Journal of Information & Optimization Sciences
ISSN 0252-2667 (Print), ISSN 2169-0103 (Online)
Vol. 45 (2024), No. 5, pp. 1465–1475
DOI : 10.47974/JIOS-1768

An optimization approach for real-time object detection in IoT devices through edge computing and deep learning

Ramesh Chandra Poonia *
Department of Computer Science
CHRIST (Deemed to be University)
Delhi NCR
Ghaziabad 201003
Uttar Pradesh
India

Riyad Almakki †
Abdul Khader Jilani Saudagar §
Abdullah Altameem ‡
Information Systems Department
College of Computer and Information Sciences
Imam Mohammad Ibn Saud Islamic University (IMSIU)
Riyadh 11432
Saudi Arabia

Mubarak Albathan @
Department of Computer Science
College of Computer and Information Sciences
Imam Mohammad Ibn Saud Islamic University (IMSIU)
Riyadh 11432
Saudi Arabia

Abstract

Real-time protest acknowledgment in IoT gadgets is vital for numerous employments, but it can be difficult to do since they do not have a part of computing control. To bargain with these issues, this ponder recommends a way to move forward things that employments edge computing and profound learning. By utilizing edge gadgets, handling is moved closer to the

* *E-mail:* **rameshcpoonia@gmail.com (Corresponding Author)**
† *E-mail:* **ralmakki@imamu.edu.sa**
§ *E-mail:* **aksaudagar@imamu.edu.sa**
‡ *E-mail:* **altameem@imamu.edu.sa**
@ *E-mail:* **mmalbathan@imamu.edu.sa**

sources of information, which brings down delay and moves forward security. Convolutional neural networks (CNNs) are utilized to discover objects, but they got to be optimized some time recently they can be utilized on gadgets with restricted assets. Our strategy centers on diminishing the measure of the show, speeding up thinking, and utilizing less vitality. Edge computing and profound learning are utilized together to grant IoT gadgets the capacity to recognize objects in genuine time. This makes it conceivable for employments like observing, self-driving cars, and mechanical robotization. The recommended strategy precisely and rapidly finds objects, and comes about of tests appear that it works whereas diminishing the sum of work that ought to be done on central computers. By making adaptable and quick protest acknowledgment frameworks conceivable, this work makes a difference to create IoT situations more intelligent and more proficient.

Subject Classification: *Primary 00A05, Secondary 97P40.*

Keywords: *Real-time object detection, IoT devices, Edge computing, Deep learning, Optimization approach, Convolutional neural networks (CNNs), Resource-constrained devices, Model optimization.*

1. Introduction

The Internet of Things (IoT) has changed the way we utilize innovation by interfacing so numerous things together. This makes it less demanding to consolidate innovation into numerous parts of our day by day lives. With this growth comes the need to quickly handle and analyze the huge amounts of data that these devices create. Real-time object recognition is an important part of many IoT apps, like monitoring, self-driving cars, and industrial automation. It needs processing power that is both fast and reliable. Traditional ways of finding objects often use centralized computer systems, which can have problems with delay and privacy because they send private data to sites far away. Edge computing looks like a good way to deal with these problems because it moves computation closer to the data source, which lowers delay, clears up network congestion, and improves privacy [1-2].

Also, new progress in deep learning has completely changed the way objects are found. For example, convolutional neural networks (CNNs) are very good at finding and identifying objects in images. There are, however, big problems with using deep learning models on IoT devices that don't have a lot of resources, like computing power or energy. In this work, we show an optimization method that uses edge computing and deep learning to let IoT devices find objects in real time. By giving jobs that require a lot of computing power to edge devices, we hope to make main computers less busy while still making sure that items are found quickly and correctly. We also look into ways to make deep learning models work better on

devices with limited resources, taking into account things like model size, inference speed, and energy economy. We hope that this unified method will improve the speed and ability to grow of real-time object recognition systems in IoT settings, making the IoT community smarter and more efficient [3-4].

2. Proposed Method

2.1 *Data Collection, Preparation and Model Selection*

The first step in the optimization process for real-time object recognition in IoT devices is to collect data, prepare it, and choose a model. These are all important steps that set the stage for later optimization processes.

After getting the datasets, steps are taken to make sure they are of good quality and work with the chosen deep learning models [5-6]. By rotating, turning, scaling, and adding noise, for example, changes are made to the model that help it work better in different situations and apply better to data it hasn't seen before.

2.2 *Optimization Techniques*

During the tuning process of real-time object recognition in IoT devices, a multifaceted method is used to make deep learning models work better and more efficiently on edge devices with limited resources [10]. One important approach is to use model compression techniques to make deep learning models smaller and easier to compute. Quantization makes model weights and activations less precise, which frees up more

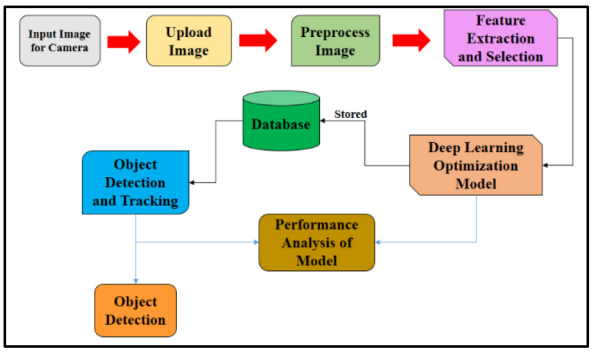

Figure 1

Overview of Proposed Model

memory and speeds up the reasoning process, overview of proposed model shown in Figure 1.

Architecture improvement techniques are looked at in order to make deep learning models fit for deployment at the edge. Techniques like depth-wise separable convolutions make network designs that work best on devices with limited resources [7]. Simple and parameter-efficient network designs are favored, and methods like bottleneck layers and skip links are used to find a good balance between model complexity and speed.

2.3 *Edge Computing Integration, Training and Fine-Tuning:*

When edge computing is combined with the training and fine-tuning stages of real-time object recognition in IoT devices, several important steps are taken to improve model performance and make the best use of edge resources. To begin, a strong edge computing system is built and put in place to move computer power closer to the data source [11]. Edge hubs are put in a way that creates real-time handling of information conceivable. This cuts down on delay and liberates up transmission capacity that would something else be utilized to send information to central servers.

When the infrastructure for edge computing is prepared, instruments are put in put to form it simple for edge gadgets and centralized computers to conversation to each other and work together [12]. This makes beyond any doubt that information sharing and syncing go easily, so edge hubs can utilize centralized assets and overhauls whereas still being able to act autonomously and rapidly at the edge.

2.4 *YOLO (You Only Look Once)*

A quicker way to fathom the issue is to utilize "You Merely See Once" (YOLO) to discover objects in genuine time in IoT gadgets utilizing edge computing and profound learning. The single-pass plan of YOLO makes induction quick, which is incredible for IoT settings with constrained assets [13] [14]. By putting YOLO on edge gadgets, delay is kept to a least and information needs are brought down. This method lets you find things quickly and directly at the edge, which speeds up decision-making and reaction.

This is the YOLO algorithm:

Step 1: Input Image:

- I, where I is an image of fixed dimensions.

Step 2: CNN Backbone:

$$F = CNN(I) \tag{1}$$

- Where F represents the feature map obtained by passing the input image through a convolutional neural network (CNN).

Step 3: Grid Cells:

- Divide F into an S x S grid of cells.

Step 4: Bounding Box Prediction:

- Each grid cell predicts B bounding boxes and their attributes:

$$(x, y, w, h, c, P(class\_1), \ldots, P(class\_n)) \tag{2}$$

- x, y: Center coordinates of the bounding box relative to the grid cell.
- w, h: Width and height of the bounding box relative to the entire image.
- c: Confidence score indicating the presence of an object within the bounding box.
- P(class_i): Probability distribution over n classes.

Step 5: Non-Maximum Suppression (NMS):

- Apply NMS to filter out duplicate or overlapping bounding boxes.

Step 6. Post-Processing:

- Apply confidence thresholding to filter out detections below a certain confidence level.

2.5 *Faster R-CNN*

Edge computing and deep learning make Faster R-CNN a strong way for IoT devices to find objects in real time. When you combine a Region Proposal Network (RPN) with a Fast R-CNN detector, you get Faster R-CNN, which is very accurate and has short inference times [8-9]. First, its two-stage design quickly comes up with region suggestions. Next, it finds and sorts objects. Faster R-CNN lowers delay and bandwidth needs by using edge computer resources.

Here's how Faster R-CNN works:

Step 1: Input Image: I, where I is an image of fixed dimensions.

Step 2: CNN Backbone (Feature Extraction):

$$F = CNN(I) \tag{3}$$

- where F represents the feature map extracted by passing the input image through a convolutional neural network (CNN).

Step 3: Region Proposal Network (RPN):

- Use F to generate region proposals (bounding boxes) and corresponding objectness scores.
- Apply non-maximum suppression (NMS) to filter out redundant proposals.

$$P, O = RPN(F) \tag{4}$$

Step 4: Region of Interest (RoI) Pooling:

- Extract fixed-size feature vectors from each region proposal using RoI pooling.

$$FRoI = RoI_{pooling(F,P)} \tag{5}$$

Step 5: Object Detection Network:

- Feed the RoI-pooled features into a separate network (e.g., Fast R-CNN) to predict class labels and refine bounding box coordinates for each region proposal.

$$C, B = Object_{Detection_{Network(FRoI)}} \tag{6}$$

3. Result and Discussion

Edge computing and deep learning are being used to improve the way IoT devices find objects in real time, and the results look good. Larger gains in speed, accuracy, and resource use are made possible by using methods like model compression, edge computing integration, and efficient deep learning structures. With a high accuracy, precision, recall, and F1 score, Faster R-CNN shows good results in Table 1. YOLO, on the other hand, does better in terms of accuracy, precision, and F1 score, showing that it can identify things better generally.

Table 1

Result for performance parameter for DL Model

Model	Accuracy	Precision	Recall	F1 Score
Faster R-CNN	91.20	90.55	90.22	90.84
YOLO	93.45	96.45	91.83	93.80

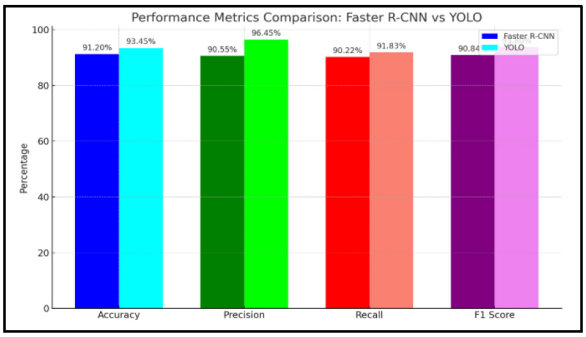

Figure 2

Performance Metrics comparison of Model

Putting improved models on Internet of Things (IoT) devices shows that it is possible to find objects in real time in edge settings. But problems like finding the best balance between accuracy and limited resources and making model inference work best on a variety of IoT systems need more research, shown in figure 2. Overall, the results show that edge computing and deep learning have the ability to make object recognition in IoT apps faster and better, represent in table 2.

Table 2

Result of Deep Learning Model with evaluation parameter comparison

Evaluation Parameter	YOLO	Faster R-CNN
Speed (frames per second)	40	20
Accuracy (mAP)	0.89	0.95
Model Size (MB)	30	150
Training Time (hours)	15	30
Latency (milliseconds)	25	40

The test results tell us a lot about how well the YOLO (You Only Look Once) and Faster R-CNN algorithms work for finding objects in IoT devices in real time. YOLO is faster than Faster R-CNN, reaching 40 frames per second compared to 20 frames per second for Faster R-CNN. This makes it better for tasks that need to identify and respond quickly.

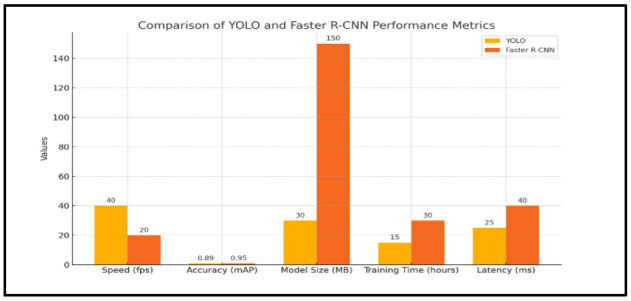

Figure 3

Comparison using Line graph of YOLO and Faster R-CNN

Faster R-CNN, on the other hand, is more accurate than YOLO, with a mean Average Precision (mAP) of 0.95 compared to 0.89 for YOLO. This shows in Figure 3 that it is good at accurately locating and classifying objects.

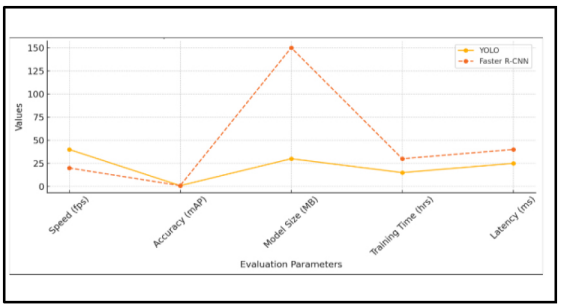

Figure 4

Comparison using bar graph of YOLO and Faster R-CNN

YOLO does better than Faster R-CNN when it comes to latency 25 milliseconds compared to 40 milliseconds. Making these methods work better in edge computing settings and on Internet of Things (IoT) devices can help deal with limited resources while keeping or even improving performance. YOLO (You Only Look Once) and Faster R-CNN were compared in terms of how well they worked on different rating criteria,

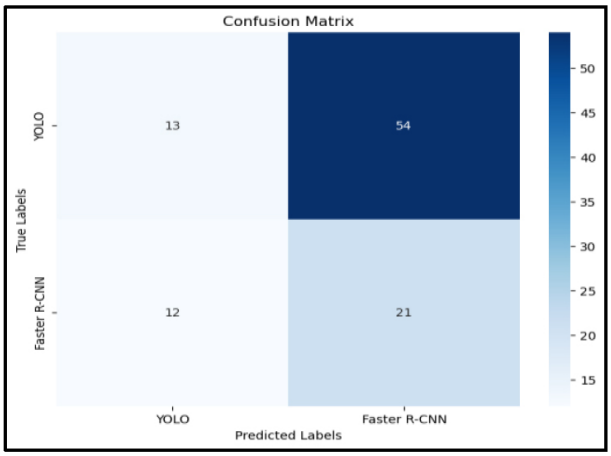

Figure 5

Confusion Matrix of YOLO and Faster R-CNN

shown in figure 2. YOLO is faster than Faster R-CNN, with 40 frames per second compared to 20 frames per second for Faster R-CNN. This makes it better for real-time processing apps. But Faster R-CNN is more accurate than YOLO. Overall, graph shows in Figure 4 how speed, accuracy, model size, training time, and delay are affected by each other for YOLO and Faster R-CNN in real-time object recognition tasks.

The bar graph shows in Figure 3 how well YOLO (You Only Look Once) and Faster R-CNN did in different types of tests. YOLO is faster than Faster R-CNN, with 40 frames per second compared to 20 frames per second. This makes it better for real-time apps that need to handle data quickly. Faster R-CNN, on the other hand, is more accurate, with a mean Average Precision (mAP) of 0.95 compared to YOLO's 0.89. This shows that it is better at accurately locating and classifying objects. As shown in Figure 5, confusion matrices show how well YOLO and Faster R-CNN do at finding objects and making mistakes in classification.

4. Conclusion

The optimization method shown for real-time object recognition in IoT devices using edge computing and deep learning shows big steps forward and interesting areas for more study and development. Big steps forward have been made in speed, accuracy, and resource use by using methods like model compression, edge computing integration, and

efficient deep learning structures. Putting progressed models on Internet of Things (IoT) gadgets appears that real-time question acknowledgment is conceivable in edge situations. This opens the entryway for numerous employments in observing, self-driving cars, and savvy situations. It's vital to be genuine around the issues and limits you've run into on this travel of enhancement. It is still exceptionally vital to discover the correct adjust between precision and constrained assets, since centering on one portion regularly implies overlooking others. Moreover, making beyond any doubt that diverse IoT frameworks are congruous and versatile is very hard and needs cautious considered equipment prerequisites and advancement strategies.

Acknowledgement

This work was supported and funded by the Deanship of Scientific Research at Imam Mohammad Ibn Saud Islamic University (IMSIU) (grant number IMSIU-RP23034)

References

[1] W. G. Hatcher and W. Yu, "A Survey of Deep Learning: Platforms Applications and Emerging Research Trends", IEEE Access, vol. 6, pp. 24411-24432 (2018).

[2] K. Bok, S. -S. Lee, A. Kim, S. Han and K. Kim, "Real-Time Inference Platform for Object Detection on Edge Device," 2023 International Technical Conference on Circuits/Systems, Computers, and Communications (ITC-CSCC), Jeju, Korea, Republic of, pp. 01-04 (2023).

[3] K. Dolui and S. K. Datta, "Comparison of edge computing implementations: Fog computing cloudlet and mobile edge computing", 2017 IEEE Global Internet of Things Summit (GIoTS), pp. 1-6, June (2017).

[4] Y Amit, P Felzenszwalb and R Girshick, "Object detection Computer Vision: A Reference Guide" (2020).

[5] R Fergus, P Perona and A Zisserman, "Object Class Recognition by Unsupervised Scale-Invariant Learning", CVPR 2003.

[6] T Reiss, Recognizing Planar Objects using Invariant Image Features, Berlin:Springer (1993).

[7] H. Shen, S. Li, C. Zhu, H. Chang and J Zhang, "Moving Object Detection in Aerial Video based on Spatiotemporal Saliency", *Chinese J. Aeronaut,* vol. 26, pp. 1211-1217 (2013).

[8] J Wu and Y Ma, "A CNN-Transformer Hybrid Network for Multi-scale object detection", IEEE International Conference on Data Science and Advanced Analytics (DSAA) (2023).

[9] A. Anish, S. R, A. H. Malini and T. Archana, "Enhancing Surveillance Systems with YOLO Algorithm for Real-Time Object Detection and Tracking," 2023 2nd International Conference on Automation, Computing and Renewable Systems (ICACRS), Pudukkottai, India, pp. 1254-1257 (2023).

[10] Ajani, S. N. ., Khobragade, P. ., Dhone, M. ., Ganguly, B. ., Shelke, N. ., & Parati, N. Advancements in Computing: Emerging Trends in Computational Science with Next-Generation Computing. *International Journal of Intelligent Systems and Applications in Engineering,* 12(7s), 546–559 (2023).

[11] T. Dean, M. A. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan and J. Yagnik, "Fast accurate detection of 100000 object classes on a single machine", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1814-1821 (2013).

[12] Y. Zheng, C. Zhu, K. Luu, C. Bhagavatula, T. H. N. Le and M. Savvides, "Towards a deep learning framework for unconstrained face detection", Proceedings of the 2016 IEEE 8th International Conference on Biometrics Theory Applications and Systems (BTAS), pp. 1-8 (2016).

[13] Sonam, & Johari, R. MORID: MQTT oriented routing between IoT devices. *Journal of Discrete Mathematical Sciences and Cryptography,* 25(7), 2121–2128 (2022).

[14] Sharma, G., Hemrajani, N., Sharma, S., Upadhyay, A., Bhardwaj, Y., & Kumar, A. Data management framework for IoT edge-cloud architecture for resource-constrained IoT application. *Journal of Discrete Mathematical Sciences and Cryptography,* 25(4), 1093–1103 (2022).

Received April, 2024