EDGE INTELLIGENCE-BASED OBJECT DETECTION AND RECOGNITION SYSTEM FOR EMBEDDED IOMT APPLICATIONS

Vinaya Gohokar* and Vijay Gohokar**

Abstract

Implementation of artificial intelligence-based algorithms on resource-constrained devices at the edge of the network is a challenging task. This paper reviews architecture, models, and requirements to implement edge intelligence-based application. A comparative study of edge devices, Raspberry pi (Rpi) with neural compute stick (NCS), and Jetson Nano (Nano) for object detection and recognition using deep learning models is presented. Incisive observations regarding impact of optimisation frameworks, libraries, and selection of the right device depending on the application are discussed. Inference time, energy use, and cost effectiveness are compared. The results obtained using Tensor-RT on Jetson Nano have proved promising for IoMT applications. Mobile Net v2 model achieves best performance.

Key Words

Edge intelligence, object detection, Internet of Things (IoT), artificial intelligence, Jetson Nano, IoMT

1. Introduction

Edge intelligence, a buzzword today, represents a system of connected devices for sensor data acquisition, caching, processing, and analysis in the vicinity of the source of data based on artificial intelligence. It aims at enhancing data processing and protecting the privacy and security of the data and users. The enormous increase in multimedia-traffic, particularly video, has drastically shifted the vision of the IoT and the term Multimedia Internet of Things (M-IoT) or Internet of Multimedia Things (IoMT) is becoming predominant. There are numerous applications of computer vision in the IoT where intelligence is required at the edge. Artificial intelligence-based algorithms based on machine learning (ML) and deep learning (DL), achieves up-to-date performance in various domains [1]–[4].

Various artificial intelligence-based algorithms are available for computer vision-based applications. Object detection and recognition are important steps in many missions critical IoMT solutions. It is very hard for object detection-based methods to be integrated into limited computing resources enabled embedded systems. Determining the best approach for object recognition depends on application under development. In many cases, machine learning can be a capable technique, particularly if features of the image to differentiate classes of objects are easily recognisable. In recent years a lot of development has been observed in deep learning for object detection and recognition. The main consideration while making selection between machine learning and deep learning is the accessibility of powerful GPU and large numbers of labeled images for training. Deep learning-based methods tend to work well with more images. Due to the computational requirements, the CNNs are typically trained on GPUs. It reduces the time taken to train the model. Deep learning solutions based on data and cloud computing-based systems come across some grave limitations at edge devices in real time practical applications [5], [6].

As it is not possible to bring the edge devices to data centres, intelligence is needed at the device itself. This is particularly important while deploying real-time computer vision-based applications on the edge devices. Various embedded platforms are available for Internet of Things-based applications, but they are constrained in capabilities in terms of power, computational speed, size, and storage memory. Delay sensitive computer vision-based applications are difficult to realise on these devices. Cloud-based architecture is unsuitable in case of these applications. To deploy AI-based models on the embedded platforms, it is necessary to convert them into hardware friendly deployable runtime packages, to realise unified performance at the edge.

Pre-trained neural network models which are trained on data sets like Image Net are used in object detection tasks. These open-source models are attracting people working in IoMT. The pre-trained models are playing a major role for speedy developments in Computer Vision research. Newcomers in this area can use these state-of-the-art models instead of developing everything from

[*] School of ECE, Dr Vishwanath Karad MIT World Peace University, Pune, India; e-mail: vvgohokar@gmail.com
[**] MMCOE, Pune, India; e-mail: vngohokar@rediffmail.com
Corresponding author: Vinaya Gohokar

(DOI: 10.2316/J.2024.206-0881)

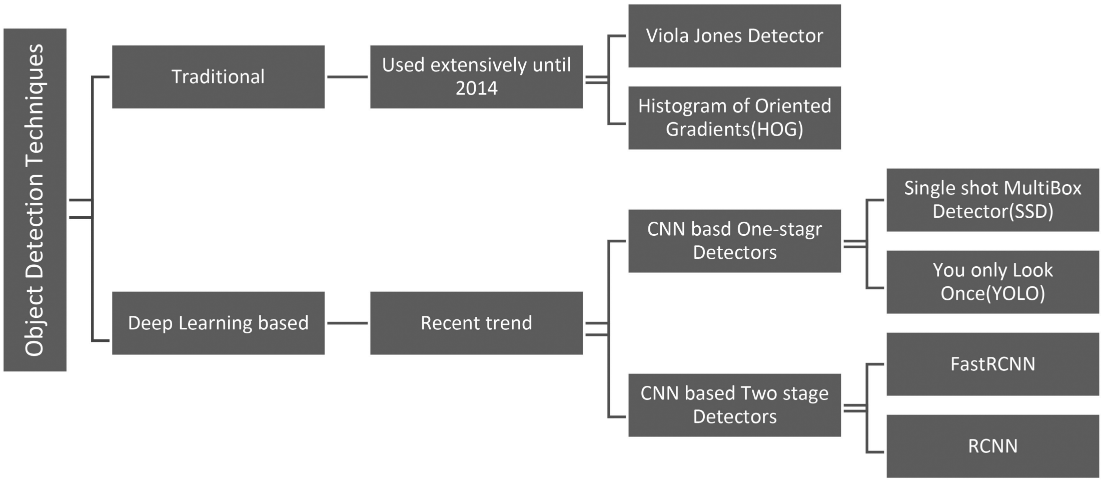

Object Detection Techniques
Traditional
Used extensively until 2014
Viola Jones Detector
Histogram of Oriented Gradients(HOG)
Deep Learning based
Recent trend
CNN basd One-stagr Detectors
Single shot MultiBox Detector(SSD)
You only Look Once(YOLO)
CNN based Two stage Detectors
FastRCNN
RCNN

Figure 1. Object detection methods.

scratch [7]. Numerous additional pertinent applications of DL, particularly those at the edge, rely on time-series processing and call for models with characteristics. Temporal convolutional networks (TCNs), a convolutional model for time-series processing is explored and is found effective [8], PG-YOLO, modified version of YOLOv5 with low computational complexity to improve the network performance is used in [9]. Object detection architecture cantered on edge computing to attain effective object detection for surveillance applications is developed in [10]. An adaptive Region-of-Interest-based image compression algorithm for end devices to efficiently compress captured images for wireless transmission is used by the researchers.

There are various hardware platforms available for edge intelligence-based system design. The study explores and compares performance of Intel's NCS stick with Rpi and NVDIA Jetson Nano for implementation of object detection algorithm. Contribution of this paper includes the following.

Section 2 explores various object detection methods for embedded platforms. Traditional and deep learning-based approaches are compared. Brief review on architecture, models and requirements for solutions that implement edge intelligence-based applications is presented in Section 3. Section 4 explains about capabilities of Jetson Nano as Edge device for IoMT applications. Edge intelligence-based object detection and recognition system on Nano is described in Section 5. Section 6 presents the results and conclusion. The future scope is also mentioned in the same section.

2. Object Detection and Recognition on Embedded Platform

Object detection and recognition are computation intensive tasks from artificial intelligence-based algorithms and hence it is very challenging on resource-constrained embedded systems. When the application needs instantaneous response, high-throughput, and unfailing inference the task is even more challenging. There are various Approaches to solve an object detection problem.

The progression of object detection can be classified as Fig. 1. Traditional object detection methods particularly used extensively before 2014 includes Viola Jones (V-J), histogram of oriented gradients (HOG), and deep learning-based detection came into dominance after 2014 and is used extensively today [11]–[13]. The deep learning approach for object detection has evolved greatly over the past few years. Object detection under this approach can be grouped into one-stage detection, where speed is the main concern and two-stage detection with accuracy as the main concern. Object detection algorithms using regression include You only look once (YOLO) and single shot detector (SSD). These are simple and effective single-stage methods. Two-stage convolution neural network (CNN)-based systems have gained remarkable success in object detection. These techniques create region proposal networks. The region proposals are further divided into categories. RCNN uses selective search for the detection of object proposals. Every extracted proposal is then converted by rescaling to an image of a fixed size. This image is further given as input into a CNN model trained on Image Net. Inference time, energy use, efficiency (throughput/watt), and cost effectiveness are important trade-offs in selecting the hardware platform. A case study is presented in [14] in which face masks were detected on the following commercial off-the-shelf edge devices: Rpi 4, Intel NCS 2, Jetson Nano, Jetson Xavier NX, and i.MX 8M. It claims that Jetson Xavier NX platform is the best in terms of latency and efficiency (FPS/Watt) and Jetson Nano is the best in terms of cost. The embedded platforms used for implementation of object detection algorithm in this work include Rpi version 3b+ added with NCS and Jetson Nano.

2.1 Review on Architecture, Models, and Requirements for Edge Intelligence-based Applications

An edge device generally should be portable and use less power to deliver scalable architecture for the deep learning neural network. Edge intelligence-based applications need high inference accuracy. It also demands high inference

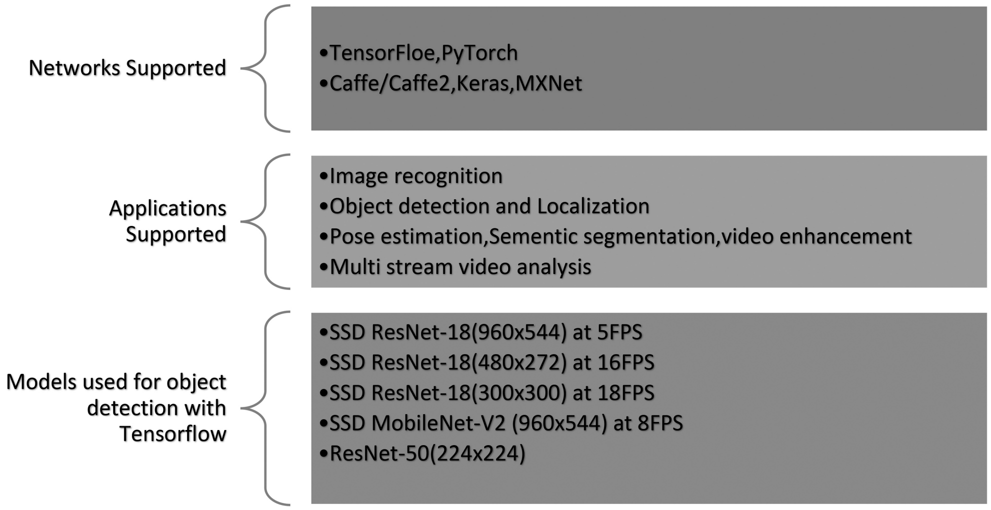

Networks Supported
•TensorFloe,PyTorch
•Caffe/Caffe2,Keras,MXNet
Applications Supported
•Image recognition
•Object detection and Localization
•Pose estimation,Sementic segmentation,video enhancement
•Multi stream video analysis
Models used for object detection with Tensorflow
•SSD ResNet-18(960x544) at 5FPS
•SSD ResNet-18(480x272) at 16FPS
•SSD ResNet-18(300x300) at 18FPS
•SSD MobileNet-V2 (960x544) at 8FPS
•ResNet-50(224x224)

Figure 2. Networks, models, and applications supported.

speed, increased throughput, and energy efficiency to achieve real-time performance. The ways to achieve this on embedded platform include the following.

1. Hardware efficient deep neural network design which includes DNN accelerators. These accelerators consider various hardware designs, such as ASIC, FPGAs, and GPUs, for improving the efficiency and speed of DNN inference and training processes.
2. Optimisation efforts on the software side, which includes compression of DNNs for less complexities, computation demands, and reduced memory footprints. Examples are the use of binary and ternary networks.

These solutions replace the floating-point multiplications which are hardware-intensive by logical operations, to make DNNs more efficient on hardware platforms. Hardware accelerators are limited by resources available to solve wide-ranging real-life applications. Most of the DNNs are not initially designed to be hardware efficient. The details of networks supported, applications, and models used for object detection with TensorFlow are listed in Fig. 2.

3. Raspberry pi 3 with Neural Compute Stick

The Rpi 3 Model B+ is the prototyping platform boasting a quad-core processor running at 1.4 GHz frequency, allowing operation in two frequency bands of 2.4 GHz and 5 GHz. It supports Bluetooth 4.2/Bluetooth Low Energy (BLE). It also has better Ethernet speed, and Power on Ethernet (PoE) capability through a separate attachment. The requirements for object detection implementation on Rpi include a camera, framework for object detection, a model, a data set, and OpenCV installed on the device. Tensor Flow object detection is the highly appreciated framework for creating a deep learning network. Tensor Flow Lite is a combination of functionalities used on embedded devices like Rpi to run TensorFlow models. It enables machine learning inference on devices. TensorFlow Lite removes the need of sending data backward and forward from a server by allowing inference on the edge device. This reduces the latency. It also helps improve privacy and the object detection can be performed without the need of Internet connectivity at low power. TensorFlow Lite uses quantised kernels to allow smaller and faster models. There are many pretrained models in this framework. The data sets used for training the models are KITTI data set, COCO data set, and Open Images data set. The TensorFlow Lite converter can convert a given TensorFlow model into lite version in an optimised flat buffer format identified by the .tflite file extension [15], [16].

Table 1
Models Pre-trained on COCO Data set

Pretrained Model	Speed in ms
ssd_mobilenet_v1_quantised_coco A	29
ssd_resnet_50_fpn_coco A	76
ssdlite_mobilenet_v2_coco	27
ssd_mobilenet_v2_quantised_coco	29
ssd_inception_v2_coco	42

Few pretrained models are listed in Table 1.

For an RGB image with 300×300 ×3 pixels, the quantised model will have each value as a single byte between 0 and 255. The Intel Movidius NCS Fig. 3 is a tiny fan USB device used for deep learning at the network edge. NCS is powered by the vision processing unit (VPU). The VPU has high performance at low power level. Use of NCS can extend the processing power of a Rpi to improve the performance of deep learning models. It supports frameworks like TensorFlow and Caffe. The model is trained on a development platform, including a laptop, desktop, or server, and the consequential binary files are deployed on the required embedded hardware.

Intel's neural compute SDK (NCSDK) can be used to support development as well as deployment of the deep learning model on Edge devices. Steps involved in object detection on pi 3 with NCS are as shown in Fig. 4. In

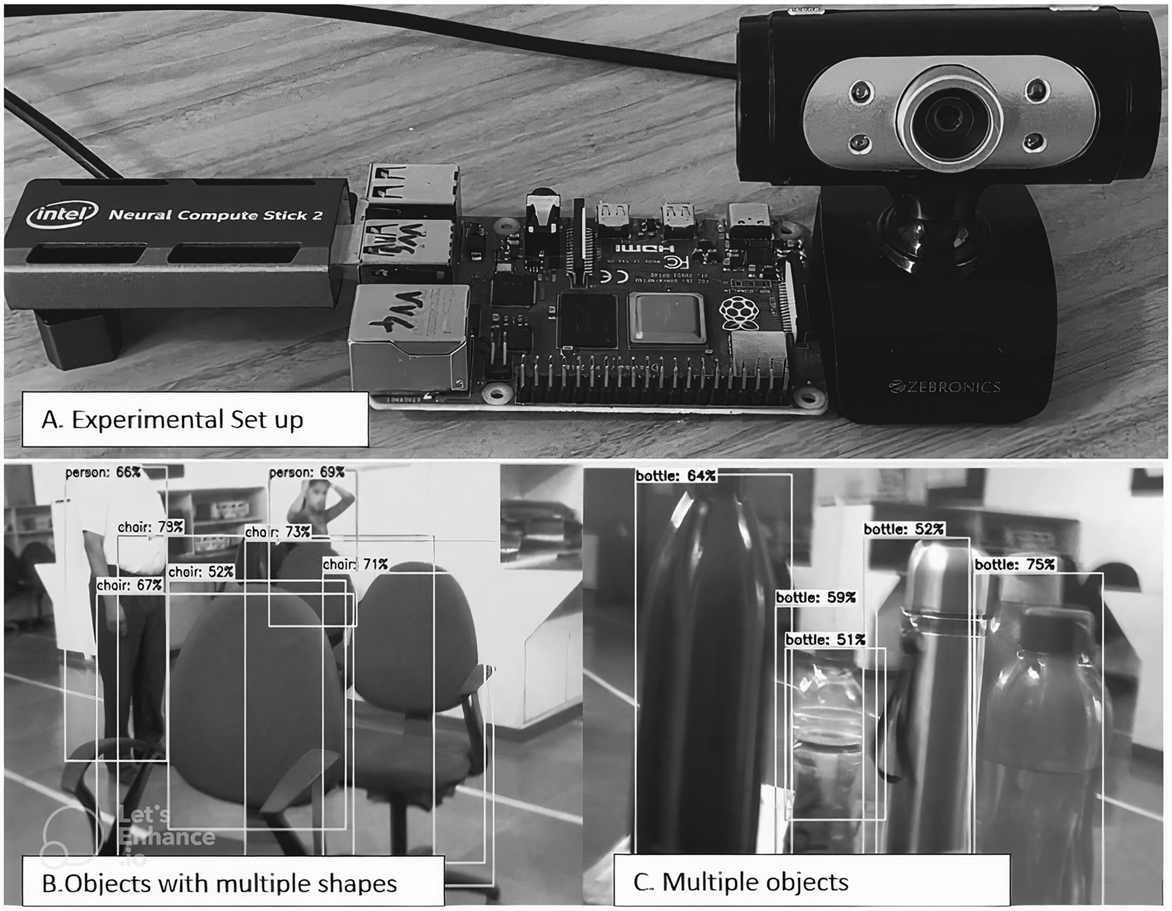


Figure 3. NCK connected to Rpi: (a) Experimental set-up; (b) Objects with different shapes; and (c) Multiple objects.

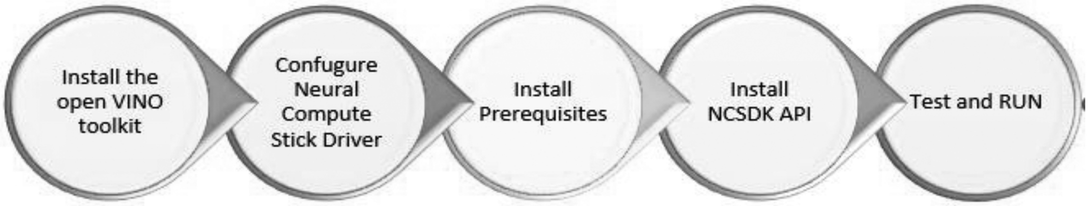

Install the open VINO toolkit
Confugure Neural Compute Stick Driver
Install Prerequisites
Install NCSDK API
Test and RUN

Figure 4. Steps involved in object detection on pi 3 with NCS.

the SDK mode, both the toolkit and API framework are installed on Rpi.

This is faster installation. The process of profiling and compiling the neural networks is to be done on a laptop or desktop.

The selected trained model is deployed on Rpi after tuning and compiling using SDK API.Intel's Open VINO toolkit streamlines deployment of multimedia-based applications to the network edge. It alters current applications into hardware-friendly solutions. The inference-optimised deployable packages work flawlessly at the edge [17], [18].

The required dependencies are installed on the device for image capturing. An environment is created to avoid version complexities. Various object detection models are installed and tested on MS-COCO data set. The results obtained are presented in Fig. 3 and the analysis is presented in Section 5.

4. Intelligent Edge Device: Jetson Nano

NVDIA Jetson Nano is an easy-to-use, powerful hardware platform which allows you to run in parallel multiple neural networks. Fig. 5 depicts important features of Jetson Nano. It brings affordable GPU on the edge. This can easily be used for computer vision-based applications using deep learning like pattern recognition, object detection and recognition.

Other important features include Micro-USB port, Gigabit Ethernet port, USB ports, HDMI output port. Connector for Display, DC Barrel jack in order to connect 5-V power input, camera connectors.

Jetson Nano can run a varied cutting-edge network, including machine learning frameworks like Caffe/Caffe2, TensorFlow, MXNet, PyTorch and others. These networks help development of applications by implementing strong

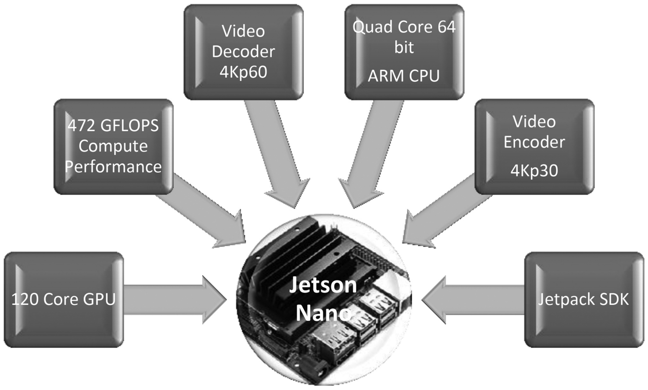


Figure 5. Important features of Jetson Nano.

capabilities such as object detection, image recognition, and localisation, semantic segmentation, pose estimation, video enhancement [19]–[21].

4.1 Jetpack SDK

NVIDIA Jetpack SDK is the all-inclusive solution for developing Artificial intelligence-based applications. Fig. 6 describes the important features of Jetpack SDK. NVIDIA's inference engine is based on TensorRT, which is run time optimised for forward propagation. This helps in accelerating the inferencing process. Features of TensorRT are depicted in Fig. 7. It can take a neural network trained on frameworks, optimise the neural network computation, and generate a lightweight run time engine which can be deployed to the edge. This will maximise the GPU performance.

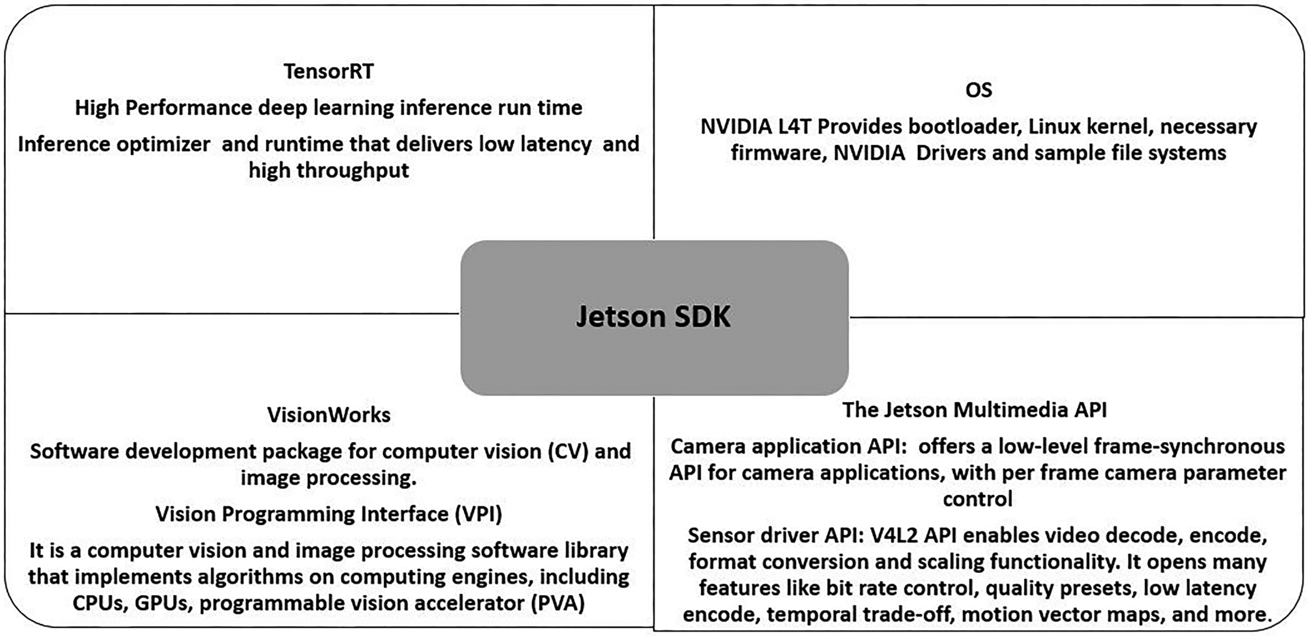
The inference optimisation process is classified depending upon the hardware optimiser, software optimisation, and model optimisation. The critical metrics for inference optimisation are throughput (number of images inferred/second), hardware cost, memory, energy consumption, and quality [18]–[23].

5. Edge Intelligence-based Object Detection and Recognition System on Jetson Nano

The experimental setup for the implementation of edge intelligence-based object detection is shown in Fig. 8. The setup uses Jetson Nano 4GB, USB Web camera, 64GB SD Card with Jetpack 4.4 and NVIDIA Tensor-RT accelerator library installed. The Pre-trained models used for analysis are Res-Net-18, Res-Net-50,ssd_mobilenet_v1_coco, ssd_mobilenet_v2_coco. The steps followed are conversion of the TensorFlow model to Universal Framework Format and generation of a Tensor-RT execution engine. The TensorFlow Object Detection API is installed using Galliot's Docker container. Various models belonging to the single-shot detection category are deployed and tested. This model is pre-trained on the MS COCO image data set over 91 different classes. The model is pre-trained on common objects. The performance is tested under various conditions for objects in day-to-day life required by a visually impaired person including bottle, book, Pen, Mobile, Laptop, Seizer, bowl, *etc.* Sample results of detection on Jetson Nano are shown in Fig. 8.

Performance comparison of Rpi with NCS and Jetson Nano is carried out for Mobilenet v2 model. Average values of precision, Accuracy, and recall are calculated after 50 individual tests. The performance metrics used are as in Fig. 9, where TP is true positive, TN is true negative, FP is false positive, and FN false negative.

Objects of different shapes and sizes are used. Table 2 presents the average parameters for comparison. It is observed that there is a reduction of 10–30% in detection accuracy with variation in light intensity.



TensorRT

High Performance deep learning inference run time

Inference optimizer and runtime that delivers low latency and high throughput

OS

NVIDIA L4T Provides bootloader, Linux kernel, necessary firmware, NVIDIA Drivers and sample file systems

Jetson SDK

VisionWorks

Software development package for computer vision (CV) and image processing.

Vision Programming Interface (VPI)

It is a computer vision and image processing software library that implements algorithms on computing engines, including CPUs, GPUs, programmable vision accelerator (PVA)

The Jetson Multimedia API

Camera application API: offers a low-level frame-synchronous API for camera applications, with per frame camera parameter control

Sensor driver API: V4L2 API enables video decode, encode, format conversion and scaling functionality. It opens many features like bit rate control, quality presets, low latency encode, temporal trade-off, motion vector maps, and more.

Figure 6. Jetpack features.

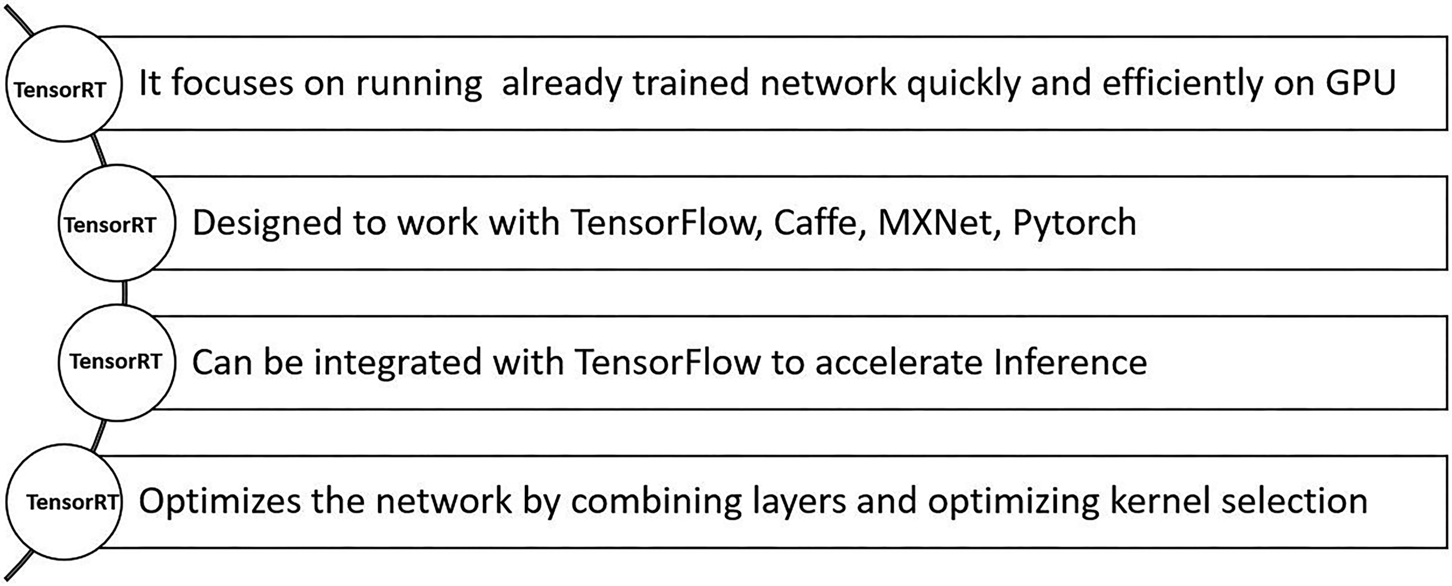

TensorRT
It focuses on running already trained network quickly and efficiently on GPU
TensorRT
Designed to work with TensorFlow, Caffe, MXNet, Pytorch
TensorRT
Can be integrated with TensorFlow to accelerate Inference
TensorRT
Optimizes the network by combining layers and optimizing kernel selection

Figure 7. Tensor-RT features.

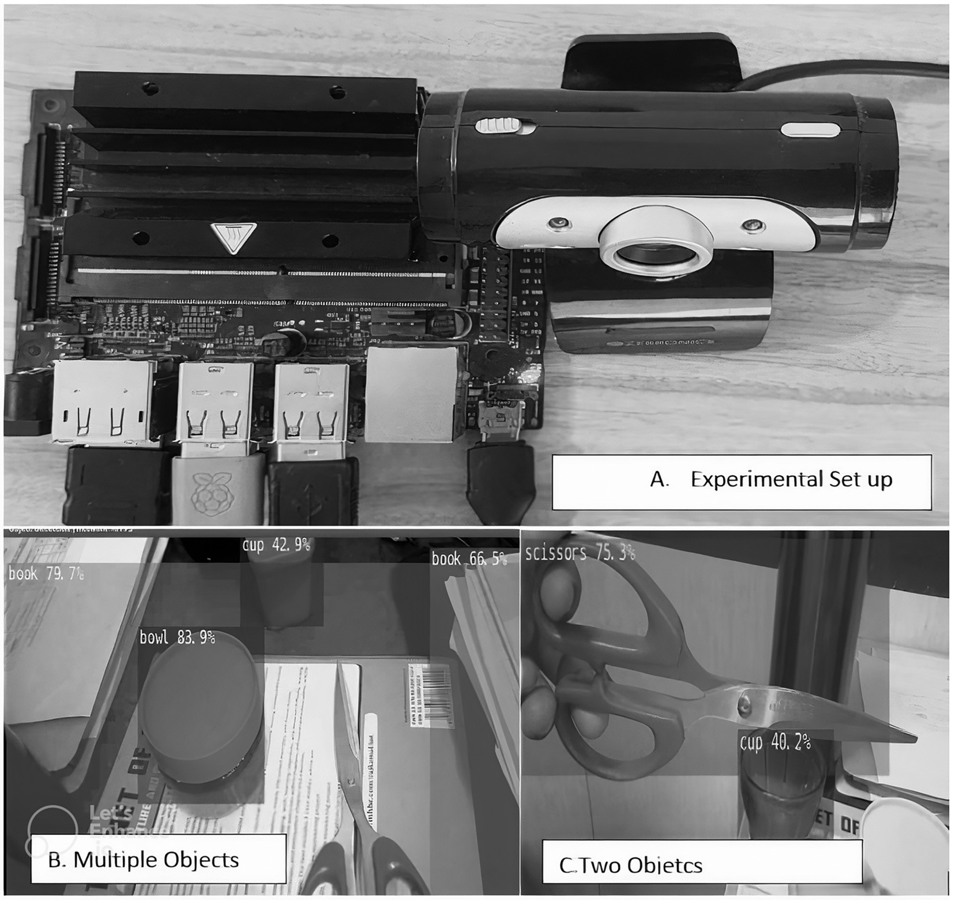


Figure 8. Experimental setup and results for object detection on Nano: (a) Experimental Set-up; (b) Multiple objects; and (c) Two objects.

Time per inference with Tensor-RT framework is checked for different object detection models. Table 3 presents the comparison of time per inference for Nano and Rpi with NCS. Energy per inference for MobileNet V2 was found 98 for both the devices.

6. Conclusion

Developing efficient edge intelligent object detection applications is a non-trivial and challenging task. Investigation of performance for variation in the hardware devices, object

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Figure 9. Performance metrics used for analysis.

Table 2
Comparison of Performance of Nano and Rpi with NCS with MobileNet v2

Parameter	Rpi with NCS	Jetson Nano
Average precision	82%	83%
Average Recall	0.9	0.92
Average Accuracy	78%	81%
objects detected in a frame	7	6

Table 3
Comparison of Time per Inference(msec) for Nano and Rpi with NCS

Model	Rpi with NCS	Jetson Nano
Res-Net-18	23	19
Res-Net-50	51	33
SSD Mobile Net V1	190	32
SSD Mobile Net V2	51	18
Alex Net	91	46

detection models, libraries, optimisation techniques, and tools are necessary. In this paper, two different state-of-the-art hardware platforms Rpi with NCS and Jetson Nano are compared. Five pre-trained COCO object detection models were evaluated and compared in terms of time and energy per inference. Insights from the study lead users to knowingly choose their required package (i.e., hardware and model) for a specific edge application. Inference time on edge devices vary significantly with object detection model. SSD Mobile Net V2 model is the fastest of all on Tensor-RT framework. Res-Net 18 is best suitable model for Rpi with NCS. Jetson Nano performs better as compared to other options. Thus, for mission critical IoMT applications Jetson Nano proves to be a better choice. Use of Dockers simplifies the deployment process on the edge platforms like Nano as well as Rpi and has a great scope in IoMT applications.

References

[1] Z. Zou, Z. Shi, Y. Guo, and J. Ye, Object detection in 20 years: A survey, ArXiv abs/1905.05055, 2019.

[2] A.A. Sheeraz, A. Bilal, A.S. Ghalib, A. Luigi, and M. Waqar, Internet of multimedia things: Vision and challenges, *Ad Hoc Networks, 33*, 2015, 87–111.

[3] V. Mazzia, A. Khaliq, F. Salvetti, and M. Chiaberge, Real-time apple detection system using embedded systems with hardware accelerators: An edge AI application*, IEEE Access, 8*, 2020, 9102–9114.

[4] L. Mingming and G. Lutao, Automatic classification of apple leaf diseases based on transfer learning, *International Journal of Robotics and Automation, 37*(1), 2022, 44–51, DOI: 10.2316/J.2022.206-0742.

[5] X. Bangquan and W.X. Xiong, Real-time embedded traffic sign recognition using efficient convolutional neural network, *IEEE Access, 7,* 2019, 53330–53346.

[6] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A.Y. Zomaya, Edge intelligence: The confluence of edge computing and artificial intelligence, *IEEE Internet of Things Journal, 7,* 2020, 74577469. doi:10.1109/jiot.2020.2984887.

[7] Z.-F. Xu, R.-S. Jia, Y. Liu, C.-Y. Zhao, and H.-M. Sun, Fast method of detecting tomatoes in a complex scene for picking robots, *IEEE Access, 8,* 2020, 55289–55299.

[8] M. Risso, A. Burrello, F. Conti, L. Lamberti, Y. Chen, L. Benini, E. Macii, M. Poncino, and D.J. Pagliari, Lightweight neural architecture search for temporal convolutional networks at the edge, *IEEE Transactions on Computers, 72*(3), 2023, 744–758.

[9] C. Dong, C. Pang, Z. Li, X. Zeng, and X. Hu, PG-YOLO: A novel lightweight object detection method for edge devices in industrial Internet of Things, *IEEE Access, 10*, 2022, 123736–123745.

[10] C. Zhang, R. Li, W. Kim, D. Yoon, and P. Patras, Driver behavior recognition via interwoven deep convolutional neural nets with multi-stream inputs, *IEEE Access, 8*, 2020, 191138–191151.

[11] C. Guo, K. Huang, L. Yarong, Z. Huyin, and Z. Wenwei, Object-oriented semantic mapping and dynamic optimization on a mobile robot, *International Journal of Robotics and Automation,* 2022, 321–331, DOI: https://doi.org/ 10.2316/J.2022.206-0498.

[12] L. Yuri, H. Menghan, Z. Guangtao, and X.Y. Simon, LSNet: Identification of copper and stainless steel using laser speckle imaging in dismal surroundings, *International Journal of Robotics and Automation, 36,* 2021, 256–263, DOI: https://doi.org/ 10.2316/J.2021.206-0568.

[13] P.N. Huu, T.P. Ngoc, and T.L.T. Hai, Developing real-time recognition algorithms on Jetson Nano hardware. in *Intelligent Systems and Networks. Lecture Notes in Networks and Systems*, N.L. Anh, S.J. Koh, T.D.L. Nguyen, J. Lloret, T.T. Nguyen, (eds), vol 471. Singapore: Springer, 2022.

[14] A. Khandewale, V. Gohokar, and P. Nawandar, Edge intelligence-based object detection system using neural compute stick for visually impaired people, *Proc. Information and Communication Technology for Intelligent Systems, Singapore,, 2020*, 433–439.

[15] C. Gao, A. Rios-Navarro, X. Chen, S.-C. Liu, and T. Delbruck, Edge DRNN: Recurrent neural network accelerator for edge inference, *IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 10*(4), 2020, 419–432.

[16] L. Liang, H. He, J. Zhao, C. Liu, Q. Luo, and X. Chu An erasure-coded storage system for edge computing, *IEEE Access, 8*, 2020, 96271–96283.

[17] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, Edge intelligence: Paving the last mile of artificial intelligence with edge computing, *Proceedings of the IEEE, 107*, 2019, 1738–1762.

[18] Y. Zhang, S. Wang, L. Lu, L. Liu, L. Xu, and W. Shi, Open EI: An open framework for edge intelligence, *Proc. IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, Dallas, TX, 2019, 1840–1851.

[19] F. Tsai and J.Y. Huang, Predicting canine posture with smart camera networks powered by the artificial intelligence of things, *IEEE Access, 8*, 2020, 220848–220857.

[20] P. Adarsh, P. Rathi, and M. Kumar, YOLO v3-Tiny: Object detection and recognition using one stage improved model, *Proc. 6th International Conference on Advanced Computing and Communication Systems*, Coimbatore, 2020, 687–694.

[21] Y. Guo, B. Zou, J. Ren, Q. Liu, D. Zhang and Y. Zhang, Distributed and efficient object detection via interactions among devices, edge, and cloud, *IEEE Transactions on Multimedia, 21*(11), 2019, 2903–2915.

[22] M. Nazeer, M. Qayyum, and A. Ahad, Real time object detection and recognition in machine learning using Jetson Nano, *International Journal from Innovative Engineering and Management Research, 11,* 2022, 1–7.

[23] E. Güney, C. Bayilmiş, and B. Çakan, An implementation of real-time traffic signs and road objects detection based on mobile GPU platforms, *IEEE Access, 10*, 2022, 86191–86203.

Biographies

Vinaya Gohokar received the Ph.D. degree in Electronics Engineering from Amaravati University in 2009. She is working as a Professor with the School of ECE, MIT World Peace University, Pune. With over 30 years of teaching experience, her areas of research include Internet of Things, computer vision, edge intelligence, *etc.* Six research scholars have received doctoral degrees under her guidance. Her research grants worth Rs. 50 lakhs from various funding agencies to her credit.

Vijay Gohokar received the Ph.D. degree from Amravati University. He is working as a Principal with MMCOE Pune. His area of expertise include power system, switchgear and protection, distribution automation, and smart grid.