SSRG International Journal of Electrical and Electronics Engineering
ISSN: 2348-8379/ https://doi.org/10.14445/23488379/IJEEE-V11I5P108

Volume 11 Issue 5, 77-84, May 2024

Original Article

3D Bounding Box Estimation Using Deep Learning and Geometry Based on Yolov7 Output on Single-Board Computer

Tuan Muhammad Naeem Bin Tuan Rashid[1], Lokman Mohd Fadzil[1*], Mohd Adib Haji Omar[2]

[1]*National Advanced IPv6 Centre (NAv6), University Sains Malaysia (USM), Penang, Malaysia.*
[2]*School of Computer Science, University Sains Malaysia (USM), Penang, Malaysia.*

[*]*Corresponding Author : lokman.mohd.fadzil@usm.my*

Received: 06 March 2024 Revised: 06 April 2024 Accepted: 04 May 2024 Published: 29 May 2024

Abstract *- This study investigates the enhancement of 3D bounding box estimation techniques for object localization on Single-Board Computers (SBCs), focusing on the Jetson Nano platform. The adaptation and optimization of deep learning models, specifically transitioning from VGG networks and YOLOv3 to more efficient alternatives like MobileNetV3 and YOLOv7, within the constraints of SBCs. The implementation leverages the advanced capabilities of MobileNetV3 for 3D bounding box generation, coupled with the superior detection accuracy and speed of YOLOv7 for object detection. The research employs an innovative loss function to improve 3D orientation predictions and utilizes geometric constraints from 2D bounding boxes for precise object localization. A comparative analysis of MobileNet V3, VGG-19, and MobileNet V2 models on the Jetson Nano inference speed and consistency reveals that MobileNet V3, optimized using TensorRT, significantly outperforms others, a preferable candidate for solutions in real-time environments. The study concludes that the strategic optimization of deep learning models on SBCs, like the Jetson Nano, markedly enhances the performance and applicability of 3D bounding box estimation in edge computing environments, offering valuable insights for deploying advanced object detection technologies in resource-constrained scenarios.*

Keywords *- 3D bounding box, Computer vision, Embedded system, IoT applications, Performance benchmarking.*

1. Introduction

Of late, deep learning and computer vision domains have seen substantial advancements, especially in object detection and localization. The use of 3D bounding boxes for object estimation has emerged as crucial for applications in autonomous driving, robotic vision, and augmented reality. However, implementing these advanced techniques on resource-limited platforms like Single-Board Computers (SBCs), which serve to offload some processing tasks from central servers for edge processing, presents significant hurdles.

While SBCs are cost-effective and compact, their computational capabilities often lag behind those of more powerful systems, necessitating the need for deep learning models to be tailored to their constraints. This study aims to advance the work of Mousavian et al. (2017) by exploring the practicality and improving single-board computer performance in terms of sophisticated object localization methods. The original research predominantly used VGG networks for 3D bounding box generation and You Only Look Once version 3 (YOLOv3) for object detection [1]. VGG is noted for its deep convolutional layers [2], and YOLOv3 for its rapid detection speed and accuracy [3], setting a benchmark in the field. Nevertheless, the advent of new models and architectures opens up possibilities for refining this framework to achieve better efficiency and effectiveness.

Accordingly, this research introduces two major updates. Firstly, the YOLOv7 adoption. Being the advanced component of the YOLO algorithm family, acclaimed for its superior detection accuracy and speed. YOLOv7's architecture includes numerous enhancements over its predecessors [4], making it the prime selection for real-time computing services on bare-bone devices.

Secondly, replacing the VGG backbone with a more efficient model like MobileNetV3, which is designed specifically for mobile and edge computing environments [5]. MobileNetV3 optimally balances latency and accuracy [5], which is vital for applications on single-board computers. Additionally, further model performance enhancement can be achieved via TensorRT optimization. As for the SBC, Jetson Nano [6] is chosen, given its standing as a representative

example of current SBC technology. This research not only aims to validate the feasibility of leveraging deep learning and geometric principles for 3D bounding box estimation on single-board computers but also seeks to elevate its efficiency, making it more applicable for real-life scenarios that require portable and low-power solutions. The findings of this study will shed light on adapting cutting-edge deep learning models for use within ecosystems with constrained resources, extending the range and applicability of advanced object detection technologies.

The paper will commence with a thorough review of the relevant literature, followed by a presentation of the implementation, which will include the modifications made for SBC compatibility. Subsequent sections will outline the results, and a detailed discussion of the outcomes, culminating in repercussions for emerging research and real-world applications.

2. Literature Review

This literature review critically examines the evolution and existing status of Deep Learning (DL) algorithms in object detection and existing 3D bounding box generation research. The review delves into the progression of the object detection model, which is YOLO, from the earliest version to the one of the latest, YOLOv7 and onto related research on 3D bounding box generation.

Before delving into the YOLO families, let us take a look at the initial breakthrough of object detection, which was the Region-based Convolutional Neural Networks (R-CNN) that offered a novel approach to object detection. R-CNN combined high-capacity CNNs with region proposal methods to localize and classify in-image objects [7]. However, the R-CNN framework was computationally expensive due to the independent processing of multiple region proposals per image.

As a result, improvements seen in Fast R-CNN [8] paved the way for shared convolutional feature maps for all region proposals, significantly improving efficiency. The subsequent iteration, Faster R-CNN, integrated a Region Proposal Network (RPN), allowing the network to spawn the region proposals, thus speeding up the process [9].

2.1. YOLO Algorithms and Its Evolution

In contrast to the region proposal-based approaches, Redmon et al. pitched the YOLOv1 algorithm in 2016 [10], revolutionising object detection with its unique approach. YOLO realigned object detection primarily as a sole regression problem, straightforwardly calculating class probabilities and bounding boxes from whole images in a single pass [10]. This unified method enabled the YOLO algorithm to achieve remarkable speeds, significantly outpacing its contemporaries like R-CNN. YOLO's initial version, while fast, lagged in terms of accuracy, particularly with small objects and objects in groups [10]. This led to the development of YOLOv2 (or YOLO9000) by Redmon and Farhadi in 2017, which improved the input image resolution and incorporated anchor boxes, enabling the model to detect miniature-sized objects [11].

Redmon and Farhadi achieved another breakthrough in YOLOv3 iteration by introducing multi-scale predictions and a deeper architecture, further improving the detection accuracy across varied object sizes [3]. YOLOv3's architecture, with 106 convolutional layers, was a significant leap, enabling the detection of objects at three different scales, thereby capturing a wider range of object sizes more effectively [3].

The evolution of the YOLO series did not stop there. Subsequent versions, such as YOLOv4 [12], YOLOv5 [13], and YOLOv6 [14] and culminating in one of the latest such as YOLOv7, have continuously refined the balance between speed and accuracy [4]. YOLOv7, in particular, incorporates advancements in network design, loss functions, and training techniques, setting new benchmarks in object detection tasks. These improvements make YOLOv7 a prime candidate for real-time object detection applications, even in environments with computational limitations [4].

2.2. 3D Bounding Box Estimation

The evolution from two-dimensional to three-dimensional bounding box detection signifies a pivotal development in computer vision, profoundly impacting fields such as autonomous driving [17], robotics [18], and augmented reality [19]. This evolution entails the complex task of deducing an object's precise location and orientation in three-dimensional space from mere two-dimensional images [20].

This advancement not only enhances the accuracy and reliability of object detection systems but also broadens their applicability across various technological domains, paving the way for more sophisticated and immersive applications [20]. Highlighted below are some of the seminal research efforts in this area, showcasing the diverse methodologies and innovations that have propelled this field forward.

2.2.1. Deep Learning Assisted Approach

An exemplary technique within this domain cited recent work involving deep learning and geometric methods in 3D bounding box estimation by employing a deep learning framework to deduce stable attributes of 3D objects [1]. These attributes are then merged with geometric constraints derived from the object's two-dimensional bounding box to construct a comprehensive three-dimensional bounding box [1]. This method capitalizes on established 2D object detection methodologies to infer the 3D bounding box dimensions, ensuring the three-dimensional bounding box's perspective projection accurately fits within the 2D detection frame [1]. It

effectively minimizes the reprojection error by calculating the optimal translation based on the constraints of the initial 2D detection box, thereby enhancing computational efficiency and facilitating real-time application scenarios [1].

Another groundbreaking technique is DETR3D [21], which utilizes deep learning to detect 3D objects from images captured from multiple viewpoints, creating a LiDAR-esque effect using purely camera-based data. This system processes RGB images from several cameras to deduce the parameters of 3D bounding boxes. DETR3D is noteworthy for its set prediction module that seamlessly transitions between 2D and 3D processing, bridging the gap between 2D feature extraction and 3D bounding box prediction through a geometry-conscious framework.

The model employs a common ResNet architecture and an optional Feature Pyramid Network (FPN) for feature extraction, subsequently projecting 3D reference points into image planes to refine object queries. This approach mitigates common drawbacks of traditional methods, such as the need for dense 3D geometry reconstructions and extensive post-processing, like Non-Maximum Suppression (NMS), thereby streamlining its suitability for real-time applications.

2.2.2. Deep Learning and LiDAR Assisted Approach

A novel deep learning architecture, termed RoIFusion, is designed to effectively integrate features from multiple modalities for 3D object detection, capitalizing on the strengths of both LIDAR and camera technologies. Unlike traditional methods that densely merge point-wise features from point clouds with corresponding pixel features from images, RoIFusion innovatively combines a limited number of 3D Regions of Interest (RoIs) from point clouds with matching 2D RoIs from images. This approach not only minimizes computational demands but also addresses the issue of misalignment between different sensor perspectives during feature combinations. Comprehensive testing on the KITTI 3D object detection benchmark has been conducted to validate the efficacy of this fusion technique, with results indicating that RoIFusion sets a new benchmark in performance [22].

While the research highlighted above demonstrates promising results, a common limitation is their reliance on supplementary hardware, such as LiDAR, or the need for high-end GPU resources, which can be prohibitive for widespread adoption. In response to these challenges, this paper seeks to advance the deep learning-assisted methodology presented in a study using deep learning and geometry [1], which will be discussed more in the implementation section. The focus is on refining crucial elements, including the object detection process and the foundational architecture for generating 3D bounding boxes. By optimizing these aspects before engaging in complex mathematical calculations, the processing efficiency is significantly enhanced, and the computational demands of the model are diminished. This strategic improvement will facilitate the deployment of the enhanced model on SBD, which is Jetson Nano for edge computing, making it more accessible and practical for real-time applications in diverse environments. This initiative underscores a commitment to pushing the boundaries of computer vision technology, making it more adaptable and efficient for a range of applications.

3. Materials and Methods

The core concept of the paper presented by Mousavian et al. (2017) revolves around identifying objects within a 3D environment and determining their poses-both orientation and position using a singular image through a synthesis of various techniques. A Convolutional Neural Network (CNN) is utilized to predict specific, stable attributes of 3D objects, such as their dimensions and orientation.

To enhance the 3D orientation prediction accuracy, the approach introduces an innovative loss function that melds discrete and continuous aspects, offering a notable improvement over the conventional L2 loss typically employed in deep learning regression tasks. Additionally, the network is designed to approximate the dimensions of the 3D object namely height, width, and length which tend to be more consistent and thus more reliably predicted across different object categories.

The approach also capitalizes on the geometric constraints imposed by the 2D bounding box visible within the image, which serves to limit the potential 3D positions of the object. By integrating the CNN's orientation and dimension predictions with the spatial constraints provided by the 2D bounding box, the method is capable of assembling a comprehensive 3D bounding box that encapsulates the entire pose of the object within the 3D space [1]. This research will focus on improving the pretrained model for regressing the 3D parameter and the object detection model used in the paper.

3.1. Object Detection Replacement

In the paper [1], Mousavian et al. utilized YOLOv3 as the foundational object detection model to pinpoint objects and extract 2D bounding boxes for subsequent 3D regression analysis. Redmon and Farhadi introduced YOLOv3 in 2018; YOLOv3 marked a significant advancement from its predecessor, YOLOv2, by incorporating multi-scale prediction capabilities through feature pyramids. This enhancement improved the model's ability to detect objects at varying scales and aspect ratios accurately. YOLOv3 calculates 4 coordinates using logistic regression for each bounding box objectness score, designating an anchor box for each object detected. If an anchor box is not assigned, it only influences the classification loss, leaving the localization and confidence loss unaffected.

Additionally, YOLOv3 uses binary cross-entropy for classifying objects, enabling it to assign multiple labels to a single bounding box-a useful trait for complex categorizations such as an object being both a "Person" and a "Man." YOLOv3 maintains its predecessors' real-time processing proficiencies while delivering better mean Average Precision (mAP) scores and more accurate localization. Despite being celebrated for its speed in object detection, YOLOv3 has been overtaken by a variety of newer algorithms.

To push the performance of object detection, the existing YOLOv3 from the paper was replaced with one of the latest entries from the YOLO family, YOLOv7. YOLOv7, created by Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao in 2022, introduced significant improvements to the YOLO object detection series. The model features a new layer design called E-ELAN, which helps it learn more efficiently by better managing how it processes information, even when dealing with complex models. This design also mixes and matches features from different parts of the model to improve learning without disrupting the flow of important information. Another major advancement in YOLOv7 is its unique way of adjusting the model's size to maintain its effectiveness, unlike previous methods that could reduce the model's performance.

The model also uses a revised approach to convolution, called RepConvN, which removes certain connections that were found to hinder performance in previous versions like YOLOv6. YOLOv7 makes a clear distinction in how it assigns labels for training, which helps in achieving more accurate results. It also incorporates some smart tweaks during the final stages of model inference, such as integrating batch normalization directly into the convolutional layers and using an exponential moving average, drawing inspiration from the YOLO strategy to enhance its detection capabilities. These updates collectively make YOLOv7 a more efficient and accurate object detection model compared to YOLOv3, as demonstrated in Table 1.

Table 1. YOLOV3 and YOLOV7 performance

Reference	Models	Test Set	Input Size	Map (%)	FPS
[3]	YOLOv3	COCO	320x320	51.5	38
[3]	YOLOv3	COCO	416x416	55.3	31
[3]	YOLOv3	COCO	608x608	57.9	23
[4]	YOLOv7	COCO	640x640	69.7	161

3.2. 3D Bounding Box Estimation Backbone Replacement

The understanding related to the paper [1] for 3D bounding box estimation led to a strategic decision to use MobileNetV3 as the backbone architecture, replacing the traditionally used VGG network. This choice is grounded in the architectural advantages and suitability of MobileNetV3 for environments with computational constraints, such as Single-Board Computers (SBCs) (Figure 1).

Howard et al. developed MobileNetV3 as part of the MobileNet family, specifically designed for low-power edge and mobile devices due to efficiency and compactness [23], uniquely balanced between computational efficiency and model performance. Key MobileNetV3 features include lightweight depthwise separable convolutions, network pruning and optimization, the use of advanced non-linearities like the h-swish activation function, and hardware-aware optimization using AutoML and Neural Architecture Search (NAS).

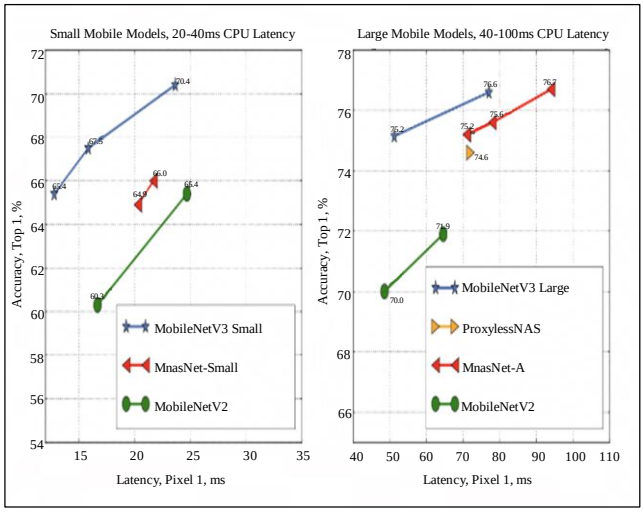

Fig. 1 Performance comparison on small and large mobile models [5]

Contrastingly, the VGG network, known for its deep architecture [15], faces challenges in deployment on SBCs (Figure 2). VGG's high computational complexity, substantial number of parameters, and large model size make it less practical for environments where processing power and memory are limited [23]. Its design, focusing on performance without particular consideration for efficiency, lacks architectural optimizations like those in MobileNetV3, leading to lower energy efficiency. This is a significant drawback for SBCs, where energy efficiency is often a critical factor.

By integrating MobileNetV3 as the backbone for 3D bounding box estimation, this research aims to leverage its lightweight and computationally efficient nature. The streamlined architecture of MobileNetV3 significantly reduces computational demands while maintaining high accuracy, aligning with the goal of developing efficient deep-learning solutions for resource-constrained environments. This makes MobileNetV3 a more appropriate choice compared to the heavier and less efficient VGG network for application in SBCs.

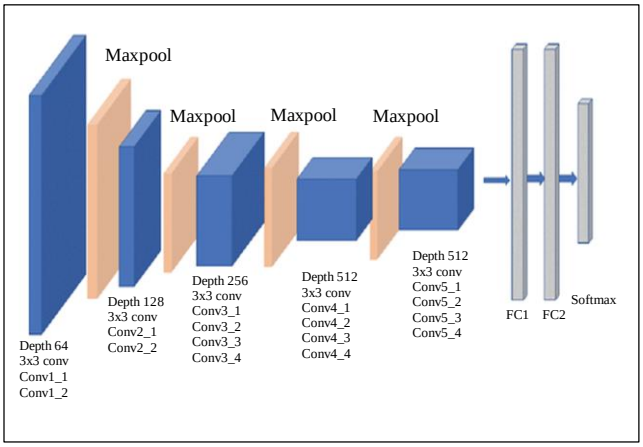

Fig. 2 VGG-19 architecture illustrations [17]

3.3. Model Optimization Using TensorRT

NVIDIA developed a superior-performance DL runtime library and inference optimizer called TensorRT [16]. It offers several key benefits for deep learning inference, particularly in production environments:

3.3.1. Performance Optimization

Performance Optimization: TensorRT can significantly increase the inference speed of deep learning models by optimizing the network structure and layer operations. This includes techniques like layer fusion, precision calibration (e.g., using FP16 or INT8 instead of FP32), and kernel auto-tuning to make the best use of the underlying hardware [16].

3.3.2. Reduced Resource Footprint

By optimizing models, TensorRT reduces both the computational footprint and memory usage, which is crucial for deploying models on edge devices with limited resources, such as embedded systems and IoT devices [16].

3.3.3. Cross-Platform Consistency

TensorRT ensures that models perform consistently across different platforms, from data centers with powerful GPUs to edge devices, making it easier to deploy and maintain AI applications across diverse environments [16].

3.3.4. Support for Major Frameworks

TensorRT provides support for importing models from major deep learning frameworks like TensorFlow, PyTorch, and Open Neural Network Exchange (ONNX), making it versatile and accessible for a wide range of applications and development workflows [16].

4. Results and Discussion

For this section, object detection will not be discussed, as it has already been demonstrated in Table 1. The result and analysis of 3D bounding box generation also consisted of two sections. The first section would be an analysis where three models in Pytorch format were run. The second section would be the best model from the previous analysis against the comparable TensorRT version.

4.1. Pytorch Models

The following analysis presents a comparative evaluation of the inference time metrics for the MobileNet V3, VGG-19, and MobileNet V2 architectures implemented on a Jetson Nano device. This examination includes MobileNet V2 to ascertain performance variations relative to its successor. Key findings are delineated below:

4.1.1. MobileNet V3 Performance on Jetson Nano (Figure 3)

- The mean inference duration across various batch sizes is noted to be approximately 0.095 seconds, indicating efficient processing capabilities.
- Inference time variability remains minimal for batch sizes ranging from 1 to 6, evidenced by a standard deviation of 0.086 seconds, underscoring consistent performance.
- Anomalously, batch size 1 exhibits a markedly elevated mean inference time of 0.265 seconds, diverging significantly from the trend observed in subsequent batch sizes.
- Batch sizes 2 through 6 demonstrate homogenous performance metrics, with inference times confined within the 0.047 to 0.049 seconds range, indicative of model stability.
- A discernible escalation in inference time to 0.162 seconds is observed at batch size 7, hinting at a decrement in performance with increasing batch size.
- The model's standard deviation is notably high at 4.067 seconds, largely impacted by the outlier at batch size 7, suggesting potential model or hardware limitations at increased batch sizes.

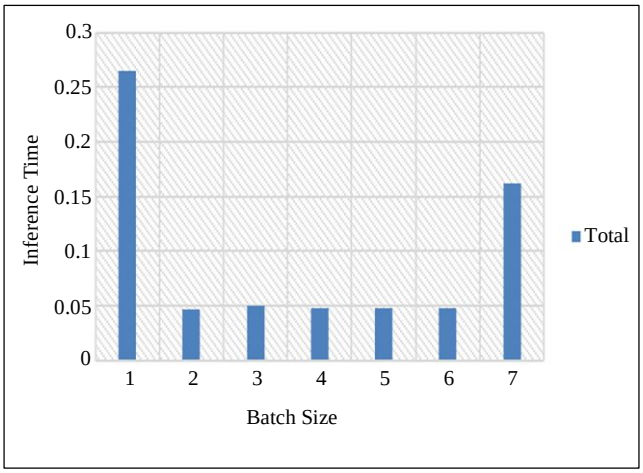

Fig. 3 Mobilenet V3 performance, inference time against batch sizes

4.1.2. VGG-19 Performance on Jetson Nano (Figure 4)

- Disregarding the outlier at batch size 7, a gradual increment in inference times is observed from batch size 1 to 6, ranging between 0.053 to 0.566 seconds.

- A substantial standard deviation of 1.620 seconds is observed, primarily attributed to the pronounced variability in inference times at batch sizes 1 and 4, which are notably high at 3.278 and 2.385 seconds, respectively.
- Noteworthy is the performance at batch sizes 2 and 3, where inference times are markedly lower (0.145 and 0.052 seconds, respectively). However, the model's efficacy diminishes beyond batch size 4, indicating scalability constraints on the Jetson Nano.
- The average inference time for VGG-19 is significantly higher at 1.465 seconds, reflecting its greater computational complexity in comparison to the MobileNet series.
- A substantial standard deviation of 1.620 seconds is observed, primarily attributed to the pronounced variability in inference times at batch sizes 1 and 4, which are notably high at 3.278 and 2.385 seconds, respectively.
- Noteworthy is the performance at batch sizes 2 and 3, where inference times are markedly lower (0.145 and 0.052 seconds, respectively). However, the model's efficacy diminishes beyond batch size 4, indicating scalability constraints on the Jetson Nano.

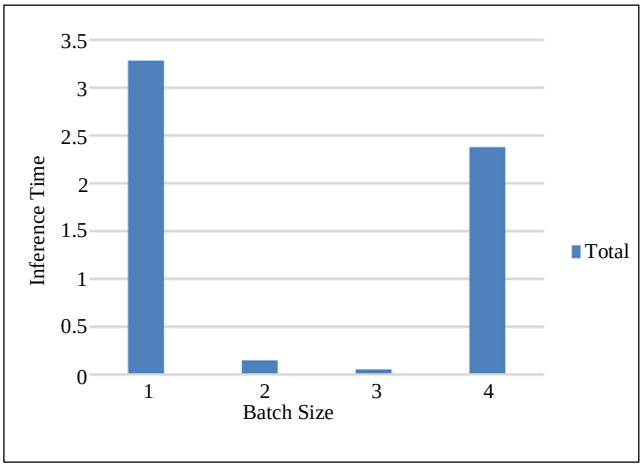

Fig. 4 VGG-19 performance, inference time against batch sizes

4.1.3. MobileNet V2 Performance on Jetson Nano (Figure 5)

- MobileNet V2 exhibits the highest average inference time among the evaluated models at 1.720 seconds, with the inference time at batch size 7 (10.934 seconds) significantly influencing this average.
- The model's standard deviation is notably high at 4.067 seconds, largely impacted by the outlier at batch size 7, suggesting potential model or hardware limitations at increased batch sizes.

From the results, MobileNet V3 emerges as the most consistent and efficient model, displaying stable inference times across varied batch sizes, making it well-suited for real-time or near-real-time inference applications on the Jetson Nano. VGG-19's performance is marred by significant variability, especially at higher batch sizes, indicating its suboptimal suitability for low-power devices like the Jetson Nano. While MobileNet V2 shows efficiency at lower batch sizes, its performance at higher batch sizes raises concerns regarding scalability and potential performance degradation.

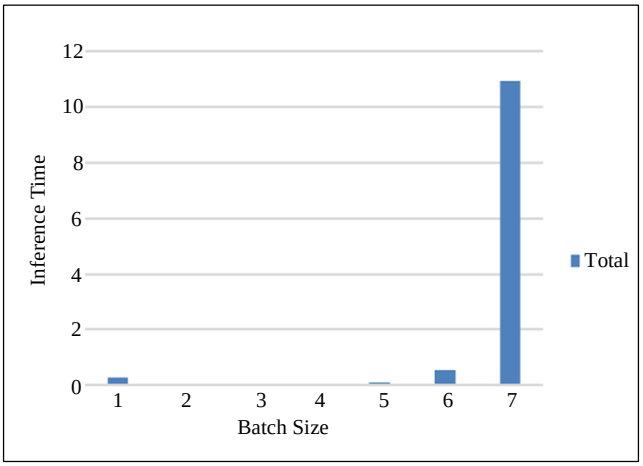

Fig. 5 Mobilenet V2 performance, inference time against batch sizes

4.2. Running MobileNet V3 as TensorRT Engine

Upon applying TensorRT optimization to the MobileNet V3 model on an SBD, a noteworthy enhancement in performance metrics is observed when juxtaposed with its PyTorch-based counterpart.

4.2.1. Optimization Effects Of Tensorrt On Mobilenet V3 in (Figure 6)

- The inference latency for the model post-TensorRT optimization remains commendably low across varying batch sizes, with an average duration of approximately 0.112 seconds.
- The observed standard deviation is minimal, at 0.010 seconds, reflecting a high degree of temporal consistency in model inference, a critical factor for time-sensitive applications.
- Remarkably, even at an increased batch size of 15, the model sustains a low inference time of about 0.097 seconds, outpacing the PyTorch version's average latency.
- The peak latency recorded for the TensorRT-optimized variant is around 0.127 seconds at batch size 21, a metric that aligns well with the requirements of real-time processing tasks.

The comparative analysis underscores the TensorRT optimization's efficacy in enhancing model performance, particularly in terms of consistency and resource optimization on the Jetson Nano. This improvement is manifested in the reduced variance of inference times and the model's ability to retain low latency across expanded batch sizes.

TensorRT's optimization mechanisms, such as layer fusion, efficient data format selection, and hardware-specific enhancements, play a pivotal role in this performance uplift.

These optimizations contribute to markedly reduced inference times, thereby ensuring more stable and predictable model behavior across a spectrum of batch sizes.

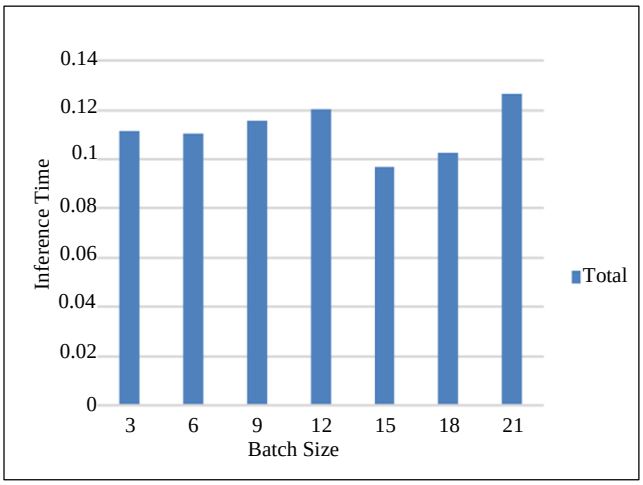

Fig. 6 MobileNet V3 (TensorRT) performance, inference time against batch sizes

5. Conclusion

In concluding this comprehensive analysis, it becomes evident that the optimization of the MobileNet V3 model using TensorRT substantially advances Jetson Nano platform capability in 3D bounding box generation performance. This optimized model showcases remarkable consistency and efficiency, rendering it exceptionally well-suited for deployment in real-time applications where rapid and predictable processing is paramount. The enhanced performance of MobileNet V3, when juxtaposed with the VGG-19 model, highlights a stark contrast. VGG-19, while robust in its capabilities, encounters significant challenges on the Jetson Nano, particularly with extended inference durations and increased variability in processing times. Such characteristics may hinder its applicability in circumstances timely response is req required.

On the other hand, the MobileNet V2 model demonstrates commendable functionality at lower batch sizes. However, it reveals discernible constraints as the batch size escalates, suggesting a potential compromise in performance under heightened operational demands. This observation underscores a pivotal consideration in edge computing environments, where computational resources are inherently limited. The selection of inherently lightweight models, coupled with the strategic application of optimization techniques like TensorRT, emerges as a critical strategy. This approach not only enhances the operational efficiency of the models but also ensures their resilience and reliability under varying workloads.

Drawing from real-world insights, the implications of such optimizations extend far beyond mere academic interest. In practical scenarios, such as autonomous vehicles, surveillance systems, and interactive AI applications, the ability to process information swiftly and reliably can be the difference between success and failure. In these contexts, the latency and predictability of model inference times are not just metrics but are integral to the safety, effectiveness, and user experience. Hence, the findings from this analysis not only contribute to the theoretical understanding of model optimization but also offer tangible guidelines for practitioners in the field of edge computing. They highlight the importance of model selection and optimization in achieving high-performance, real-time processing capabilities, essential for the burgeoning array of applications reliant on edge computing technologies.

Acknowledgments

The authors would like to acknowledge all Universiti Sains Malaysia (USM) staff and students, especially National Advanced IPv6 Center (NAv6), RCMO and BJIM staff, and those working under the Intelligent Connected Streetlights (ICS) research project for their full support, resulting in the publication of this paper.

Funding Statement

This paper is the outcome of the Intelligent Connected Streetlights (ICS) research project work supported by Renesas-Universiti Sains Malaysia (USM) industry matching grant as per MoA#A2021098 agreement with grant account no [7304.PNAV.6501256.R128].

References

[1] Arsalan Mousavian et al., "3D Bounding Box Estimation Using Deep Learning And Geometry," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, USA, pp. 5632-5640, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[2] Karen Simonyan, and Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv*, pp. 1-14, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[3] Joseph Redmon, and Ali Farhadi, "Yolov3: An Incremental Improvement," *arXiv*, pp. 1-6, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[4] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao, "YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors," *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Vancouver, Canada, pp. 7464-7475, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[5] Andrew Howard et al., "Searching For Mobilenetv3," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea (South), pp. 1314-1324, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[6] NVIDIA, Jetson Nano Developer Kit, NVIDIA Jetson Nano. [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/product-development/

[7] Ross Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, USA, pp. 580-587, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[8] Ross Girshick, "Fast R-CNN," *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile, pp. 1440-1448, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[9] Shaoqing Ren et al., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137-1149, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[10] Joseph Redmon et al., "You Only Look Once: Unified, Real-Time Object Detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, USA, pp. 779-788, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[11] Joseph Redmon, and Ali Farhadi, "YOLO9000: Better, Faster, Stronger," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, USA, pp. 6517-6525, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[12] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao, "Yolov4: Optimal Speed and Accuracy of Object Detection," *arXiv*, pp. 1-17, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[13] Glenn Jocher et al., "Ultralytics/Yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation," *Zenodo*, 2022. [CrossRef] [Publisher Link]

[14] Chuyi Li et al., "YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications," *arXiv*, pp. 1-17, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[15] Dr. Info Sec, VGG-19 Convolutional Neural Network, Machine Learning, 2021. [Online]. Available: https://blog.techcraft.org/vgg-19-convolutional-neural-network/

[16] NVIDIA TensorRT, NVIDIA Developer, [Online]. Available: https://developer.nvidia.com/tensorrt

[17] Sampurna Mandal et al., "Lyft 3D Object Detection for Autonomous Vehicles," *Artificial Intelligence for Future Generation Robotics*, pp. 119-136, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[18] Tan Zhang et al., "Sim2real Learning of Obstacle Avoidance for Robotic Manipulators in Uncertain Environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 65-72, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[19] Linh Kästner, Vlad Catalin Frasineanu, and Jens Lambrecht, "A 3D-Deep-Learning-Based Augmented Reality Calibration Method for Robotic Environments Using Depth Sensor Data," *2020 IEEE International Conference on Robotics and Automation (ICRA)*, Paris, France, pp. 1135-1141, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[20] Yutian Wu et al., "Deep 3D Object Detection Networks Using Lidar Data: A Review," *IEEE Sensors Journal*, vol. 21, no. 2, pp. 1152-1171, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[21] Yue Wang et al., "DETR3D: 3D Object Detection from Multi-View Images via 3D-to-2D Queries," *arXiv*, pp. 1-12, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[22] Pytorch, Models and Pre-Trained Weights. [Online]. Available: https://pytorch.org/vision/stable/models.html

[23] Laith Alzubaidi et al., "Review of Deep Learning: Concepts, CNN Architectures, Challenges, Applications, Future Directions," *Journal of Big Data*, vol. 8, pp. 1-74, 2021. [CrossRef] [Google Scholar] [Publisher Link]