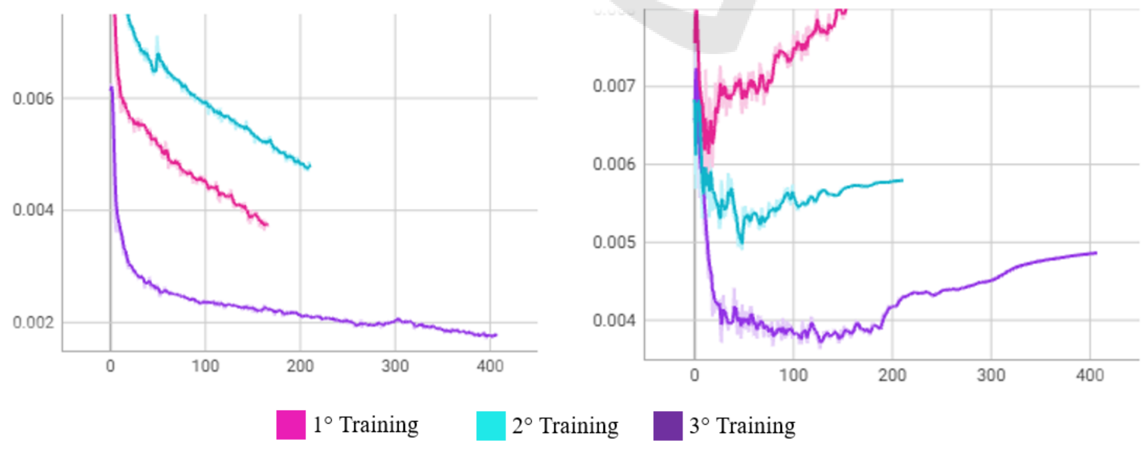


Figure 11: Function Object Loss Training vs Validation.

– To this end, firstly, the model execution time to perform inference was obtained, which in other words would be expressed as the time it takes for the model to detect the weapons in each image provided to it.

– Secondly, the execution time of a frame was calculated, that is, the time the application takes from the moment it receives an image from the security camera until it displays it on the screen.

Both times are very important, especially the first one because it tells us how long it takes to execute the heaviest processing of the whole application.

4.3 Discussion

In this subsection, the results obtained in the previous section are detailed and discussed.

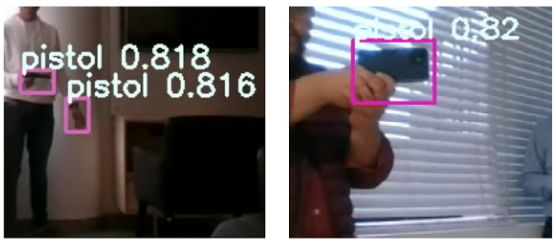
4.3.1 Model Validation with Metrics

As mentioned earlier, three trainings were conducted, and three different metrics were collected for each one. The best model should excel in each of these metrics.

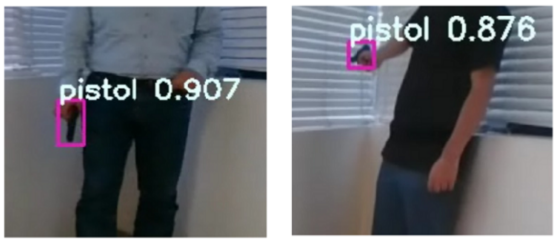
First, we evaluated the mAP metric, which is the most critical in terms of accuracy. According to the visualization in Figure 1, the first training achieved an mAP of 0.563 at epoch 140, the second training achieved a value of 0.597 at epoch 153, and finally, the third training achieved 0.625 at epoch 211. Based on these figures, it can be deduced that for this metric, the best model is the result of the third training.

Now, we evaluate the Function Loss metric for its Box variation. It can be observed in Figure 10 that all three models show a consistent downward trend in both validation and training losses. This indicates that all three trainings have a good fit, so there is no clear winner, at least in this variation of the metric. Next, we evaluate the Function Loss metric in the Object variation. Figure 11 shows that for the first two trainings, there is a decreasing trend in the training data, but the opposite is observed in the validation data. This indicates that both models are not fitting well and are suffering from overfitting. On the other hand, the third training maintains a stable decreasing trend in both cases and shows a slight increase starting from epoch 180. Based on the results of the Function Loss metric, it can be concluded that the best model was the result of the third training. However, it is ideal to use the training from an epoch close to 200, as overfitting behavior started to emerge after this stage.

Regarding the detection model, it can be concluded that the third training yielded the best results, which is why it was chosen to be used in conjunction with the desktop application. However, there is still room for improvement in this model, as there are still false positives, but the accuracy of the detections is sufficient to be a reliable model. The Figure 12 depicts the aforementioned testing.



(a) False Positives



(b) True Positives

Figure 12: Testing with the best model.

4.3.2 Desktop Application Validation with Metrics

Finally, the model inference execution time and the processing time for each frame turned out to be different depending on the type of processing unit used, as shown in Table 1. For the computer using CPU, a time of 0.3 seconds for inference and 0.4 seconds for processing one frame was obtained. On the other hand, for the GTX 1050 GPU, an execution time of 0.09 seconds for inference and 0.11 seconds for frame processing was achieved. Also, for the RTX 3060 GPU, an execution time of 0.03 seconds for inference and 0.06 seconds for frame processing was obtained. Finally, for the RTX 2070 Super GPU, an execution time of 0.02 seconds for inference and 0.06 seconds for frame processing was obtained. In conclusion, as the computing power of the processing unit increases, the application will display images with less delay, so we recommend using higher power graphics cards than the GTX 1050 GPU tested.

5 CONCLUSION

First and foremost, with the use of real-time detection models like YOLOV7, we have been able to confirm in this research that semi-automation of monitoring is feasible. As demonstrated in the studies, a specialized graphics card is required for the intended performance of the model and application, though a high-end card will guarantee superior performance.

On the other hand, we discovered that the accuracy of the model is more dependent on the volume and quality of the data with which it is trained. Thanks to the comparisons between the first dataset found and the dataset constructed from images of a video game, it can be concluded that video games can become a great repository of images that can be used to solve a real-world context problem using detection and classification models. Although the findings were great it is important to note that there are a significant number of false positives. This could be because the model is confused by certain images or there are still unexplored angles of view during the training.

These results allow for the identification of improvements that could be made to the project. Firstly, although the model has shown good performance with a single camera input, no tests were conducted with more than one, so its operation would not be optimal in a large security system. As a result, the next step would be to test the model with multiple inputs. To achieve this, it would be recommended to retrain YOLOV7 models or its most recent version YOLOV8 in their ”tiny” variants, which are used to run models on low-resource computers (Cornejo et al., 2021; Lozano-Mejía et al., 2020). Second, the trials' discovery of false positives shows that the model can be significantly strengthened by being trained on a bigger amount of data (Rodriguez-Meza et al., 2021).

Given that GTA V was launched ten years ago, it is advised to try using more contemporary video games, which offer a comparable selection of weapons and settings and greater graphic quality to further approximate reality, in order to continue practicing with video game graphics. Finally, a good continuation of this project would be to track the movement of the detected armed criminals by leveraging the identification of distinctive features.

REFERENCES

(2009). Mean average precision. In *Encyclopedia of Database Systems*, page 1703. Springer US.

Ashraf, H., Imran, M., Qahtani, A., Alsufyani, A., Almutiry, O., Mahmood, A., Khan, M., and Habib, M. (2022). Weapons detection for security and video surveillance using cnn and yolo-v5s. *Computers, Materials and Continua*, 70:2761–2775.

Bhatti, M. T., Khan, M. G., Aslam, M., and Fiaz, M. J. (2021). Weapon detection in real-time CCTV videos using deep learning. *IEEE Access*, 9:34366–34382.

Cornejo, L., Urbano, R., and Ugarte, W. (2021). Mobile application for controlling a healthy diet in peru using image recognition. In *FRUCT*, pages 32–41. IEEE.

Khemani, B., Patil, S., Kotecha, K., and Tanwar, S. (2024). A review of graph neural networks: concepts, architectures, techniques, challenges, datasets, applications, and future directions. *J. Big Data*, 11(1):18.

Lozano-Mejía, D., Vega-Uribe, E., and Ugarte, W. (2020). Content-based image classification for sheet music books recognition. In *EirCon*, pages 1–4. IEEE.

Nakkach, C., Zrelli, A., and Ezzedine, T. (2022). Smart border surveillance system based on deep learning methods. In *ISNCC*, pages 1–6. IEEE.

Olmos, R., Tabik, S., and Herrera, F. (2018). Automatic handgun detection alarm in videos using deep learning. *Neurocomputing*, 275:66–72.

Rodriguez-Meza, B., Vargas-Lopez-Lavalle, R., and Ugarte, W. (2021). Recurrent neural networks for deception detection in videos. In *ICAT*, pages 397–411. Springer.

Sung, C. and Park, J. (2021). Design of an intelligent video surveillance system for crime prevention: applying deep learning technology. *Multim. Tools Appl.*, 80(26):34297–34309.

Tian, Y., Yang, G., Wang, Z., Wang, H., Li, E., and Liang, Z. (2019). Apple detection during different growth stages in orchards using the improved YOLO-V3 model. *Comput. Electron. Agric.*, 157:417–426.

Zhou, Y., Xia, L., Zhao, J., Yao, R., and Liu, B. (2024). Efficient convolutional neural networks and network compression methods for object detection: a survey. *Multim. Tools Appl.*, 83(4):10167–10209.