IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 5, May 2022

DOI: 10.17148/IJARCCE.2022.115191

AIOE BASED REAL TIME THREAT DETECTORS FOR SMART SURVEILLANCE

Er.V.Kokila[1], T.Nalin[2], M.Neelamegam[3]

[1]Asst. Prof Department of ECE & KRISHNASAMY COLLEGE OF ENGINEERING & TECHNOLOGY

[2,3]Department of ECE & KRISHNASAMY COLLEGE OF ENGINEERING & TECHNOLOGY

Abstract: A distributed AI-Powered system that can help security personnel detect various types of weapons in real time. The deep learning architecture interfaces uses single shot detection to allow users to interact with the threat detector system conveniently at the camera and cloud sides. A motion detection module is proposed for detecting moving objects in surveillance videos in real - time. The developed module is integrated seamlessly with both the camera and cloud sides. Security is always a main concern in every domain, due to a rise in crime rate in a crowded event or suspicious lonely areas. Abnormal detection and monitoring have major applications of computer vision to tackle various problems. Due to growing demand in the protection of safety, security and personal properties, needs and deployment of video surveillance systems can recognize and interpret the scene and anomaly events play a vital role in intelligence monitoring. This project implements automatic weapon detection using a convolution neural network (CNN) using Alexnet. Proposed implementation uses datasets, which had pre-labelled images. Results are tabulated, achieve good accuracy, but their application in real situations can be based on the trade-off between speed and accuracy.

Keywords: AI power system, image processing, MatLab, M-files . CNN

I. INTRODUCTION

According to the nonprofit Gun Violence Archive [1], around 100 people are killed using guns, and 200 more are shot and wounded every day in the United States. In 2019, there have been more than 350 mass shootings in the United States. The effects of such gun violence extend far beyond these casualties. It reshapes the lives of millions of people who are witnessing it and living in fear of the next shooting. So, there is a growing social demand for developing effective technological methods to compact gun violence.

The advancement of Artificial Intelligence (AI), Machine Learning (ML), and Internet of Things (IoT) opens up an opportunity to implement such intelligent surveillance systems. Imagine an AI-powered security camera system that can alert about various weapons and guns, masked faces, and suspicious objects in real-time. This can prevent a mass shooting or terrorist attack before it happens.

Deep learning method is gained a good achievement. Most researchers attempted to extract local features from leaves, flowers, and bark for plant classification by using characteristic variation of weapon. There are several datasets are released, However, these datasets consist of picture which were scanned on a simple background, so it gives very high results by using hand-crafted algorithms to extract features. Moreover, a single image of plant in a complex background composed with many leaves, flowers or trees is needed to further investigate. However, hand-crafted features are limited in this case and many preprocessing steps need to consider.

These images are naturally acquired a real environment. In recent years, the CNN model has had tremendous success, it has been playing a principal role to understand the various features of images. We apply CNN models, Alexnet.

The objectives of this thesis are as follows:

- To investigate the feasibility of using computational intelligence image processing in relation to find the weapon.
- To generate image segmentation and features extraction algorithms to distinguish weapon from other surrounding parts using the three neural network systems
- To develop neural network systems based weapon classification algorithms by means of a committee machine to combine several networks based on statistical moment features of weapon images.
- To produce generic algorithms based global optimisation, so as to enhance the colour normalisation.
- To compare the results of the proposed methods with that of the common and renowned method.

In the age of modern science and technology, people use surveillance cameras in dif- ferent areas to prevent crime [1]. Numerous camera systems are installed in different ar- eas, and security guards need to monitor all of these cameras at

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 5, May 2022

DOI: 10.17148/IJARCCE.2022.115191

the same time [2,3]. Gen- erally, after a crime occurs, security guards arrive at the scene, then after checking the recorded images they analyze the images and collect the necessary evidence.

II. PROJECT METHODOLOGY

The neuron image in the database is processed before proceeding to the training process. Figure 3.3 shows the flowchart on how the image is processed.

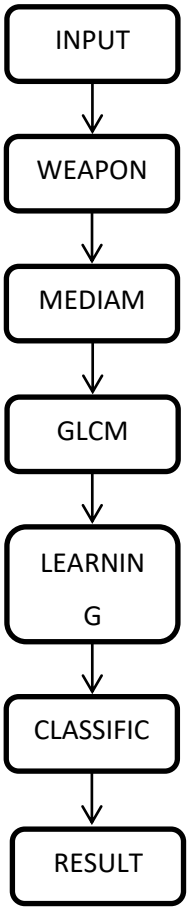

Fig 1: ALGORITHM FLOW CHART

In Figure , the leaf image is pre-processed before extracting the features. The method of extracting features used are mediam Filter and GLCM. The extracted features from are used to obtain the final output class through CNN learning. The overall program flow for this project is shown in Figure 1.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 5, May 2022

DOI: 10.17148/IJARCCE.2022.115191

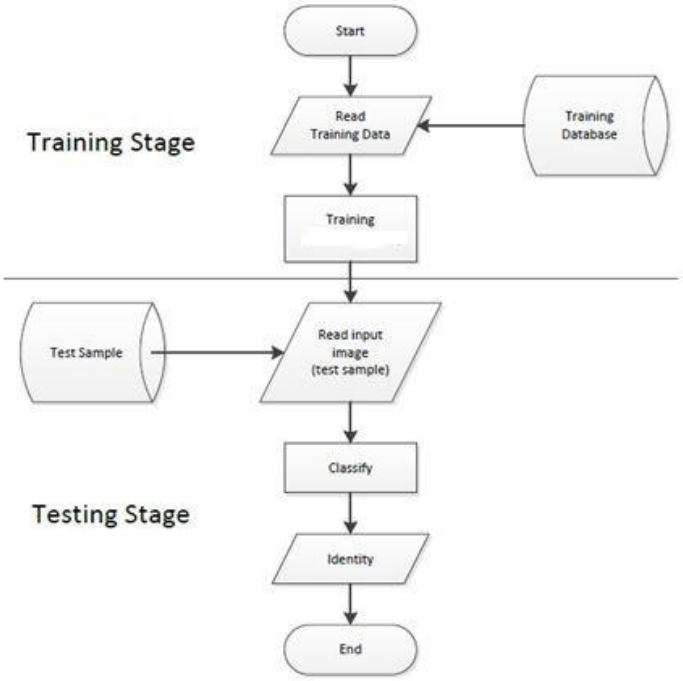

Fig 2: OVERALL PROGRAM FRAMEWORK

The flowchart is divided into two stages, which are training and testing stage. The training stage is a process of training the framework to classify the image sample based on the training data. The testing stage is used to classify the test sample by applying the trained framework.

In the training stage, the important features are extracted from the training sample images (leaf region). The extracted features are used to design a svm. At the end of training stage, the performance of the trained svm.

In the testing stage, the test sample (neuron region) is read by using the same feature extraction method as the trained forest. The extracted features are used to obtain the final output class through the trained random forest. The class of the test image is the maximum class output vote of the random forest.

A. IMAGE PROCESSING

The neuron image in the database is processed before proceeding to the training process. Figure 3.3 shows the flowchart on how the image is processed.

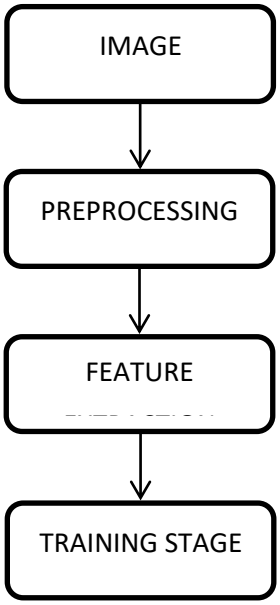

Fig 3: FLOW CHART OF LEAF IMAGE PROCESSING

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 5, May 2022

DOI: 10.17148/IJARCCE.2022.115191

The Leaf image is through the pre-processing where RGB colour image is converted to grayscale image. In the training stage, extracted features will be used to train the framework by trainng. The details of each stage will be discussed in the following section.

PRE-PROCESSING

Pre-processing stage is a process of optimizing the image quality. Filter operations applied to the image to reduce image details for faster computational speed.

.

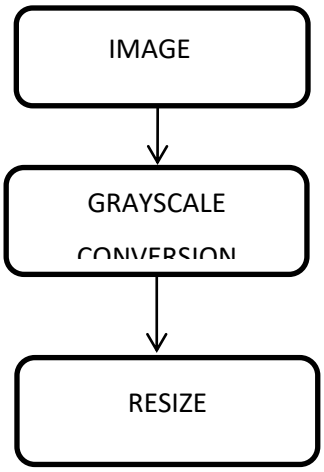

FIG 4: FLOW CHART OF PRE-PROCESSING STAGE

In the pre-processing stage, RGB colour image is converted to grayscale image. An RGB image has three channels which are red, green and blue, where each channel has 8 bits, making a total of 24 bits; whereas a grayscale image contains only 1 channel, which displays the intensity level in 8-bits. Grayscale conversion helps to reduce the dimensional size of image for faster computation.

After the conversion process, the original dimension of the image, 180 × 200 is resized to 64 × 64 pixels. This will reduce the computational speed.

III. CONVOLUTIONAL NEURAL NETWORK

Convolutional Neural Networks is all about using Deep Learning with Computer Vision. A good way to gain intuition about this is to think about a Neural Network Architecture and how it is applied to visual tasks i.e. Images and Video.
Like a Neural Network, a typical Convolutional Neural Network consists of a multiple hidden layers called a Convolutional Layer where the linear function computes the strided convolutions over an image to extract features. It also consists of a pooling layer that computes another function such as Max Pool or Average Pool to reduce the size of the image in the neuron to speed up the computation. It does it by extracting the features of the neuron image and ignoring the rest, this makes the network more robust. There is also fully connected layer which is like a hidden layer in a neural network where the sum of the outputs of each layer are flattened and where each value is an input to the next layer followed by an activation function and an output.

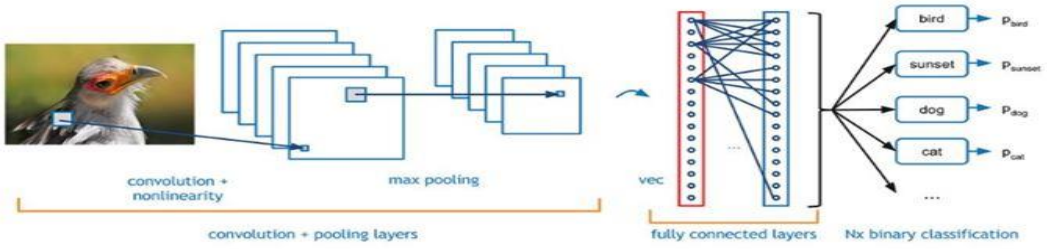

Fig5: CONVOLUTIONAL NEURAL NETWORK ARCHITECTURE EXAMPLE

A. CONVOLUTIONAL LAYER

Convolutional layers apply a convolution operation to the input, passing the result to the next layer. The convolution emulates the response of an individual neuron to visual stimuli.

Each convolutional neuron processes data only for its receptive fully connected. A very high number of neurons

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 5, May 2022

DOI: 10.17148/IJARCCE.2022.115191

would be necessary, even in a shallow (opposite of deep) architecture, due to the very large input sizes associated with images, where each pixel is a relevant variable. For instance, a fully connected layer for a (small) image of size 100 x 100 has 10000 weights for each neuron in the second layer. The convolution operation brings a solution to this problem as it reduces the number of free parameters, allowing the network to be deeper with fewer parameters, For instance, regardless of image size, tiling regions of size 5 x 5, each with the same shared weights, requires only 25 learnable parameters. In this way, it resolves the vanishing or exploding gradients problem in training traditional multi-layer neural networks with many layers by using backpropagation.

B. POOLING:

Convolutional networks may include local or global pooling lay which combine the outputs of neuron clusters at one layer into a single neuron in the next layer. For example, max pooling uses the maximum value from each of a cluster of neurons at the prior layer. Another example is average pooling, which uses the average value from each of a cluster of neurons at the prior layer.

C. FULLY CONNECTED:

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional multi-layer perceptron neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images.

D. RECEPTIVE FIELD:

In neural networks, each neuron receives input from some number of locations in the previous layer. In a fully connected layer, each neuron receives input from every element of the previous layer. In a convolutional layer, neurons receive input from only a restricted subarea of the previous layer. Typically the subarea is of a square shape (e.g., size 5 by 5). The input area of a neuron is called its receptive field. So, in a fully connected layer, the receptive field is the entire previous layer. In a convolutional layer, the receptive area is smaller than the entire previous layer.

E. WEIGHTS:

Each neuron in a neural network computes an output value by applying some function to the input values coming from the receptive field in the previous layer. The function that is applied to the input values is specified by a vector of weights and a bias (typically real numbers). Learning in a neural network progresses by making incremental adjustments to the biases and weights. The vector of weights and the bias are called a filter and represents some feature of the input
(e.g., a particular shape). A distinguishing feature of CNNs is that many neurons share the same filter. This reduces memory footprint because a single bias and a single vector of weights is used across all receptive fields sharing that filter, rather than each receptive field having its own bias and vector of weights.

IV. MATLAB

If you are new to MATLAB, you should start by reading Manipulating Matrices. The most important things to learn are how to enter matrices, how to use the: (colon) operator, and how to invoke functions. After you master the basics, you should read the rest of the sections below and run the demos. At the heart of MATLAB is a new language you must learn before you can fully exploit its power. You can learn the basics of MATLAB quickly, and mastery comes shortly after. You will be rewarded with high productivity, high-creativity computing power that will change the way you work. Development Environment - introduces the MATLAB development environment, including information about tools and the MATLAB desktop. Manipulating Matrices - introduces how to use MATLAB to generate Matrices and perform mathematical operations on matrices. Graphics - introduces MATLAB graphic capabilities, including information about plotting data, annotating graphs, and working with images. Programming with MATLAB - describes how to use the MATLAB language to create scripts and functions, and manipulate data structures, such as cell arrays.

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 5, May 2022

DOI: 10.17148/IJARCCE.2022.115191

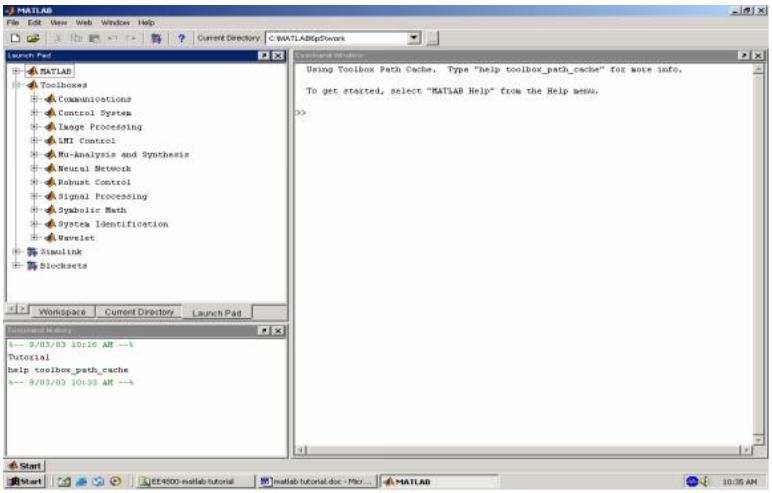

Fig6: MATLAB window

The Command Window is the window on the right hand side of the screen. This window is used to both enter commands for MATLAB to execute, and to view the results of these commands. The Command History window, in the lower left side of the screen, displays the commands that have been recently entered into the Command Window. In the upper left hand side of the screen there is a window that can contain three different windows with tabs to select between them. The first window is the Current Directory, which tells the user which M-files are currently in use. The second window is the Workspace window, which displays which variables are currently being used and how big they are. The third window is the Launch Pad window, which is especially important since it contains easy access to the available toolboxes, of which, Image Processing is one. If these three windows do not all appear as tabs below the window space, simply go to View and select the ones you want to appear. In order to gain some familiarity with the Command Window.

M-FILE:

M-file – An M-file is a MATLAB document the user creates to store the code they write for their specific application. Creating an M-file is highly recommended, although not entirely necessary. An M-file is useful because it saves the code the user has written for their application. It can be manipulated and tested until it meets the user's specifications. The advantage of using an Mfile is that the user, after modifying their code, must only tell MATLAB to run the M-file, rather than reenter each line of code individually. 3.2. Creating an M-file – To create an M-file, select File\New ►M-file.

Saving – The next step is to save the newly created M-file. In the M-file window, select File\Save As… Choose a location that suits your needs, such as a disk, the hard drive or the U drive. It is not recommended that you work from your disk or from the U drive, so before editing and testing your M-file you may want to move your file to the hard drive. Opening an M-file – To open up a previously designed M-file, simply open MATLAB in the same manner as described before. Then, open the M-file by going to File\Open…, and selecting your file. Then, in order for MATLAB to recognize where your M-file is stored, you must go to File\Set Path… This will open up a window that will enable you to tell MATLAB where your M-file is stored. Click the Add Folder… button, then browse to find the folder that your M-file is located in, and press OK. Then in the Set Path window, select Save, and then Close. If you do not set the path, MATLAB may open a window saying your file is not in the current directory. In order to get by this, select the "Add directory to the top of the MATLAB path" button, and hit OK. This is essentially the same as setting the path, as described above. Writing Code – After creating and saving your M-file, the next step is to begin writing code. A suggested first move is to begin by writing comments at the top of the M-file with a description of what the code is for, who designed it, when it was created, and when it was last modified. Comments are declared by placing a % symbol before them. Comments appear in green in the M-file window

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified ⁜ Impact Factor 7.39 ⁜ Vol. 11, Issue 5, May 2022

DOI: 10.17148/IJARCCE.2022.115191

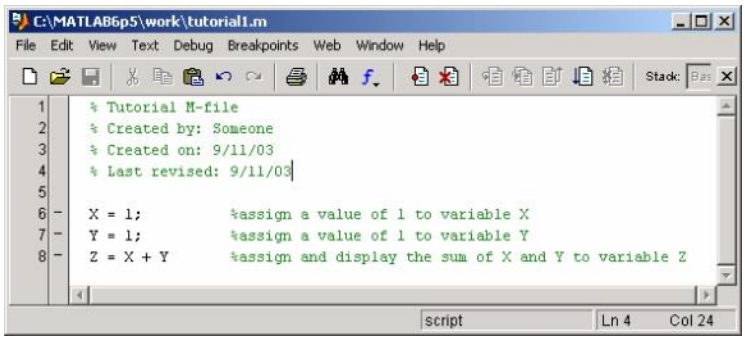

Fig 7: EXAMPLE of M-file

Images – The first step in MATLAB image processing is to understand that a digital image is composed of a two or three dimensional matrix of pixels. Individual pixels contain a number or numbers representing what grayscale or color value is assigned to it.

V. CONCLUSION

In today's conditions, where criminal activities are increasing, it is very important to detect and recognize whether there is a weapon on a person based on images taken from security cameras, without requiring human intervention. Weapon detection and recogni- tion are important to prevent criminal activities before they occur and so that the appro- priate parties can take necessary action. Most criminal activities are carried out using handheld or carried weapons. Handheld or carried weapons are the most important ele- ments used for various crimes, such as theft, illegal hunting, and terrorism. It is necessary to determine the types of weapons that may constitute a criminal element in order to pre- dict whether there will be any criminal element in the images taken from security cameras and to take the necessary precautions beforehand.

In conclusion, the model proposed in this study will contribute to the elimination of many security gaps by increasing the task effectiveness and efficiency of security forces in security applications. In this respect, the fact that the results can be directly transferred to new applications increases the original value of the study. It is expected that the results of the study will lead and contribute to similar studies being undertaken, especially regard- ing autonomous security units. In future studies, an infrastructure investigation could be done on robot soldiers that can automatically monitor and analyze input data and send alerts to security forces in order to process data in real time using security control systems and to increase classifi- cation accuracy. In addition, studies should be developed to detect coated guns.

REFERENCES

1. Raturi, G.; Rani, P.; Madan, S.; Dosanjh, S. ADoCW: An Automated method for Detection of Concealed Weapon. In Proceedings of the Fifth International Conference on Image Information Processing, Shimla, India, 15–17 November 2019; pp. 181–186.
2. Bhagyalakshmi, P.; Indhumathi, P.; Lakshmi, R.; Bhavadharini, D. Real Time Video Surveillance for Automated Weapon Detection. Int. J. Trend Sci. Res. Dev. 2019, 3, 465–470, doi:10.31142/ijtsrd22791.
3. Lim, J.; Jobayer, M.I.A.; Baskaran, V.M.; Lim, J.M.; Wong, K.; See, J. Gun Detection in Surveillance Videos using Deep Neural Networks. In Proceedings of the Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, Lanzhou, China, 18–21 November 2019; pp. 1998–2002, doi:10.1109/APSIPAASC47483.2019.9023182.
4. Yuan, J.; Guo, C. A deep learning method for detection of dangerous equipment. In Proceedings of the Eighth International Conference on Information Science and Technology, Granada, Cordoba, and Seville, Spain, 30 June–6 July 2018; pp. 159–164, doi:10.1109/ICIST.2018.8426165.
5. Romero, D.; Salamea, C. Convolutional models for the detection of firearms in surveillance videos. Appl. Sci. 2019, 9, 1–11, doi:10.3390/app9152965.
6. Ilgin, F.Y. Energy-based spectrum sensing with copulas for cognitive radios. Bull. Polish Acad. Sci. Tech. Sci. 2020, 68, 829–834, doi:10.24425/bpasts.2020.134177.
7. Navalgund, U.V.; Priyadharshini, K. Crime Intention Detection System Using Deep Learning. In Proceedings of the International Conference on Circuits and Systems in Digital Enterprise Technology, Kottayam, India, 21–22 December 2018; pp. 1–6, doi:10.1109/ICCSDET.2018.8821168.
8. Chandan, G.; Jain, A.; Jain, H. Real time object detection and tracking using Deep Learning and OpenCV. In Proceedings of the International Conference on Inventive Research in Computing Applications, Coimbatore, India,

DOI: 10.17148/IJARCCE.2022.115191

11–12 July 2018; pp. 1305–1308, doi:10.1109/ICIRCA.2018.8597266.
9. Deng, L.; Yu, D. Deep learning: Methods and applications. Found. Trends Signal Process. 2014, 7, 197–387, doi:10.1561/2000000039. 10. Bengio, Y. Learning deep architectures for AI. Found. Trends Mach. Learn. 2009, 2, 1–27, doi:10.1561/2200000006.
11. Lecun, Y.; Bengio, Y.; Hinton, G. Deep learning. Nature 2015, 521, 436–444, doi:10.1038/nature14539.
12. Song, H.A.; Lee, S.Y. Hierarchical representation using NMF. Lect. Notes Comput. Sci. 2013, 8226, 466–473, doi:10.1007/978–3- 642-42054-2_58.
13. Masood, S.; Ahsan, U.; Munawwar, F.; Rizvi, D.R.; Ahmed, M. Scene Recognition from Image Using Convolutional Neural Network. Procedia Comput. Sci. 2019, 167, 1005–1012, doi:10.1016/j.procs.2020.03.400.
14. Wu, X.; Sahoo, D.; Hoi, S.C.H. Recent advances in deep learning for object detection. Neurocomputing 2020, 396, 39–64, doi:10.1016/j.neucom.2020.01.085.
15. Krishna Sai, B.N.; Sasikala, T. Object Detection and Count of Objects in Image using Tensor Flow Object Detection API. In Proceedings of the International Conference on Smart Systems and Inventive Technology, Tirunelveli, India, 27–29 November 2019; pp. 542–546, doi:10.1109/ICSSIT46314.2019.8987942.
16. Verma, G.K.; Dhillon, A. A Handheld Gun Detection using Faster R-CNN Deep Learning. In Proceedings of the 7th International Conference on Computer and Communication Technology, Allahabad, India, 24–26 November 2017; pp. 84–88, doi:10.1145/3154979.3154988.
17. Warsi, A.; Abdullah, M.; Husen, M.N.; Yahya, M. Automatic Handgun and Knife Detection Algorithms: A Review. In Proceedings of the 14th International Conference on Ubiquitous Information Management and Communication, Taichung, Taiwan, 3–5 January 2020; pp. 1–9, doi:10.1109/IMCOM48794.2020.9001725.
18. Olmos, R.; Tabik, S.; Herrera, F. Automatic handgun detection alarm in videos using deep learning. Neurocomputing 2018, 275, 66–72, doi:10.1016/j.neucom.2017.05.012.
19. Asnani, S.; Ahmed, A.; Manjotho, A.A. Bank Security System based on Weapon Detection using HOG Features. Asia J. Eng. Sci. Technol. 2014, 4, 23–29.
20. Developing a Real-Time Gun Detection Classifier. Available online: https://www.scribd.com/document/380866575/Developing- a-Real-Time-Gun-Detection-Classifier (accessed on 8 May 2021).
21. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Proceedings of the International Conference on Learning Representations, San diego, CA, USA, 7–9 May 2015; pp. 1–15.
22. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; Volume 2016, pp. 770–778, doi:10.1109/CVPR.2016.90.
23. Olmos, R.; Tabik, S.; Lamas, A.; Pérez-Hernández, F.; Herrera, F. A binocular image fusion approach for minimizing false positives in handgun detection with deep learning Inf. Fusion 2019, 49, 271–280, doi:10.1016/j.inffus.2018.11.015.
24. Castillo, A.; Tabik, S.; Pérez, F.; Olmos, R.; Herrera, F. Brightness guided preprocessing for automatic cold steel weapon detection in surveillance videos with deep learning. Neurocomputing 2019, 330, 151–161, doi:10.1016/j.neucom.2018.10.076.
25. Ineneji, C.; Kusaf, M. Hybrid weapon detection algorithm, using material test and fuzzy logic system. Comput. Electr. Eng. 2019, 78, 437–448, doi:10.1016/j.compeleceng.2019.08.005.
26. Dwivedi, N.; Singh, D.K.; Kushwaha, D.S. Weapon classification using deep convolutional neural network. In Proceedings of the Conference on Information and Communication Technology, Allahabad, India, 6–8 December 2019; pp. 1–5, doi:10.1109/CICT48419.2019.9066227.
27. Egiazarov, A.; Mavroeidis, V.; Zennaro, F.M.; Vishi, K. Firearm detection and segmentation using an ensemble of semantic neural networks. In Proceedings of the European Intelligence and Security Informatics Conference, Oulu, Findland, 26–27 November 2019; pp. 70–77, doi:10.1109/EISIC49498.2019.9108871.
28. Ben Abdallah, H.; Abdellatif, T.; Chekir, F. AMSEP: Automated Multi-level Security Management for Multimedia Event Processing. Procedia Comput. Sci. 2018, 134, pp. 452–457, doi:10.1016/j.procs.2018.07.184.
29. Grega, M.; Matiolański, A.; Guzik, P.; Leszczuk, M. Automated detection of firearms and knives in a CCTV image. Sensors 2016, 16, doi:10.3390/s16010047.
30. Tiwari, R.K.; Verma, G.K. A Computer Vision based Framework for Visual Gun Detection Using Harris Interest Point Detector. Procedia Comput. Sci. 2015, 54, 703–712, doi:10.1016/j.procs.2015.06.083.
31. Xu, Z.; Tian, Y.; Hu, X.; Pu, F. Dangerous human event understanding using human-object interaction model. In Proceedings of the International Conference on Signal Processing, Communications and Computing, Ningbo, China, 19–22 September 2015; pp. 1–5, doi:10.1109/ICSPCC.2015.7338786.
32. Angelova, A.; Krizhevsky, A.; Vanhoucke, V. Pedestrian detection with a Large-Field-Of-View deep network. In Proceedings of the International conference on robotics and automation, Seattle, WA, USA, 26–30 May 2015; pp. 704–711, doi:10.1109/ICRA.2015.7139256.

DOI: 10.17148/IJARCCE.2022.115191

33. Hinton, G.E. A practical guide to training restricted boltzmann machines. In Neural Networks: Tricks of the Trade; Montavon, G., Orr, G.B., Müller, K.R., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 599–619, doi:10.1007/978-3-642-35289-8-32.
34. Agarap, A.F. Deep Learning using Rectified Linear Units (ReLU). arXiv 2018, arXiv:1803.08375, 2018.
35. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. Phys. Lett. B 2014, 15, 1929–1958.
36. AdaMax Online Training for Speech Recognition. Available online: http://cslt.riit.tsinghua.edu.cn/mediawiki/images/d/df/Adamax_Online_Training_for_Speech_Recognition.pdf (accessed on 10 December 2020).
37. Istock. Available online: https://www.istockphoto.com (accessed on 20 August 2020).
38. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J.; Berkeley, U.C.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; Volume 1, p. 5000, doi:10.1109/CVPR.2014.81.