ISSN: 0974-5823 Vol. 7 No. 5 May, 2022

International Journal of Mechanical Engineering

Comparative Analysis on Light Weighted Deep Learning Models for Implementation in Real-time Video Surveillance Systems

Immanuel K *[1] , Dinesh Kumar A[2] , Gnana Kousalya [3]

*1 Assistant Professor, Department of ECE, St. Joseph's Institute of Technology, OMR, Chennai – 119

2 Assistant Professor, Department of ECE, St. Joseph's Institute of Technology, OMR, Chennai – 119

3 Professor, Department of ECE, St. Joseph's Institute of Technology, OMR, Chennai – 119

Abstract

Real-time Object Detection in video streams often requires heavy hardware and intensive algorithms to produce accurate results. It is difficult to develop and implement a deep learning model for real-time object detection in video surveillance on lighter hardware. The manual surveillance techniques seem to be time-consuming and require a lot of data. Real-time video surveillance models will reduce processing time and data requirements. Convolutional Neural Networks (CNN) perform well in Deep Learning Models when it comes to Object Recognition. In this paper, we discuss how these models are used in real-time video surveillance and provide a comparative analysis of them. The base network and fully connected layer will enhance results for a light-weighted execution. The Single Shot Detection (SSD) will be implemented to ensure more frames per second.

Keywords: Video Surveillance, Deep Learning, Real-time Object Recognition.

Introduction

In recent years, Computer Vision has gained significance as a tool for simple solutions to complex problems. As a result, the human brain processes information in a flash and provides conclusions and results rapidly. A similar approach is being transferred to computers and making them understand the scene and give conclusions and results in real-time. To make the system understand the input, deep learning surpasses the traditional use of shallow networks. As a result, the two-dimensional array of data values collectively known as pixels is now used to learn the attributes of the objects in the frame and perform intelligent analysis of them. CNN (Convolutional Neural Network) was introduced to lighten this task of making the system understand the scene by not feeding the entire array of pixels directly to the decision-making system, but by designing a model with many connected layers to extract only the key features necessary for analysis. The CNN is a deep learning neural network with Convolutional Layers, Pooling Layers, and Fully Connected Layers for extracting attributes from inputs. There are two phases to the process. The first phase is the training phase, and the second phase is the testing phase. During the training phase, a set of known inputs is given to the system for the purpose of making it learn the attributes of each member of the set in its own way. During the testing phase, the system is exposed to real-time inputs to produce analytical results based on the knowledge gained during the training phase. While this approach might be suitable for a single image in which the system has enough time to run the entire model and provide the results after some computation, when it is applied to a real-time situation in which continuous frames arrive as input for making decisions within a very short period, effective implementation of the model is required.

In this paper, we will discuss the concepts of detecting objects in real-time which can be used in the development of a video surveillance system that can autonomously make decisions based on the attributes extracted.

Materials and methods

The choice of the deep learning model which should be used in real-time object detection must be lightweight. Only a lightweight model will be suitable for running on a platform with low computational power. The accuracy of the detection results will have a trade-off with the complexity cost and model size of the network. This research will maintain the balance between these two factors. The hardware used to implement the models and get the results is the NVIDIA Jetson Nano Development Kit(B01). The video source is taken from the JVC ProHD 3D camcorder through an HDMI port, the same is given as input to the hardware using a USB converter module. The specifications of the video stream are given as 1920x1080 (Full HD, YUV 4:2:0) source with a frame rate of 30 fps(progressive). The experimental setup is shown in (Figure 1).

Figure 1: The Experimental Setup

Results and discussion

The real-time video surveillance model is given as (Figure 2).

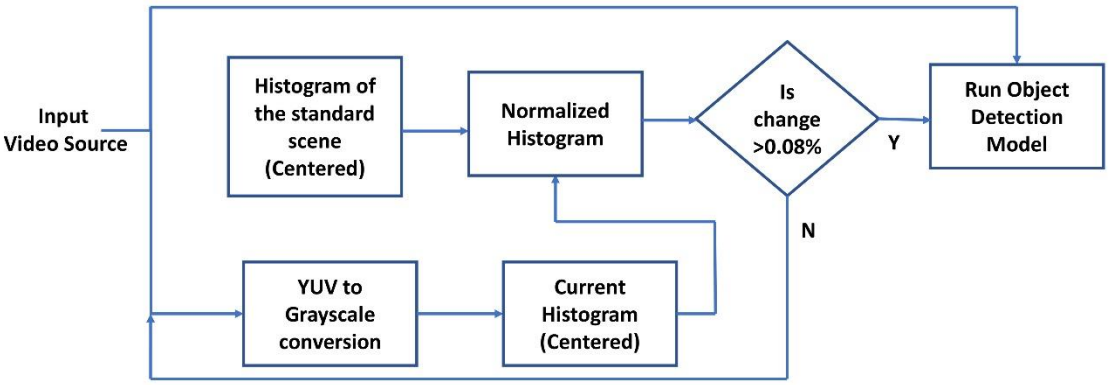


Figure *2*: The Real-time Video Surveillance Model

The Deep Learning models always load the target hardware on which they are run, to lighten the load the following methodology is used. The scene to be analyzed under standard conditions (without any additional objects added to the scene) is first considered. The Histogram of the Gray Scale converted scene is noted as the standard value. The one frame is sampled per second and a histogram is computed. The histogram is centered to implemented illumination variation compensation. The normalized histogram is computed with the inputs as the histogram under standard conditions and the current sampled histogram. If the error is greater than 0.08% which means if there be a change in an area of 40x40, the object detection model will be started to identify the objects in the scene, else the sampling process and normalized histogram computation is performed.

The pair of histograms for S, C, where S be the standard scene and C be the current scene is given as [1]

$$\sum_{j=1}^{n} \min(S_j, C_j) \qquad (1)$$

where n is the quantization levels used. The normalized fractional match value between 0 and 1 the intersection is given by

$$H(S, C) = \frac{\sum_{j=1}^{n} \min(S_j, C_j)}{\sum_{j=1}^{n} S_j} \qquad (2)$$

The object detection runs if the match value (Normalized Histogram Value) goes below 0.08%. The log is maintained for the objects detected with the timestamp. After 10 secs again the normalized histogram is computed if the match value falls within the

defined range the object detection model is stopped and the system returns to the initial state, or else the object detection model runs till the match value falls within the prescribed range.

Comparative Analysis of the Deep Learning Model used for Object Detection

The CNN Deep Learning model used for the Object Detection has two parts. The first part is the Base Network used for Image Classification followed by the second part which is the fully connected layer.

The Base Network we will consider for our analysis will be MobileNet V1[2], MobileNet V2 [3] and Inception V3 [4]. These models are primarily considered because of their light-weighted structure. In the MobileNet V1, the Depth Wise Separable Convolution is employed to reduce the size and complexity of the model. The smaller size of the model will reduce the number of parameters to be computed and the less complexity implies fewer convolutions. The model also incorporates two parameters the Width Multiplier α and Resolution Multiplier ρ. The width multiplier parameter adds a depthwise separable convolution cost to the model and the typical values range from 0 to 1. The resolution multiplier parameter controls the input resolution of the image and the typical values range from 0 to 1. In the implemented model $\alpha = 0.5$ and $\rho = 1$. The MobileNet V1 has some Non-Linearities in the narrow layers which were removed by changing the structure in the MobileNet V2. The MobileNet V2 has two blocks, one is the residual block with stride 1 and the second block is stride 2 used for downsizing. The MobileNet V1 contains only 2 Layers (Depth wise Convolution and PointWise Convolution with activation function), whereas the MobileNet V2 has 3 Layers(Point Wise Convolution, Depth wise Convolution and Point Wise Convolution). The Width Multiplier value also ranges from 0.35 to 1.4. Both the versions of MobileNet are lighter compared to other object classification models since they use the Depth Wise Convolution followed by the PointWise Convolution, this also makes the model run on a lesser number of computations with a lesser number of parameters. Because of this factor, there is a slight decrease in the performance. To overcome this problem, the Inception V3 model is considered. This model uses the Standard Convolution to improve the performance of image classification. The initial version of the Inception model used a 5x5 convolutional block, using smart factorization methods, convolutions are transformed to reduce the computational complexity of the system. This operation also reduces the computational time for the strides to get completed. The convolution operation is organized through filter banks to make the model wider instead of deeper. The factorized convolutions reorganize the dimensions of the convolutional operations to reduce the computational complexity and time.

The fully connected layer for the above-discussed base models is replaced with the Single Shot Multibox Detector[5]. The SSD is based on a feed-forward convolutional network that will produce fixed-sized bounding boxes and the presence of the object classes is scored which is followed by a non-maximum suppression step to produce the final decisions. A small kernel is first selected to either produce a score for a category or a shape offset relative to the chosen box coordinates. The offset is predicted relative to the considered default box. Class scores are computed at locations 4 offsets relative to the original default box shape. This box is grown allowing different default box shapes in several feature maps effectively discretizing the possible space for the output box. This makes the bounding box for object detection easier and more cost-effective.

Real-time Execution and Results from the models

All three models were imported into the device, converted into the Open Neural Network Exchange(ONNX) format and run. The NVIDIA CUDA RT [6] cores were enabled, the NVIDIA TENSOR RT [7] with Pytorch was used to run the model on the hardware in real-time. The Pre-trained detection models were imported and additional training was carried out for detecting customized objects. The model was run on a video stream and a single image. The results are given below.

Timing Report Generated for Single Image Object Detection :

Input Image – 844 x 563, 3 Channel JPEG image

Model	Pre-Process Time (ms)		Network Loading Time (ms)		Post-Process Time (ms)		Visualisation Time (ms)		Total Time (ms)	
	CPU	CUDA	CPU	CUDA	CPU	CUDA	CPU	CUDA	CPU	CUDA
SSD MobileNet V1	0.0787	1.8324	1661.3394	1659.1391	0.0569	0.0524	33.2730	33.7625	1694.7481	1694.7866
SSD MobileNet V2	0.0783	1.4788	1565.1918	1563.4633	0.0760	0.0749	37.9278	38.4570	1603.2741	1603.4742
SSD Inception V2	0.0940	1.8217	121.4458	119.3156	0.0557	0.0546	35.7145	36.0733	157.3102	157.2654

Table 1: Timing report generated on running the different Deep Learning Models using NVIDIA Jetson Nano

The Timing report generated on running the different models is available in (Table 1). On comparing the results both MobileNet versions take longer times than the Inception model to get loaded. The execution time of the MobileNet versions including the pre-processing and the post-processing time is comparatively lower than the Inception Model. An overlay is initiated to show the bounding boxes of the detected objects, with the display of the classified label.

Figure 3: Result of MobileNet V1

Figure 4: Result of MobileNet V2

Figure 5: Result of Inception V2

Comparing the on-screen results given by the models (Figures 3,4,5). The accuracy score of the MobileNet versions 1 and 2 are comparatively lesser than the Inception V2. But the MobileNet versions can detect the person who is far-off in view in the image whose portion looks blurred because of the Depth of Field (DOF) of the lens used. The Inception V2 fails to identify that.

Results of Real-time Object Recognition from Video Stream – 1920x1080 (YUV – 4:2:0) 30 fps

Model	FPS	Accuracy Score	Dropped Out Frames
SSD MobileNet V1	28-30	90.4 to 92.5	1-2
SSD MobileNet V2	23-24	94.3 to 96.2	6 - 7
SSD Inception V2	19-20	95.4 to 97.2	11-12

Table 2: Results on running the different Deep Learning Models using NVIDIA Jetson Nano on real-time video stream

Results tabulated in (Table 2) gives us a detailed idea about the performance of different deep learning models implemented to run in real-time. The MobileNet versions have an improved frame rate because of their lighter structure, whereas the Inception has a lesser frame rate. Considering the accuracy score given by the models, Inception stands first because of its detailed structure whereas the MobileNet V2 comes closer to the accuracy score of the Inception.

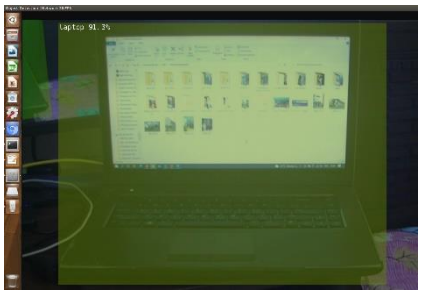


Figure 6: Result of MobileNet V1

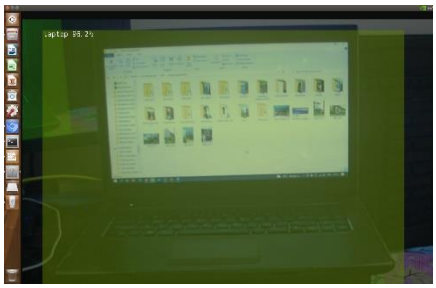


Figure 7: Result of MobileNet V2

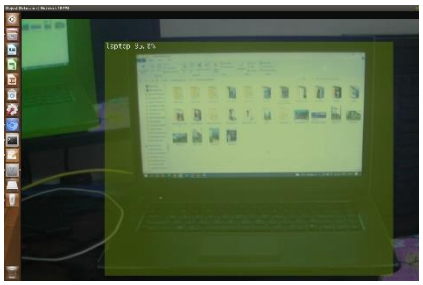
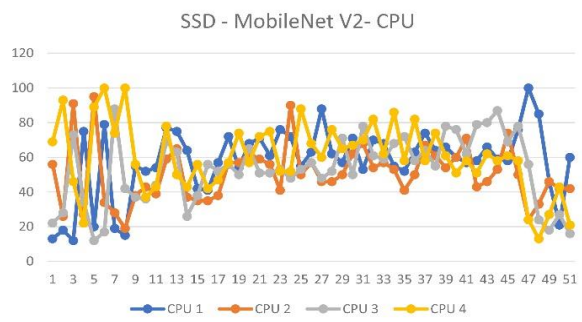


Figure 8: Result of Inception V2



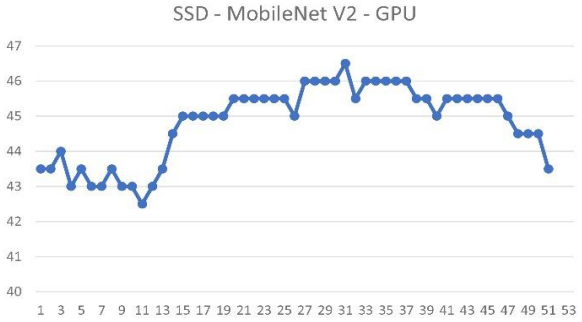
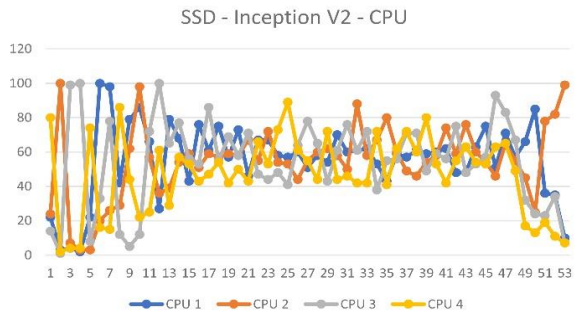


Figure 9: CPU Utilization on running MobileNet V1 **Figure 10:** GPU Utilization on running MobileNet V1



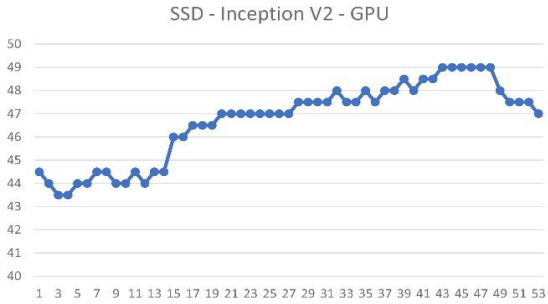
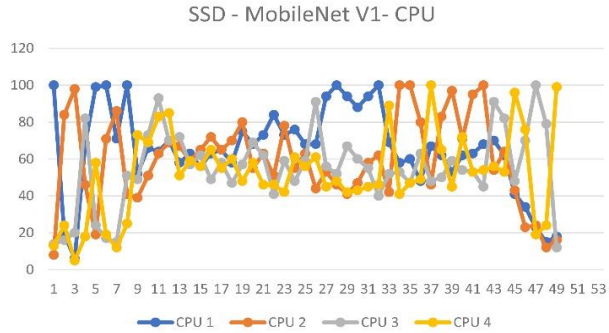


Figure 11: CPU Utilization on running MobileNet V2 **Figure 12:** GPU Utilization on running MobileNet V2



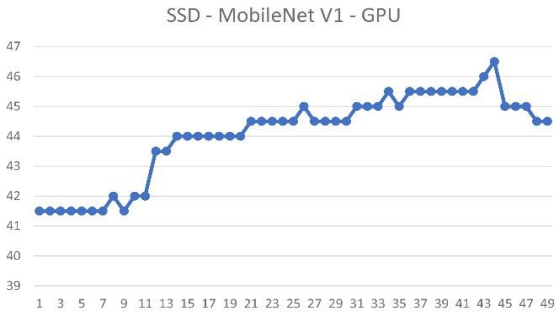


Figure 13: CPU Utilization on running Inception V2 **Figure 14:** GPU Utilization on running Inception V2

The real-time results of running the three deep learning models on the video streams can be compared from the figures shown (Figure 9,10,11,12,13,14). The CPU Utilization and the GPU Utilization is plotted with the sampling rate of 1000ms. Since the hardware has four CPU cores operating at 1.43 GHz, the utilization of the individual cores is plotted. During the initial phase, the deep learning model is loaded, then the CPU Utilization peaks, when the model runs and starts detecting the objects, the average utilization drops down. During this phase the GPU utilization increases by 3 – 4% which can be seen from the results.

Conclusions

The three deep learning models were implemented on the hardware and detailed results were noted and analyzed. The trade-off between the complexity of running the model and the results always exist. To have a balance between both, based on the application the model can be implemented. For accurate results compromising slightly on the real-time characteristics, the Inception model can be used, considering the real-time scenario, and compromising slightly on the accuracy the MobileNet versions can be used. These models suit well for a real-time video surveillance system.

References

[1] Kevin Jeffay, Hongjiang Zhang, In The Morgan Kaufmann Series in Multimedia Information and Systems,Readings in Multimedia Computing and Networking, Morgan Kaufmann, 2002, Pages xiii-xvii, ISBN 9781558606517, https://doi.org/10.1016/B978-155860651-7/50084-X. (https://www.sciencedirect.com/science/article/pii/B978155860651750084X)

[2] Kaur, Chamandeep & Boush, Mawahib & Hassen, Samar & Hakami, Wafaa & Abdalraheem, Mohammed & Galam, Najla & Hadi, Nedaa & Benjeed, Atheer. (2022). Incorporating sentimental analysis into development of a hybrid classification model: A comprehensive study. International Journal of Health Sciences. 6. 1709-1720. 10.53730/ijhs.v6nS1.4924.

[3] Sandler, Mark, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. "Mobilenetv2: Inverted residuals and linear bottlenecks." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4510-4520. 2018.

[4] Burayk, Dr & Mohammed, Dr & Kaur, Chamandeep. (2022). PURCHASE INFLUENCE DETERMINANTS OF PRESCHOOLERS AND PRIMARY SCHOOL-GOING CHILDREN. International Journal of Early Childhood Special Education. 14. 2022.

[5] Liu, Wei, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. "Ssd: Single shot multibox detector." In European conference on computer vision, pp. 21-37. Springer, Cham, 2016.

[6] Anjum, Afsana & Kaur, Chamandeep & Kondapalli, Sunanda & Hussain, Mohammed & Begum, Ahmed & Hassen, Samar & Boush, Dr & Benjeed, Atheer & Abdulraheem, Dr. (2021). A Mysterious and Darkside of The Darknet: A Qualitative Study. Webology. 18. 285-294.

[7] Howard, Andrew G., Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861 (2017)

[8] Szegedy, Christian, Sergey Ioffe, Vincent Vanhoucke, and Alexander A. Alemi. "Inception-v4, inception-resnet and the impact of residual connections on learning." In Thirty-first AAAI conference on artificial intelligence. 2017.

[9] Parallel Programming models developed by NVIDIA Corporation

[10] Chopra, Pooja & Gollamandala, Vijay & Ahmed, Ahmed & Bala Gangadhara Tilak Babu, Sayila & Kaur, Chamandeep & Prasad N, Achyutha & Nuagah, Stephen. (2022). Automated Registration of Multiangle SAR Images Using Artificial Intelligence. Mobile Information Systems. 2022. 1-10. 10.1155/2022/4545139.