Bulletin of Electrical Engineering and Informatics
Vol. 14, No. 1, February 2025, pp. 357∼365
ISSN: 2302-9285, DOI: 10.11591/eei.v14i1.8131

Object detection in video surveillance using MobileNetV2 on resource-constrained low-power edge devices

Harshad Lokhande, Sanjay R. Ganorkar
Department of E&TC Engineering, Sinhgad College of Engineering, Savitribai Phule Pune University, Pune, India

Article Info

Article history:

Received Jan 2, 2024
Revised Sep 17, 2024
Accepted Sep 28, 2024

Keywords:

Convolutional neural network
Edge computing
Low-power devices
Microcontrollers
Neuromorphic edge
TinyML

ABSTRACT

Edge-based video surveillance systems encounter significant obstacles in object detection due to the limited computational power and energy efficiency of edge devices, which are required to deliver real-time processing capabilities. Traditional object detection models are excessively resource-hungry for these environments, making optimization techniques absolutely essential. This study robustly explores the implementation of quantized transfer learning utilizing SSD MobileNet V2 with 8-bit quantization to significantly elevate the performance of object detection on resource-constrained edge devices. Experimental results decisively indicate that the Raspberry Pi 5 and Nvidia Jetson Orin Nano surpass other devices, achieving total latencies of 5 ms and 85 ms, respectively, affirming their exceptional suitability for real-time applications. The quantized int8 model secures an impressive accuracy of 80.65% while dramatically lowering memory consumption and latency when compared to the unoptimized int32 model. Furthermore, the model demonstrates outstanding performance on a masked-unmasked dataset with an F1 score of 0.92 for masked detection. These findings underscore the transformative potential of quantization in enhancing both inference speed and resource efficiency in edge-based surveillance systems. Future research will boldly investigate advanced hybrid quantization strategies and architectural enhancements to achieve an optimal balance of efficiency and accuracy, alongside scalability across a broader spectrum of edge devices and datasets.

Corresponding Author:

Harshad Lokhande
Department of E&TC Engineering, Sinhgad College of Engineering
Savitribai Phule Pune University, Pune, Maharashtra, 411041, India
Email: lokhande.harshad@gmail.com

1. INTRODUCTION

Object detection in remote surveillance presents significant challenges, especially when implemented on edge devices with limited computational power and energy efficiency. These devices are undoubtedly expected to execute real-time processing, which intensifies the complexity of the task, as traditional models require more resources than edge devices can deliver. To effectively tackle these challenges, optimization techniques such as model pruning, quantization, and hardware acceleration have been decisively adopted to enhance both performance and energy efficiency in edge computing environments [1].

In smart city environments, where continuous video surveillance is non-negotiable, depending solely on centralized cloud infrastructures for processing can create unacceptable latency and reliance on stable internet connectivity, which is not always guaranteed. To decisively counter these issues, hybrid frameworks that merge edge and cloud computing have been proposed. By relocating computation closer to the data source, these edge-cloud collaborative frameworks drastically reduce latency and elevate system responsiveness. Alongside computational efficiency, privacy concerns are of utmost importance when deploying object detection models in edge environments, particularly in sensitive applications like autonomous driving and surveillance. To effectively uphold data privacy while ensuring robust performance, techniques such as federated learning and split learning have emerged as powerful solutions [2]. Edge devices frequently encounter challenges in executing complex deep learning models due to hardware constraints, leading to noticeable performance bottlenecks. This has propelled the advancement of optimization methods that create lightweight models specifically tailored for edge environments. Recent breakthroughs in model optimization for edge devices encompass techniques such as TinyML, which emphasizes running highly compressed models on ultra-low-power devices, and neural architecture search (NAS), which automates the design of efficient neural networks optimized for particular hardware constraints. MobileNetV3 and EfficientNet-Lite represent other recent advancements, engineered to elevate performance and accuracy on resource-limited edge devices through advanced architecture search and compound scaling techniques [3].

In addition to these methods, dynamic inference techniques such as EdgeTPU-based quantized models and Neural Accelerator-driven optimizations have gained significant traction. These techniques enable models to modify their computational complexity based on available resources, further enhancing energy efficiency without compromising performance. Moreover, zero-shot NAS is rapidly becoming a favored method for identifying lightweight models that adapt seamlessly to the specific computational limits of edge devices.

Our research introduces an unparalleled real-time solution for object detection in video streams, specifically designed for low-power edge computing devices. This section delivers a sharp review of existing systems for object detection on edge devices, showcasing a variety of optimization techniques employed to evaluate the performance of 32-bit and 8-bit quantization models. The following chapter investigates the methodology, concentrating on the architectural framework and mathematical analysis that scrutinize the trade-offs between model size and accuracy. Moreover, the findings are rigorously compared with previously explored optimization techniques to deliver a thorough assessment of their effectiveness in boosting model performance on resource-constrained hardware.

2. RELATED WORK

The demand for deep learning on edge devices drives research into model compression. Limited resources on these devices require methods that shrink models while keeping performance intact. Techniques like pruning, quantization, knowledge distillation, and federated learning compression each have unique advantages and challenges for edge AI [4]. Pruning is essential for model compression, targeting the removal of less significant weights or neurons. Global pruning targets the least important weights overall, leading to slight accuracy gains and smaller models. Research shows that network pruning can drastically reduce parameters and FLOPs in models like MobileNetV2 without sacrificing accuracy. Moreover, magnitude-based and Taylor pruning effectively reduce weights based on their significance, balancing size and performance in tasks such as audio classification [5].

Quantization stands out as a highly effective compression method. By skillfully reducing parameter precision, typically transforming 32-bit floating-point values into 8-bit integers, quantization significantly lowers memory usage and accelerates inference speed [6]. Nonetheless, the challenge of accuracy degradation persists, with certain applications experiencing an accuracy decline of up to 13.75% after implementing 8-bit quantization [7]. Regardless, the technique's straightforwardness and efficiency solidify its status as a favored option for compressing models on edge devices.

Knowledge distillation entails training a more compact ”student” model to emulate the performance of a larger ”teacher” model. This strategy facilitates substantial model size reduction with negligible impact on accuracy, making it exceptionally well-suited for deployment across diverse edge environments. In distributed inference systems, knowledge distillation consistently demonstrates its ability to preserve high performance levels while simultaneously reducing computational demands [7].

Federated learning compression effectively tackles the issues of privacy and communication within decentralized training settings. The FedComp framework employs tensor-wise index-sharing and meticulous parameter packing to significantly lessen memory and communication overhead without compromising model performance. This methodology is particularly advantageous for privacy-sensitive applications, such as healthcare, where local training on edge devices is paramount [8]. The true potential of federated learning lies in its capacity to train models on-device, thereby enhancing privacy while also presenting challenges related to synchronization and bandwidth [9].

Furthermore, complexity-driven compression concentrates on evaluating the inherent computational requirements of specific layers within CNN architectures [10]. Techniques like layer complexity analysis pinpoint the most resource-hungry layers for targeted compression, optimizing resource utilization while ensuring accuracy is upheld. This method has consistently proven effective in diminishing computational complexity in CNNs while retaining superior performance [11].

Despite the advancements achieved in model compression, considerable challenges endure. The resource limitations of edge devices, encompassing restricted computational power, memory, and energy, complicate the deployment of extensive models [12]. Preserving accuracy during compression is a paramount concern, as aggressive pruning and quantization can lead to performance setbacks [13]. Hybrid techniques that integrate pruning and quantization have demonstrated significant potential, successfully striking favorable balances between size reduction and accuracy. Energy efficiency remains a critical issue, with hardware-based solutions like HardCompress leveraging post-quantization trimming and dictionary-based compression to minimize energy consumption, particularly in devices equipped with systolic array accelerators [14].

In accordance with Figure 1, various methodologies for model training are contemplated: LIT, RKD, and GKD. The employment of LIT might lead to an escalation in model intricacy owing to its emphasis on acquiring intermediate representations, while RKD and GKD are focused on knowledge transmission, potentially enabling the utilization of less intricate models. Concerning computational resources, GKD may necessitate more resources due to its graph-oriented approach, whereas RKD is anticipated to be more computationally economical [15]. Concerning performance, LIT is deemed appropriate for scenarios requiring high performance where intricacy can be managed, whereas RKD is favored for efficiency and reduced model dimensions.

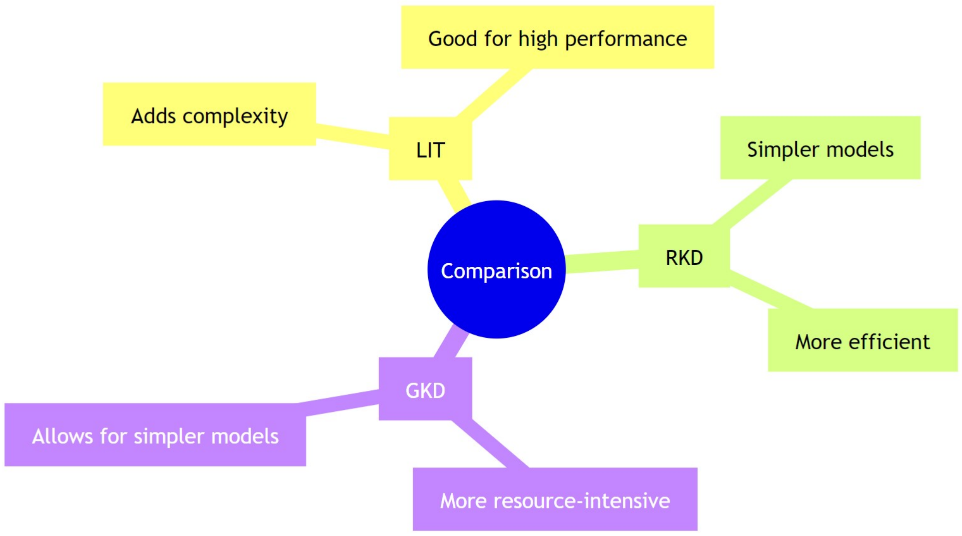

Figure 1. Comparison, branching out to LIT, RKD, and GKD

Microcontrollers' processing unit faces a trade-off between precision and speed. Precision of weights affects inference accuracy on specific hardware. RKD might be better for tinyML due to efficiency in data transfer. Choice depends on tinyML application's requirements and constraints[16].

Various methods are available to compress the architecture, with a specific focus on dealing with the feature maps. The diagram in Figure 2 demonstrates the different levels of energy or memory needed for various architectures when the number of features is decreased [17]. The energy-accuracy trade-off in deep learning models for edge devices is an essential factor for maximizing performance within resource constraints. From the provided graph, the baseline implementation (F) achieves the pinnacle of accuracy (95%) but does so with the highest energy consumption, showcasing exceptional accuracy even with moderate energy reductions. The F/$\sqrt{2}$ configuration provides a strategic compromise, delivering nearly identical accuracy

(∼94.8%) while consuming significantly less energy (∼0.5 normalized energy), making it ideal for scenarios where both accuracy and energy efficiency are paramount [18]. Conversely, F/2 experiences the sharpest decline in accuracy, reaching only 93.5% even at elevated energy levels (∼0.35 normalized energy), positioning it as the most energy-efficient yet least accurate option. This trade-off underscores the imperative of selecting the right models tailored to application-specific priorities, such as optimizing accuracy for critical tasks or emphasizing energy efficiency in resource-limited environments [19], [20]. Federated learning protocols are continuously advancing, enhancing communication efficiency and model performance while steadfastly maintaining decentralized training processes. The F/$\sqrt{2}$ model stands out as the premier choice for edge AI applications, delivering an exceptional balance between accuracy and energy efficiency. It is perfectly tailored for a diverse array of AI-driven tasks where achieving top-tier accuracy is paramount, while also prioritizing energy conservation.

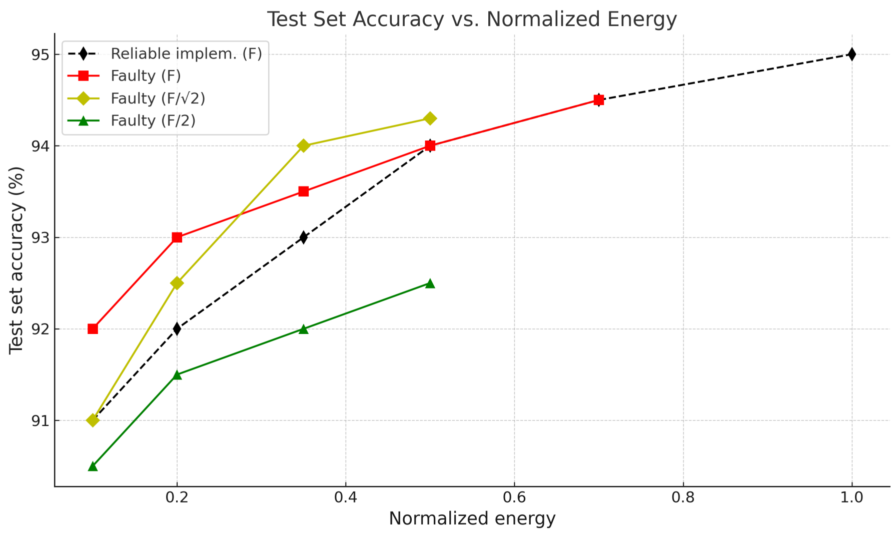

Figure 2. The performance of different implementations under varying energy conditions

3. METHOD

The model was trained considering different edge device devices for an object detection. The displayed flowchart effectively delineates a transfer learning process utilizing a pre-trained model (MobileNetV2) for advanced deep learning tasks, emphasizing critical stages in model adaptation and optimization [21]. Transfer learning is an assertive strategy where a model designed for a specific task is repurposed as the foundational element for a subsequent task, drastically shortening training time and enhancing performance by capitalizing on previously acquired features [22].

As shown in Figure 3, The process initiates with data preparation, which involves the collection, cleansing, and transformation of raw data into a training-ready format. Data augmentation is instrumental in preventing overfitting and significantly boosts the model's generalization capabilities on previously unseen data. Subsequently, a pre-trained model (MobileNetV2, in this instance) is activated. MobileNetV2 is strategically selected for its efficient architecture, engineered for mobile and edge computing tasks while delivering exceptional accuracy. After integrating the pre-trained model, the network is modified by incorporating custom layers specifically designed for the task at hand, usually involving fully connected layers that replace the original classifier [23].

In the concluding stages, the top layers are unfrozen, and the model is comprehensively retrained to fine-tune all parameters, empowering the model to absorb more task-specific features across every layer. The process culminates with a final retraining of the entire model, ensuring that both the pre-trained and new layers are optimally attuned for the specific task. This systematic approach to transfer learning robustly accelerates the model training process and results in enhanced accuracy, particularly when managing limited datasets. It harnesses the advantages of pre-trained models, minimizes computational expenses, and eliminates the necessity of training deep neural networks from the ground up. In this work, we strategically utilize transfer learning to adapt a pre-trained MobileNetV2 model for a highly specialized image classification task. The base model is confidently initialized with pre-trained weights and precisely configured with an input shape of 96×96×1 to flawlessly accommodate grayscale images.

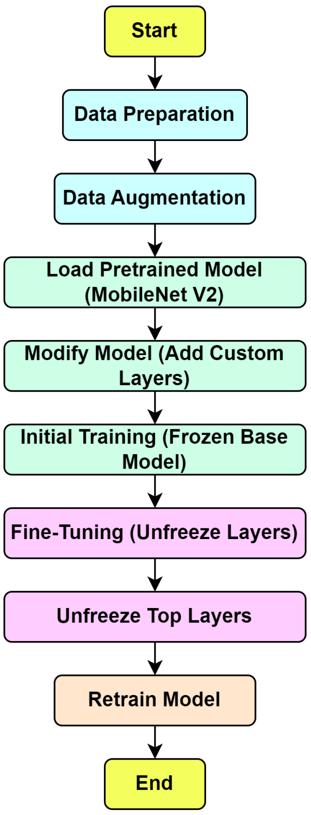
Start
Data Preparation
Data Augmentation
Load Pretrained Model (MobileNet V2)
Modify Model (Add Custom Layers)
Initial Training (Frozen Base Model)
Fine-Tuning (Unfreeze Layers)
Unfreeze Top Layers
Retrain Model
End

Figure 3. Working methodology with transfer learning algorithm

To optimize the model for our specific dataset, we decisively remove the last three layers of the base model and seamlessly append a custom classification head. This head is comprised of a Reshape layer, a Dense layer with 16 units and ReLU activation [24], a Dropout layer with a rate of 0.1, a Flatten layer, and a final Dense layer featuring a softmax activation that corresponds perfectly to the number of target classes. During the initial training phase, we strategically keep the base model's weights frozen to preserve the invaluable learned feature representations, while only the newly added top layers undergo training using the Adam optimizer [25] with a learning rate of 0.0005. Here (1), shows how the parameters θ are updated at each iteration using the Adam optimization algorithm with our specified learning rate $\alpha = 0.000045$.

$$l\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{1}$$

The denominator $\sqrt{\hat{v}_t} + \epsilon$ scales the learning rate for each parameter inversely proportional to the square root of the estimated variance of the gradients. Bias correction terms $\hat{m}_t$ and $\hat{v}_t$ are used to adjust for the initialization at zero.

3.1. Parameter selection

The choice of hyperparameters in our transfer learning framework was decisively crafted to maximize performance while efficiently managing computational resources. We utilized a pre-trained MobileNetV2 model with an alpha of 0.35, which offers a good balance between model complexity and performance, making it suitable for scenarios where computational resources are limited.

After the initial training, we confidently proceed with fine-tuning to further amplify the model's performance. We unfreeze the entire base model to enable all layers to be trainable and selectively freeze a portion of the layers based on a precise fine-tuning percentage of 65%. Specifically, we freeze the earlier layers of the model up to the calculated layer index, empowering the deeper layers to adapt to the new task while steadfastly maintaining foundational features learned from the pre-trained weights. The model is recompiled with a meticulously reduced learning rate of 0.000045 to facilitate fine-tuning without overshooting minima and is trained for an additional set of epochs.

The graph shown in Figure 4 illustrating training and validation accuracy per epoch showcases a swift convergence in the early stages of training, with training accuracy approaching 100% by epoch 10. To enhance performance, it is essential to adopt a learning rate schedule. Initiating with a higher learning rate (e.g., 0.01) during the first epochs guarantees rapid convergence, followed by a systematic decrease (e.g., halving every 5-10 epochs) to ensure stable learning. This strategy effectively balances quick initial learning with meticulous fine-tuning, ultimately boosting validation accuracy and model generalization while averting overfitting. Upon completion of the preprocessing stage, the model is subsequently trained for classification utilizing the Keras framework, specifically employing the Neural Network Transfer methodology.Various experiments have been conducted to explore the impact of adjustments in parameters such as training rate, dropout rate, learning rate, DSP block configuration, and the architecture of the neural network.

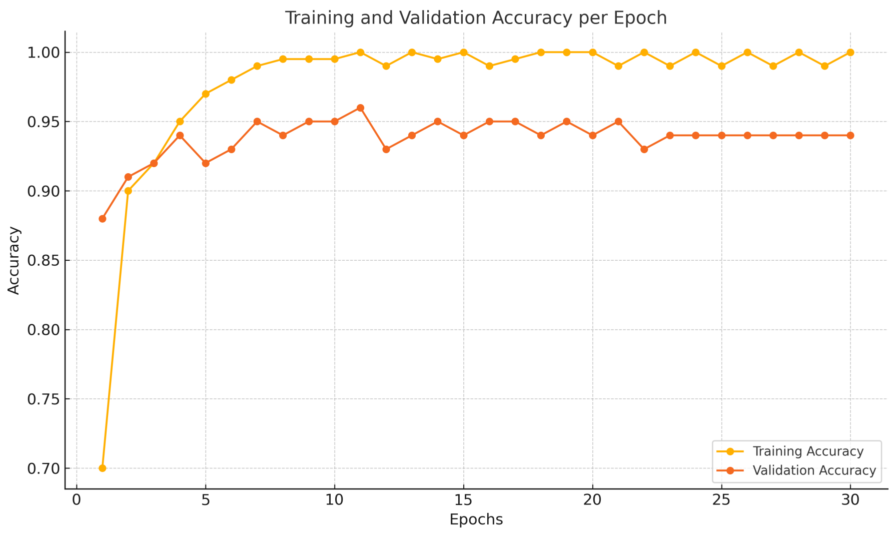

Figure 4. Training and validation accuracy per epoch for mask-unmasked datset using transfer learning

The Table 1 showcases an array of hyperparameter configurations utilized in the SSD MobileNet V2 during a transfer learning experiment, rigorously evaluating the effects of input layer dimensions, block type (RGB vs. Grayscale), neuron quantity, and dropout rate on latency. The findings decisively reveal that smaller input dimensions (96×96) and grayscale blocks consistently deliver superior performance with lower latency, exemplified by trial 3 (96×96), RGB, 16 neurons, 0.1 dropout) achieving an impressive response time of 5 ms. For real-time applications that demand minimal latency, the configuration in trial 3 stands out as the optimal choice, while trial 7 may excel in feature extraction for scenarios where accuracy takes precedence over speed.

Table 1. Hyperparameter settings and performance indicators [23]

Trial .o.	Input layer size	Block type	Neurons size	Layer dropout	Appx. latency (ms)
1	96×96	RGB	16	0.1	12
2	96×96	RGB	16	0.1	1636
3	96×96	RGB	16	0.1	5
4	96×96	RGB	16	0.35	8
5	96×96	Grayscale	16	0.1	34.52
6	160×160	RGB	64	0.1	20245.3
7	160×160	Grayscale	64	0.1	90.04

3.2. Results and discussions

The Table 2 confidently illustrates the performance of the SSD MobileNet V2 executed on a Raspberry Pi 4, utilizing both 8-bit quantized and 32-bit unoptimized models. The quantized int8 model boasts a remarkable decrease in memory usage, requiring just 334.6 K RAM and 585.1 K Flash, in stark contrast to the unoptimized int32 model, which demands 893.7 K RAM and 1.6 MB Flash. Additionally, the quantized model excels in inference speed, achieving a total latency of 6 ms, while the unoptimized model lags at 9 ms. Although there is a slight dip in accuracy (80.65% for int8 versus 81.72% for int32), the int8 quantized model delivers superior efficiency, rendering it ideal for deployment on resource-limited edge devices such as the Raspberry Pi 4. Therefore, the quantized int8 model stands out as the optimal choice for applications that prioritize speed and resource efficiency, even with a minor compromise in accuracy.

Table 2. Quantized int8 model

Model	Performance metrics	Image	Transfer learning	Total time
Quantized int8	Latency	1 ms	5 ms	6 ms
	RAM	4.0 K	334.6 K	334.6 K
	Flash	-	585.1 K	585.1 K
	Accuracy			80.65%
Unoptimzed int32	Latency	1 ms	8 ms	9 ms
	RAM	4.0 K	893.7 K	893.7 K
	Flash	-	1.6 MB	-
	Accuracy			81.72%

The Table 3 offers an insightful comparative evaluation of inference latency across a spectrum of state-of-the-art edge computing devices utilizing 8-bit quantized SSD MobileNet V2. The results clearly demonstrate that devices equipped with superior hardware, such as the Nvidia Jetson Orin Nano and Raspberry Pi 5, showcase the lowest total latency of 85 ms and 5 ms respectively, positioning them as the prime candidates for low-latency applications. In stark contrast, microcontroller-based platforms like the Arduino Nano 33 BLE Sense and Espressif ESP-EYE face considerably higher total latencies of 1,221 ms and 1,600 ms, respectively, which significantly hampers their effectiveness for real-time inference tasks. The Nvidia Jetson Nano exhibits balanced performance with a latency of 20 ms, making it well-suited for moderately demanding edge AI tasks. Consequently, for applications that demand minimal inference latency, the Raspberry Pi 5 and Nvidia Jetson Orin Nano emerge as the most powerful choices, while microcontroller-based devices are more appropriate for less urgent applications.

Table 3. Comparison between edge computing devices with latency by quantized (int8) optimization

Edge computing device	Image (in ms)	Transfer learning (in ms)	Total (in ms)
Arduino Nano BLE Sense	11	1,210	1,221
Espressif ESP-EYE	15	1,585	1,600
Arduino Portenta H7	1	104	105
Raspberry Pi 4	1	5	6
Raspberry Pi 5	1	4	5
Nvidia Jetson Nano	1	19	20
Nvidia Jetson Orin Nano	1	84	85

Moreover, the data indicates a trade-off between hardware capabilities and the effectiveness of optimization methodologies. Despite the Raspberry Pi 5 exhibiting commendable performance with quantized (int8) optimization, the Espressif ESP-EYE (ESP32 240 MHz) encounters notable latency issues under the same optimization, suggesting that the selection of hardware and optimization technique should align with the specific demands of the application. This becomes particularly important for devices with limited storage capacity or when the model needs to be stored in memory-constrained environments.While quantizing the model results in a reduction of accuracy from 88.17% to 83.87%, this decrease is often considered acceptable due to significant improvements in latency and reductions in RAM and flash storage usage. The decision to adopt this trade-off depends on the specific requirements and constraints of the application.

The confusion matrix from Table 4, effectively conveys the distinguished performance of our binary classification model on a validation dataset contrasting masked and unmasked samples, achieving a significant overall accuracy of 84.5% with a slight loss of 0.43. The model excels at correctly identifying 90.5% of masked samples and 75.9% of unmasked samples, although it does exhibit a higher misclassification rate for unmasked images (24.1%). The F1 scores of 0.87 for masked and 0.80 for unmasked reveal that the model is superior in identifying masked individuals. The elevated F1 score for the masked category signifies enhanced precision and recall, establishing the model as a highly reliable tool for mask detection, though there are clear opportunities to refine performance in the unmasked category to achieve even more balanced outcomes.

Table 4. The confusion matrix (on validation set)

	Mask	Unmask
Mask	90.5%	9.5%
Unmask	24.1%	75.9%
F1 score	0.87	0.80

4. CONCLUSION

The findings showcased in this study on quantized transfer learning for object detection in edge-based video surveillance systems decisively highlight the superiority of utilizing SSD MobileNet V2 with 8-bit quantization. The latency analysis across a range of edge devices conclusively identifies the Raspberry Pi 5 and Nvidia Jetson Orin Nano as the most proficient options, achieving total latencies of 5 ms and 85 ms respectively, affirming their excellence for real-time applications. Furthermore, the int8 quantized model exhibited remarkable resource savings, consuming considerably less RAM and flash memory while delivering competitive accuracy (80.65%) in comparison to the unoptimized int32 model. The quantization process effectively minimized the model size and memory footprint, leading to reduced latency and expedited inference times, making it exceptionally suited for resource-constrained settings. In addition, the validation results on the masked-unmasked dataset demonstrate outstanding performance with an F1 score of 0.92 for detecting masked individuals. Future research should systematically explore hybrid quantization methodologies and architectural enhancements to improve accuracy while maintaining efficiency, in conjunction with comprehensive testing across varied datasets and edge devices to assess scalability. The slight accuracy reduction post-quantization, while advantageous for resource efficiency, presents a challenge for precision-critical applications, which forthcoming research must tackle head-on.

ACKNOWLEDGMENT

We extend our sincere gratitude to the Research Center at Sinhgad College of Engineering, which is affiliated with Savitribai Phule Pune University, for their outstanding provision of hardware and software support, which was essential to the advancement of our research. Furthermore, we recognize MIT Art Design and Technology University for their invaluable contribution to data collection efforts. The collaboration and resources rendered by both institutions were crucial to the accomplishment of our study, and we wish to express our profound appreciation.

REFERENCES

[1] M. Sigala, A. Beer, L. Hodgson, and A. O'Connor, *Big Data for Measuring the Impact of Tourism Economic Development Programmes: A Process and Quality Criteria Framework for Using Big Data*. 2019.

[2] G. Nguyen *et al.*, "Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey," *Artificial Intelligence Review*, vol. 52, no. 1, pp. 77–124, 2019, doi: 10.1007/s10462-018-09679-z

[3] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *Journal of big data*, vol. 6, no. 1, 2019, doi: 10.1186/s40537-019-0197-0.

[4] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep Learning Approach for Intelligent Intrusion Detection System," *IEEE Access*, vol. 7, pp. 41525–41550, 2019, doi: 10.1109/ACCESS.2019.2895334.

[5] K. Sivaraman, R. M. V. Krishnan, B. Sundarraj, and S. S. Gowthem, "Network failure detection and diagnosis by analyzing syslog and SNS data: Applying big data analysis to network operations," *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 9 Special Issue 3, pp. 883–887, 2019, doi: 10.35940/ijitee.I3187.0789S319.

[6] A. D. Dwivedi, G. Srivastava, S. Dhar, and R. Singh, "A decentralized privacy-preserving healthcare blockchain for IoT," *Sensors (Switzerland)*, vol. 19, no. 2, pp. 1–17, 2019, doi: 10.3390/s19020326.

[7] F. Al-Turjman, H. Zahmatkesh, and L. Mostarda, "Quantifying uncertainty in internet of medical things and big-data services using intelligence and deep learning," *IEEE Access*, vol. 7, pp. 115749–115759, 2019, doi: 10.1109/ACCESS.2019.2931637.

[8] S. Kumar and M. Singh, "Big data analytics for healthcare industry: Impact, applications, and tools," *Big data mining and analytics*, vol. 2, no. 1, pp. 48–57, 2019, doi: 10.26599/BDMA.2018.9020031.

[9] L. M. Ang, K. P. Seng, G. K. Ijemaru, and A. M. Zungeru, "Deployment of IoV for Smart Cities: Applications, Architecture, and Challenges," *IEEE Access*, vol. 7, pp. 6473–6492, 2019, doi: 10.1109/ACCESS.2018.2887076.
[10] B. P. L. Lau *et al.*, "A survey of data fusion in smart city applications," *Information Fusion*, vol. 52, no. January, pp. 357–374, 2019, doi: 10.1016/j.inffus.2019.05.004.
[11] Y. Wu *et al.*, "Large scale incremental learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, vol. 2019-June, 2019, pp. 374–382, doi: 10.1109/CVPR.2019.00046.
[12] A. Mosavi, S. Shamshirband, E. Salwana, K. wing Chau, and J. H. M. Tah, "Prediction of multi-inputs bubble column reactor using a novel hybrid model of computational fluid dynamics and machine learning," *Engineering Applications of Computational Fluid Mechanics*, vol. 13, no. 1, pp. 482–492, 2019, doi: 10.1080/19942060.2019.1613448.
[13] V. Palanisamy and R. Thirunavukarasu, "Implications of big data analytics in developing healthcare frameworks – A review," *Journal of King Saud University-Computer and Information Sciences*, vol. 31, no. 4, pp. 415–425, 2019, doi: 10.1016/j.jksuci.2017.12.007.
[14] J. Sadowski, "When data is capital: Datafication, accumulation, and extraction," *Big data & society*, vol. 6, no. 1, pp. 1–12, 2019, doi: 10.1177/2053951718820549.
[15] J. R. Saura, B. R. Herraez, and A. Reyes-Menendez, "Comparing a traditional approach for financial brand communication analysis with a big data analytics technique," *IEEE Access*, vol. 7, pp. 37100–37108, 2019, doi: 10.1109/ACCESS.2019.2905301.
[16] D. Nallaperuma *et al.*, "Online Incremental Machine Learning Platform for Big Data-Driven Smart Traffic Management," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 12, pp. 4679–4690, 2019, doi: 10.1109/TITS.2019.2924883.
[17] S. Schulz, M. Becker, M. R. Groseclose, S. Schadt, and C. Hopf, "Advanced MALDI mass spectrometry imaging in pharmaceutical research and drug development," *Current opinion in biotechnology*, vol. 55, pp. 51–59, 2019, doi: 10.1016/j.copbio.2018.08.003.
[18] C. Shang and F. You, "Data Analytics and Machine Learning for Smart Process Manufacturing: Recent Advances and Perspectives in the Big Data Era," *Engineering*, vol. 5, no. 6, pp. 1010–1016, 2019, doi: 10.1016/j.eng.2019.01.019.
[19] H. N. Lokhande, S. R. Ganorkar, "Optimizing Real-Time Object Detection on Edge Devices: A Transfer Learning Approach," *International Journal of Intelligent Systems and Applications in Engineering (IJISAE)*, vol. 12, no. 21s, pp. 3896-3903, 2024.
[20] M. Huang, W. Liu, T. Wang, H. Song, X. Li, and A. Liu, "A queuing delay utilization scheme for on-path service aggregation in services-oriented computing networks," *IEEE Access*, vol. 7, pp. 23816–23833, 2019, doi: 10.1109/ACCESS.2019.2899402.
[21] G. Xu, Y. Shi, X. Sun, and W. Shen, "Internet of things in marine environment monitoring: A review," *Sensors (Switzerland)*, vol. 19, no. 7, pp. 1–21, 2019, doi: 10.3390/s19071711.
[22] M. Aqib, R. Mehmood, A. Alzahrani, I. Katib, A. Albeshri, and S. M. Altowaijri, *Smarter traffic prediction using big data, in-memory computing, deep learning and gpus*, vol. 19, no. 9. 2019.
[23] S. Leonelli and N. Tempini, *Data Journeys in the Sciences*. 2020.
[24] N. Stylos and J. Zwiegelaar, *Big Data as a Game Changer: How Does It Shape Business Intelligence Within a Tourism and Hospitality Industry Context?*, 2019.
[25] Q. Song, H. Ge, J. Caverlee, and X. Hu, "Tensor completion algorithms in big data analytics," *arXiv*, vol. 13, no. 1, 2017, doi: 10.48550/arXiv.1711.10105.

BIOGRAPHIES OF AUTHORS

Harshad Lokhande is a highly accomplished professional with a strong academic background and a passion for research and teaching. He holds a Master's degree in Electronics and Telecommunication Engineering. His industrious journey spans an impressive 15 years, during which he has excelled in both the educational and industrial domains. His research interests reflect his commitment to innovation and cutting- edge technology. He is particularly fascinated by embedded systems, internet of things (IoT), and edge computing, exploring their potential to revolutionize various industries. Currently, serves as an Assistant Professor at MIT Arts Design and Technology University in Pune, India. He can be contacted at email: lokhande.harshad@gmail.com.

Sanjay R. Ganorkar serves as the Head of the Department (Information Technology) at Sinhgad College of Engineering and boasts an illustrious career spanning over 25 years in Image Processing. A prolific scholar, he has contributed more than 65 papers to National and International Journals, Conferences, and Symposia, marking his presence as a significant voice in the field. Dr. Ganorkar is also an esteemed reviewer for various Scopus and Web of Science indexed journals, bringing his critical expertise to the forefront of academic discourse. His role as a Session Chair for International Conferences and his advisory and convening responsibilities in Ph.D. Research programmes further underscore his commitment to advancing research and mentoring the next generation of scholars in engineering and technology. He can be contacted at email: ganorkarsanjay@gmail.com.