International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 14, No.1 (Aug-2023)

http://dx.doi.org/10.12785/ijcds/140141

Deep Learning-Based Real-Time Weapon Detection System

Amjed Al-Mousa[1], Omar Z. Alzaibaq[1] and Yazan K. Abu Hashyeh[2]

[1]*Computer Engineering Department, PSUT, Amman, Jordan*
[2]*Electrical Engineering Department, PSUT, Amman, Jordan*

Received 4 Feb. 2023, Revised 18 Mar. 2023, Accepted 06 May. 2023, Published 01 Aug. 2023

Abstract: In recent years, the rate of gun violence has risen at a rapid pace. Most current security systems rely on human personnel to monitor lobbies and halls constantly. With the advancement of machine learning and, specifically, deep learning techniques, future closed-circuit TV (CCTV) and security systems should be able to detect threats and act upon this detection when needed.
This paper presents a security system architecture that uses deep learning and image-processing techniques for real-time weapon detection. The system relies on processing a video feed to detect people carrying different types of weapons by periodically capturing images from the video feed. These images are fed to a convolutional neural network (CNN). The CNN then decides if the image contains a threat or not. If it is a threat, it would alert the security guards on a mobile application and send them an image of the situation. The system was tested and achieved a testing accuracy of 92.5%. Also, it was able to complete the detection in as fast as 1.6 seconds.

Keywords: CNN, Security Cameras, Weapon Detection, CCTV, Deep Learning

1. Introduction

The threat of mass shootings and robberies has been rampant in recent years. Moreover, the gun violence crime rates are very high, especially in countries where carrying firearms is legal. In the US, the total number of gun violence-related deaths in 2022 from January until the 15th of September is 31,499 [1]. This is in addition to 28,194 injuries, a massive number that cannot be ignored. Therefore, advanced modern security systems are essential to limit this problem.
One of the most widely used security systems is CCTV cameras. CCTV is a type of situational crime prevention (SCP) strategy in which levels of formal surveillance are increased within a target area. SCP is focused on preventing crime by reducing the number of criminal opportunities [2], [3]. These systems rely on a surveillance operator monitoring multiple screens simultaneously, and while waiting for those unlikely-to-happen threats, they become less focused, and detecting threats becomes harder [4]. Therefore, these systems are not very efficient and are economically challenging because of the running costs of the surveillance operators.

Artificial intelligence (AI) and machine learning (ML) techniques have been used successfully to solve many medical, technological, educational, and financial challenges [5], [6], [7], [8], [9]. Deep neural networks (DNNs) have also gained even more traction in image-based problems [10], [11]. They can also be used to make surveillance systems able to detect weaponized personnel and alert the proper authorities immediately [12].

The proposed architecture will impact the security surveillance field in various aspects as it integrates traditional technology with the thriving areas of AI, ML, and the internet of things (IoT), [13], [14]. The most critical expected impacts can be listed as follows:

- Attacks are prevented before happening
- Less successful robberies
- Fewer casualties in incidents involving weapons and shooting
- Security and surveillance running costs are reduced

This work's core contribution is showing the ability of DNNs in detecting weapons by developing a DNN architecture and training it by creating a massive dataset of images of people. Some people would be entering a lobby without any weapons *Negative*, while others enter the lobby carrying weapons; these will be tagged as *Positive*. These tagged images are then used to train a DNN that is capable of classifying any image as a threat (*Positive*) or not (*Negative*). Once the data is collected and the DNN is trained, the system can be deployed in production. The proposed deployment can use existing indoor cameras to

E-mail address: a.almousa@psut.edu.jo, omar.alzaibaq@gmail.com, yazankheiri@gmail.com

monitor people entering a building lobby. The camera(s) should be installed facing the entrance area providing real-time images. Once the system detects the threat, it automatically notifies the security guards in the building on their mobile phones. The security guards receive an image of what is happening in that building area and respond accordingly. It is worth noting that this work does not extend to concealed weapons [15], nor the detection of suspects based on worrying expressions or unnatural behavior [16]. It is also not concerned with predicting crimes or robberies before they happen [17].

To summarize the objectives of the paper are as follows:

- Establish the ability of CNNs in detecting handheld weapons using CCTV feeds.
- Identify the proper image pre-processing steps needed to facilitate the detection process.
- Achieve reasonable performance metrics using the proposed architecture.

The rest of the paper is organized as follows: Section 2 provides an overview of the literature and related work. Section 3 details the method used to solve the threat detection problem, including the details of the DNN model used. The results of the system testing are discussed in section 4. Section 5 presents the proposed system's limitations and practical considerations. Finally, section 6 presents the work's conclusion and future research directions.

2. Related Work and Literature Review

This section summarizes a few papers that used different approaches in automatic weapon detection using deep learning and image processing.

The first approach uses background reduction to remove static objects by using a reference image of the place, then compares it to a present image of the same place to eliminate similarities [18]. By using Canny edge detection, the obtained image will be a silhouette. After that, the model uses a sliding window, MPEG-7 feature extraction, and support vector machine (SVM) to decide whether to send an alert. The sliding window size is determined by trial and error after installing the system, which means the system needs to be adjusted with different sliding window sizes depending on installation.

The second approach uses deep learning for automatic handgun detection [19]. It compares the sliding window results and Regional Convolutional Neural Networks (R-CNN). After using The Histogram of Oriented Gradients (HOG) descriptor for feature extraction and then using a sliding window, the processing speed for detecting pedestrians is 14 s/image which is not very practical for real-time applications. This is in addition to the large computational power needed for the sizeable neural network. The real challenge is to create a method that dynamically optimizes all of the CNN's parameters simultaneously [20].

The third approach by Verma and Dhillon uses Fast R-CNN, and deep learning [21]. A VGG-16-based classification model (16 convolutional layers) is used, which focuses on prediction loss minimization. Its methods are the same as the previous approaches, but with a minor difference of having a fixed input size of 224p x 224p RGB images.

The fourth approach [22] uses the same basics mentioned above (sliding window, HOG feature extraction, SIFT, and Harris key point detection) with unipolar sigmoid and bipolar sigmoid as activation functions. The idea is to detect the humans first and then check whether there is a weapon in the picture. Notice that detecting humans using background reduction is faster than HOG, which is essential in real-time applications. However, for background reduction to be efficient, the camera must be installed indoors because the background subtraction method is not flexible and can be affected by slight changes in light intensity or object occlusions.

The fifth approach by Lai and Maples tried various pre-trained models such as VGG-16 mentioned earlier, Overfeat1, Overfeat2, and Overfeat3 [23]. The problem the paper is trying to address is mainly detecting in real-time, and despite the significant accuracy in GoogleNet Overfeat, it needed 14s per image, which is impractical. This paper's final results of reasonably high accuracy and a satisfactory classification time of 1.3 seconds are achieved using Overfeat-3 with tuned hyper-parameters.

The sixth approach proposes a detection model (ResNet50V2) trained on the Open Images V6 dataset to detect the visual relationships of "holds" and "wears" between people and objects [24].

The seventh approach utilizes the transfer learning concept to train the VGGNet architecture based on VGGNet-16 weights [25]. While it achieved high accuracy, the dataset used is for stand-alone weapons. Another approach based on the VGGNet architecture classified seven classes of weapons and achieved an accuracy of 98.4 which was higher than other basic models like VGG16 and Resnet. The training was done on a small dataset from the internet containing just over five thousand images. However, the images trained only contained the weapon with no other object. In addition, there was no mention of the time required to process each image [26].

In[27], the main focus point was cold steel weapons. It tackled the challenge formed by the light reflection on this type of weapon by utilizing different regional proposal search algorithms. And using Dacolt darkening and contrast to handle different brightness and lighting conditions.

Based on the summary of the previous work, it is clear that there is a gap in finding a method that is:

- Able to detect weapons in near real-time.
- Computationally light.
- Tested on more than one type of weapon.

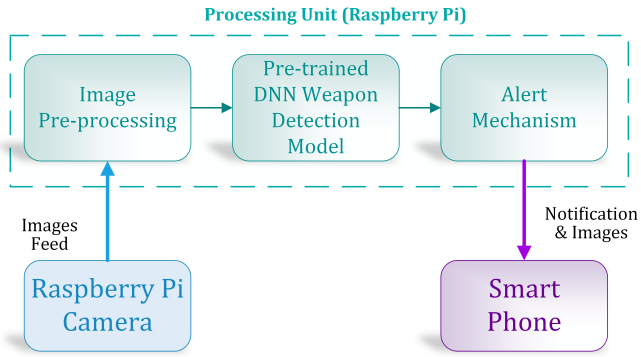


Figure 1. High-level block diagram of the proposed system

- Tested in a real environment with noisy image background.
- Auto-configured and does not need lots of calibration.

3. Methodology

Figure 1 shows a high-level block diagram of how the system is expected to operate. The Raspberry Pi Camera captures the real-time video feed and feeds it to the processing unit. The processing unit performs the image processing steps required and then feeds the processed images to the pre-trained DNN. The DNN was trained using a large number of images that were collected and properly labeled previously. The trained DNN acts as a classifier for the incoming image, whether it contains a weapon. If the DNN is classified as *Positive*, i.e., contains a weapon, the system alerts the security guards on their smartphones and sends them an image of the current situation. Otherwise, the image is dropped, and the system proceeds to process a new frame. The details of the system components are elaborated on in the following sections.

A. Experimental Setup & Dataset Collection

Data is the most critical part of building any deep-learning solution. Such image-based applications require a large amount of data (images). Data is collected in the Innovation Lab at Princess Sumaya University for Technology (PSUT). Raspberry Pi B+ (An ARM Cortex-A53 1.4 GHz processor with WiFi capability and 1GB of SRAM) is used with a connected camera set on a table 1.8 meters high. The camera faces the lab entrance, which is roughly 4.5 meters away. The Raspberry Pi is connected to a display to monitor the process. A live log of the time required to process and classify each image and the classification results are displayed.

The chosen area to collect the data is very active, with many students entering and exiting during the day, which helps collect data for training and testing the model. Students' pictures were taken with their permission. In addition, students were given three types of handgun replicas to mimic an attack. Students were given little restriction on how to hold the gun replicas. Some students pointed them out, some held them on their side, while others concealed or semi-concealed the guns. The images are captured with a resolution of 1920p x 1080, a resolution that can be scaled down if it turns out that a lower resolution might be more suitable. The total size of the dataset is around 14k. However, data augmentation techniques, such as scaling, rotation, and reflection, are used to get more images. If an image and a horizontally reflected replica of that image enter a neural network, it will process them as two completely different examples. Thus, from the captured images, twice the original dataset size is achieved, around 28k as in Table I. Note that the target here is to detect whether there is a gun or not, not detect the gun replica type.

TABLE I. Dataset classes distribution

Tag	Count	Percentage
Positive	10k	36%
Negative	18k	64%

A core problem in such datasets is that the target, in most cases, represents a small object in the image, making it harder to classify an image as a negative because any shape close to the shape or the color intensity of guns might be classified as a positive. Figure 2 shows positive class examples on the top and negative class examples on the bottom.

B. Image Processing

For the DNN to focus its learning on the important aspects of the image and not on static objects in the lobby. An empty reference image is captured as background and continuously subtracted from new images being taken. Thus, static objects like furniture will be removed from the image. Initially, images were loaded with the resolution (1920p x 1080p), which was slow to process. Therefore, the images are resized to (960p x 540p). Figure 3 summarizes the pre-processing steps applied to each input image.

The first stage is image smoothing. Gaussian Blur removes the salt and pepper noise. Figure 4 shows a comparison between the image without smoothing and with smoothing using both median and Gaussian blur. Gaussian blur is chosen. If one chooses not to blur the images as in Figure 4a, some unwanted edges of the door and other components remain. Using Gaussian blur is shown in Figure 4c, where all unwanted edges are filtered out. Meanwhile, the median blur is shown in Figure 4d. Here some noise is filtered out, but the edges of the door remain because median blur keeps the edges. After the mask is generated, the reference image is subtracted, and a binary threshold is applied. Furthermore, morphological transformations are needed to filter noise after the threshold. The opening is used, which performs erosion, then dilation. Iterating this procedure several

Figure 2. Samples from the dataset

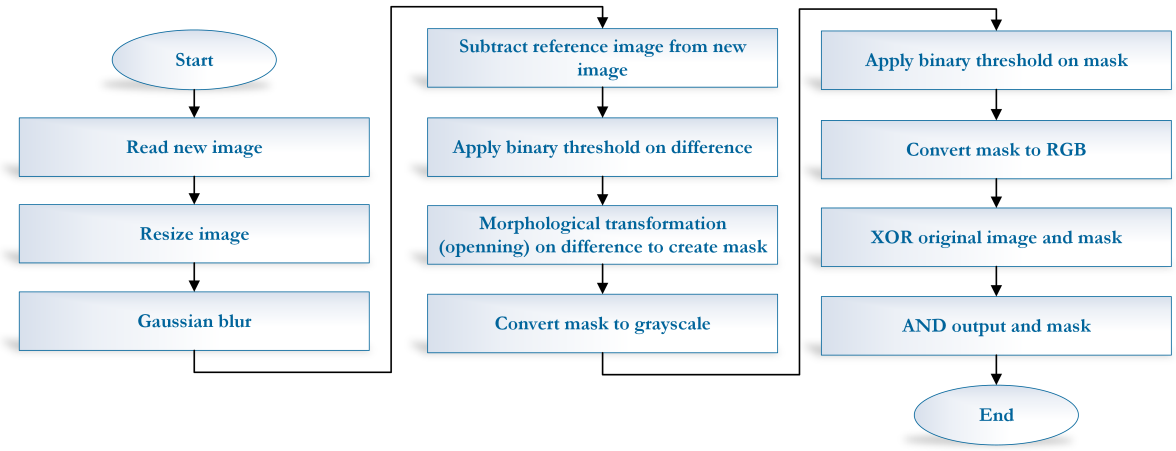


Figure 3. Flowchart presents a summary of image pre-processing steps

times removed most of the image's noise. The kernel size selected is (5, 5), and the number of iterations is two. Figure 5 shows the results of using too many iterations on a small image. Figure 5a shows the output after two iterations, while Figure 5b shows the output after six iterations. Figure 5b shows less noise; however, the critical information, which is the firearm held by the person, is cropped because of the excessive number of iterations.

Processing images in grayscale instead of colored images increases the speed, so the image is converted to grayscale, and another binary threshold is applied. The most effective way to apply the mask is using bitwise logical operations. Bitwise XOR is used, which would result in inverting the foreground. Then a bitwise AND is used to apply the mask to the original image. Figure 6 shows the final output after performing image processing steps. Figure 6a has too many features, which might confuse the model, increase the time needed for model training, and negatively affect the accuracy. In contrast, Figure 6b with the background eliminated and the human with or without the weapon is visible, the desired features appear to be more evident than those in Figure 6a, and the system's accuracy would be higher.

C. *The CNN Architecture*

The input layer is the input interface of the neural network where input images are loaded. Because of the GPU memory limit and relatively small dataset size, a resolution

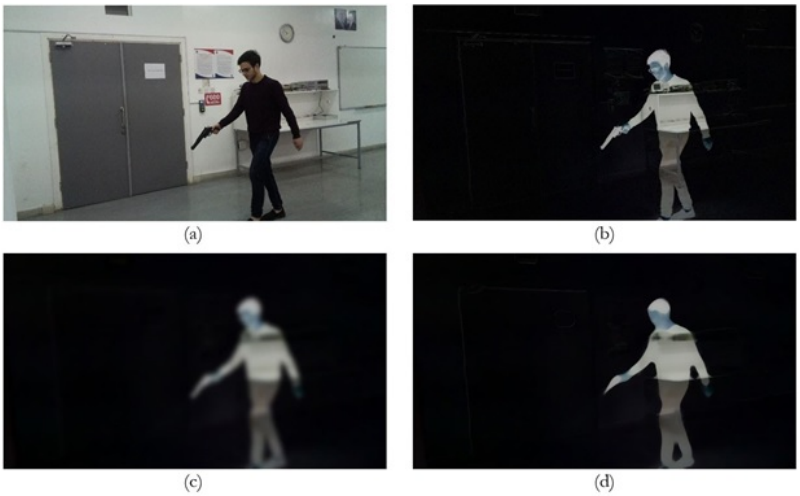
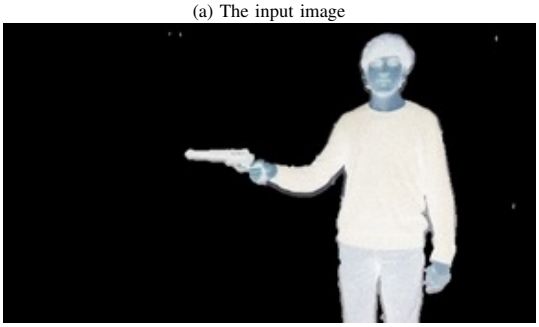


Figure 4. Applying blur. (a) Original image. (b) Subtracted without blurring. (c) Gaussian blur. (d) Median blur

Figure 5. Applying opening. (a) using two iterations. (b) using six iterations

(a) The input image



(b) Background subtracted image

Figure 6. Background subtraction

of 640p x 360p (aspect ratio of 16:9) is chosen. Grayscale images are used for training rather than colored images, as the image colors do not add much useful information and are simpler to process. Accordingly, the dimensions of the input layer are (360, 640, 1).

After referring to [28] and experimenting with setting the number of the hidden layers, three convolution hidden layers are chosen with the ReLU activation function. The number of input variables is the number of pixels, which is 230,400. Based on [28], and [29], the number of feature maps in the hidden layers is 64, 128, and 256 feature maps, respectively. Flattening is applied to the third hidden layer so a fully connected layer can be added afterward with a width of 2048 and a ReLU activation function.

After each of the three hidden layers, a pooling layer is applied to the previous layer's output. The pooling function used is max-pooling as weapons appear as the brightest objects in the pre-processed images as they are black in the original images, and bright pixels carry higher pixel values than dark pixels in the grayscale format, so when max-pooling is applied, bright objects will be highlighted, and dark objects will vanish as in Figure 7. Average pooling does not highlight the desired features, but it is still better than min-pooling, where white objects fade away. The size of the max-pooling window applied after each hidden layer is 3x3.

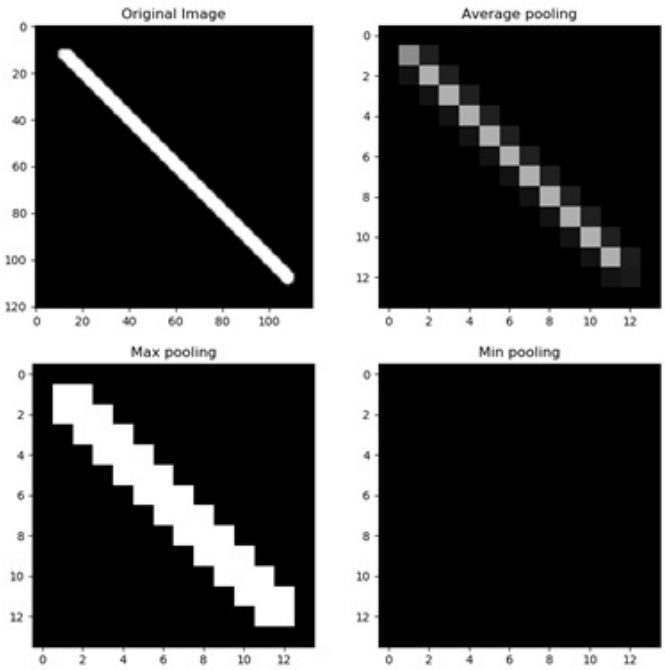


Figure 7. A comparison between different pooling functions [30]

TABLE II. Detailed CNN architecture

Layer #	Layer (Type)	Output Shape	Params #
1	InputLayer	640 x 360 x 1	0
2	Conv2D	638 x 358 x 64	640
3	MaxPooling2D	212 x 119 x 64	0
4	Conv2D	210 x 117 x 128	73856
5	MaxPooling2D	70 x 39 x 128	0
6	Conv2D	68 x 37 x 256	295168
7	MaxPooling2D	22 x 12 x 256	0
8	Flatten	67584	0
9	Dense	2048	138414080
10	Dense	1	2049

Finally, the output layer contains only one neuron because the problem here is a binary classification problem with two possible outputs that can be carried on a single neuron. The model structure is tabulated in Table II. The optimization algorithm for minimizing the cost function is mini-batch gradient descent; it avoids the overshooting resulting from the high variance in the stochastic gradient descent and guarantees convergence to the global minimum, achieved in batch gradient descent[31]. The cost function used is binary cross-entropy, and the learning rate has been manually tweaked, starting from a typical value of around 0.01 for multi-layer networks [32], concluding that the optimal learning performance has been observed at a learning rate of 0.1.

A high-end virtual machine (VM) instance is customized on Google cloud platform (GCP) to perform the training of the DNN. The summary of its specifications is as follows:

- 2 vCPUs with a memory of 13GB.
- NVIDIA Tesla K80 with a memory of 12GB.

D. Alert Notification System

The Raspberry Pi and the mobile application are connected using Wi-Fi technology in a local network. If the system detects a threat, a push notification is sent to the mobile if the application is not open to alert the user. The application and the server are connected to Firebase. Firebase acts like the middleman between the client and the server, making it easier for the server to handle multiple clients simultaneously. The notification and image are sent to all the connected devices simultaneously. When the client connects to the server, the server loads the resized captured image, converts it into a byte array, and sends it. A smaller version of the image is used here to send the image faster.

4. Results and Discussion

The results shown below will cover three architectures. The first is the baseline architecture. While the second investigates the use of drop-outs. Finally, the third architecture uses an additional layer to max-pool the inputs.

A. Baseline Architecture

The first experiment uses the dataset without pre-processing, where the collected data is used directly to train the model. Here the training and testing accuracy is between 57.8% - 62.5%, which is not adequate for the application to be successful. Thus, this approach is dropped, and pre-processing steps are applied to eliminate unneeded information and increase the training and testing accuracy. The dataset is shuffled and split into 90% for training and 10% for validation. Before splitting, 400 examples are randomly removed from the dataset and stored in a separate folder for testing purposes. Although testing on such a small test dataset might not provide precise results, it is chosen to be small to leave as much data for training as possible. Batches of 32 labeled images are used to train the CNN. After training the CNN designed earlier for ten epochs, results are plotted as shown in Figure 8 showing the evolution of the training accuracy, and the validation accuracy as the number of epochs (iterations) increases. The training accuracy seems to increase much faster than the testing accuracy; however, the model appears to continue learning, and the accuracy is improving.

After the tenth epoch, the model is tested on the testing dataset (400 samples, 200 per class). The testing results are shown in Table III. Accuracy, recall, precision, and f1-score are calculated using Equations 1-4:

$$Accuracy = \frac{True\ Positive + True\ Negative}{Test\ Dataset\ Size} \quad (1)$$

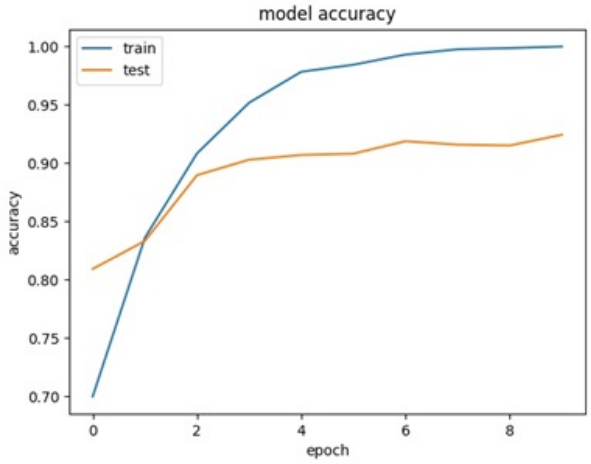


Figure 8. Training accuracy and validation accuracy for baseline architecture

TABLE III. Performance metrics for baseline architecture

Metric	Value
Validation Accuracy	92.4%
True Negative	187 out of 200
True Positive	183 out of 200
Accuracy	92.5%
Precision	93.4%
Recall	91.5%
F1-score	92.4%
Pre-Processing Time	0.3 s
Classification Time	2.0 s

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \quad (2)$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \quad (3)$$

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall} \quad (4)$$

Lower-resolution images (640p x 360p) drastically decreased the time required to pre-process and classify each image compared to the full-size images. The time decreased from 9.5 seconds to 2.3 seconds, divided as follows: the average pre-processing time is 0.3 seconds, and the average classification time is 2 seconds. This also decreased the time required to send the image, as the time required to encode, transmit, and decode the image was also drastically reduced.

B. *Dropout Architecture*

To reduce the gap between the training and validation accuracy, 'dropout' is added after each of the hidden layers. What dropout does is that it 'drops out' random neurons at a specific rate as a regularization technique to reduce overfitting. As a result, more training epochs are required to reach the same training accuracy because some neurons are dropped during training, but each epoch will take a shorter time. The difference between the validation and the training accuracy has decreased during the first few epochs, as shown in Figure 9. However, results have not improved, as the validation accuracy has not exceeded 91.9%. One might notice that stopping learning at the fifth epoch might be better than continuing until the twelfth epoch, as more overfitting occurred while not significantly increasing the validation accuracy. This technique is one of the regularization techniques used in machine learning, and it is called 'early stopping'.

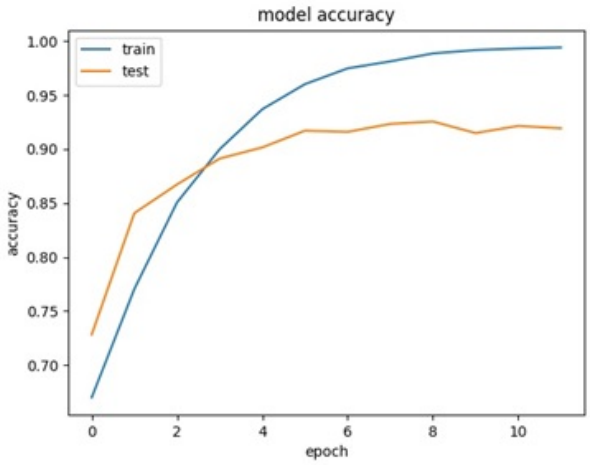


Figure 9. Training accuracy and validation accuracy for dropout architecture

C. *Max-pooling Inputs Architecture*

A max-pooling function of 2x2 window size is applied after the input layer and before the hidden layers to highlight the desired features as a pre-processing step before training. This enabled the increase of the batch size to 64 and has shown significantly faster learning and validation performance. The increase in the batch size can be justified by the resolution reduction (the image entering the CNN now has a quarter of the number of pixels in the original image), which significantly reduced the trainable parameters from **139 million** to **76 million**. It has also given similar validation and testing results: validation accuracy of 91.8% and test accuracy of 92.5%, as shown in Table IV. Although this model has a slightly lower validation accuracy, it is highly preferred over the last two architectures because of its faster classification response. The average classification time is now 1.3 seconds, down from 2.0 seconds in the baseline architecture. These times were measured by embedding

TABLE IV. Performance metrics for max-pooling inputs architecture

Metric	Value
Validation Accuracy	91.8%
True Negative	185 out of 200
True Positive	185 out of 200
Accuracy	92.5%
Precision	92.5%
Recall	92.5%
F1-score	92.5%
Pre-Processing Time	0.3 s
Classification Time	1.3 s

timestamps within the code, thus accurately measuring the time from capturing the image till the classification is decided.

5. Limitations & Practical Considerations

While a fully functional demonstration has been presented in section 4, there are several factors to be considered if such a system is to be deployed in a real environment:

- Lighting: The system should be trained and tested to assess the impact of lighting on how resilient the CNN architecture is to lighting changes.
- Weapon Type: The current dataset was created using three types of handguns. The dataset can be expanded to include other types of weapons, like knives, rifles, and semi-automatic guns.
- Camera Positioning: The results presented are based on a fixed position selected to allow a frontal view of subjects. Further experiments with other positions should be considered to analyze the impact of the position on the system's accuracy.
- Image Quality & Specifications: While the system had to down-sample the images from the camera feed, images with different resolutions, aspect ratios, and those taken with non-standard lenses should be analyzed.

6. Conclusion & Future Work

The main goal of this research was to develop an effective architecture that can detect different types of weapons in building entrances and alert security guards of any incidents. This was achieved by tapping into an image feed and processing captured images through a pipeline of image pre-processing steps that reduce noise, reduce size, and highlight the valuable information in the image. Processed images are later fed to a CNN. The CNN was trained using more than 28K labeled images. The baseline system achieved a 92.5% accuracy, 93.4% precision, 91.5% recall, and 92.4% f1-score. All of these numbers have changed to 92.5% with the max-pooling architecture. Also, it processed and classified an image within 1.6 s. Thus, eliminating the need for continuous human monitoring and eliminating the chances of human errors.

The current system was trained and tested on a reasonably practical setup. However, future work can be expanded to analyze the limitations presented in section 5. In addition, other new deep architectures, like fully convolutional neural networks, should also be evaluated. Such architectures might enhance the detection accuracy while reducing the required computations.

References

[1] GunViolenceArchive, "Gun violence archive.org," 2022. [Online]. Available: https://www.gunviolencearchive.org

[2] E. L. Piza, B. C. Welsh, D. P. Farrington, and A. L. Thomas, "Cctv surveillance for crime prevention, a 40-year systematic review with meta-analysis," *Criminology & Public Policy*, vol. 18, no. 1, pp. 135–159, 2019.

[3] K. L. Ritchie, D. White, R. S. S. Kramer, E. Noyes, R. Jenkins, and A. M. Burton, "Enhancing cctv: Averages improve face identification from poor-quality images," *Applied Cognitive Psychology*, vol. 32, no. 6, pp. 671–680, 2018.

[4] R. Nunes-Vaz and S. Lord, "Designing physical security for complex infrastructures," *International Journal of Critical Infrastructure Protection*, vol. 7, no. 3, pp. 178–192, 2014.

[5] H. Al-Zubaidi, M. Dweik, and A. Al-Mousa, "Stroke prediction using machine learning classification methods," in *2022 International Arab Conference on Information Technology (ACIT)*, 2022, pp. 1–8.

[6] M. Atari and A. Al-Mousa, "A machine-learning based approach for detecting phishing urls," in *2022 International Conference on Intelligent Data Science Technologies and Applications (IDSTA)*, 2022, pp. 82–88.

[7] Z. Bitar and A. Al-Mousa, "Prediction of graduate admission using multiple supervised machine learning models," in *2020 SoutheastCon*, 2020, pp. 1–6.

[8] S. Khalifeh and A. A. Al-Mousa, "A book recommender system using collaborative filtering method," in *International Conference on Data Science, E-Learning and Information Systems 2021*, ser. DATA'21. New York, NY, USA: Association for Computing Machinery, 2021, p. 131–135.

[9] A. Atwah and A. Al-Mousa, "Car accident severity classification using machine learning," in *2021 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, 2021, pp. 186–192.

[10] A. J. Moshayedi, A. S. Roy, A. Kolahdooz, and Y. Shuxin, "Deep learning application pros and cons over algorithm," *EAI Endorsed Transactions on AI and Robotics*, vol. 1, no. 1, p. e7, Feb. 2022. [Online]. Available: https://publications.eai.eu/index.php/airo/article/view/19

[11] A. J. Moshayedi, A. S. Khan, S. Yang, and S. M. Zanjani, "Personal image classifier based handy pipe defect recognizer (hpd): Design and test," in *2022 7th International Conference on Intelligent Computing and Signal Processing (ICSP)*, 2022, pp. 1721–1728.

[12] A. J. Moshayedi, A. S. Roy, A. Taravet, L. Liao, J. Wu, and M. Gheisari, "A secure traffic police remote sensing approach via a deep learning-based low-altitude vehicle speed detector through uavs in smart cites: Algorithm, implementation and evaluation,"

Future Transportation, vol. 3, no. 1, pp. 189–209, 2023. [Online]. Available: https://www.mdpi.com/2673-7590/3/1/12

[13] Z. Ullah, F. Al-Turjman, L. Mostarda, and R. Gagliardi, "Applications of artificial intelligence and machine learning in smart cities," *Computer Communications*, vol. 154, pp. 313–323, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366419320821

[14] Z. Sabeur, C. M. Angelopoulos, L. Collick, N. Chechina, D. Cetinkaya, and A. Bruno, "Advanced cyber and physical situation awareness in urban smart spaces," in *Advances in Neuroergonomics and Cognitive Engineering*, H. Ayaz, U. Asgher, and L. Paletta, Eds. Cham: Springer International Publishing, 2021, pp. 428–441.

[15] M. Parande and S. Soma, "Concealed weapon detection in a human body by infrared imaging," *International Journal of Science and Research (IJSR)*, vol. 4, pp. 182–188, 2015.

[16] H. Bouma, J. van Rest, K. van Buul-Besseling, J. de Jong, and A. Havekes, "Integrated roadmap for the rapid finding and tracking of people at large airports," *International Journal of Critical Infrastructure Protection*, vol. 12, pp. 61–74, 2016.

[17] M. P. de la Cruz López, J. J. Cartelle Barros, A. del Caño Gochi, M. C. Garaboa Fernández, and J. Blanco Leis, "Assessing the risk of robbery in bank branches to reduce impact on personnel," *Risk Analysis*, vol. n/a, no. n/a, 2021.

[18] M. Grega, A. Matiolański, P. Guzik, and M. Leszczuk, "Automated detection of firearms and knives in a cctv image," *Sensors*, vol. 16, no. 1, 2016.

[19] R. Olmos, S. Tabik, and F. Herrera, "Automatic handgun detection alarm in videos using deep learning," *Neurocomputing*, vol. 275, pp. 66–72, 2018.

[20] M. S. Al-Daweri, S. Abdullah, and K. A. Z. Ariffin, "A homogeneous ensemble based dynamic artificial neural network for solving the intrusion detection problem," *International Journal of Critical Infrastructure Protection*, vol. 34, p. 100449, 2021.

[21] G. K. Verma and A. Dhillon, "A handheld gun detection using faster r-cnn deep learning," in *Proceedings of the 7th International Conference on Computer and Communication Technology*, ser. ICCCT-2017. New York, NY, USA: Association for Computing Machinery, 2017, p. 84–88.

[22] R. Vajhala, R. Maddineni, and P. R. Yeruva, "Weapon detection in surveillance camera images," Master's thesis, , Department of Applied Signal Processing, 2016.

[23] J. Lai and S. Maples, "Developing a real-time gun detection classifier," http://cs231n.stanford.edu/reports/2017/pdfs/716.pdf, 2017.

[24] T. Truong and S. Yanushkevich, "Detecting subject-weapon visual relationships," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2020, pp. 2047–2052.

[25] N. Dwivedi, D. K. Singh, and D. S. Kushwaha, "Weapon classification using deep convolutional neural network," in *2019 IEEE Conference on Information and Communication Technology*, 2019, pp. 1–5.

[26] V. Kaya, S. Tuncer, and A. Baran, "Detection and classification of different weapon types using deep learning," *Applied Sciences*, vol. 11, no. 16, 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/16/7535

[27] A. Castillo, S. Tabik, F. Pérez, R. Olmos, and F. Herrera, "Brightness guided preprocessing for automatic cold steel weapon detection in surveillance videos with deep learning," *Neurocomputing*, vol. 330, pp. 151–161, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231218313365

[28] S. Karsoliya, "Approximating number of hidden layer neurons in multiple hidden layer bpnn architecture," *International Journal of Engineering Trends and Technology*, vol. 3, 2012.

[29] M. Madhiarasan and S. N. Deepa, "Comparative analysis on hidden neurons estimation in multi layer perceptron neural networks for wind speed forecasting," *Artificial Intelligence Review*, vol. 48, no. 4, pp. 449–471, Dec 2017.

[30] M. Basavarajaiah, "Maxpooling vs minpooling vs average pooling," 2021. [Online]. Available: https://medium.com/@bdhuma/95fb03f45a9

[31] S. Ruder, "An overview of gradient descent optimization algorithms," 2017.

[32] Y. Bengio, *Practical Recommendations for Gradient-Based Training of Deep Architectures*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 437–478.

Amjed Al-Mousa a senior IEEE member and an associate professor of computer engineering at PSUT since 2012. Dr. Al-Mousa has received his Ph.D. from Santa Clara University, M.Sc. from Virginia Tech, and B.Sc. from the University of Jordan all in Electrical Engineering. Dr. Al-Mousa combines this academic experience with more than 13 years of industry experience at Silicon Valley. He started his career by working at Intel Corporation in 2001 and later moved to PDF solutions to work on Design for Manufacturing software tools. After that, he assumed a senior manager position for data analytics at SEVEN networks. Dr. Al-Mousa has served twice as the head of the computer engineering department at PSUT. He developed course content in the fields of Artificial Intelligence, Machine Learning, Cloud Computing, and Big Data. His research interests are in the fields of intelligent systems design and machine learning with varying industrial applications. He can be contacted at email: a.almousa@psut.edu.jo.

Omar Z. Alzaibaq received his B.Sc. degree in Computer Engineering from Princess Sumaya University for Technology (PSUT), Jordan, in 2019. He is currently working as a systems and software engineer at Iotistic Solutions. His main fields include development, Linux-based scripting, fleet management systems, and artificial intelligence.

Yazan Abu Hashyeh received his B.Sc. degree in electronics Engineering from Princess Sumaya University for Technology (PSUT), Jordan, in 2019. Since October 2019, he has worked as an Artificial intelligence engineer at IOTISTIC Solutions. During his position as an Artificial Intelligence engineer. He works part-time Lecturer with Pioneers Academy teaching Python and Artificial intelligence. His main field of work is in image processing and software development.