YOLOv5 Inference Performance Analysis on Edge with different Machine Learning Frameworks

James Rigor Camacho, Angela Caliwag, Wansu Lim
Kumoh National Institute of Technology

Abstract

In recent years, one of the key innovations in manufacturing automation is the improvement of object detection and classification through surveillance. To have a more reliable system that performs monitoring in a manufacturing facility, machines are integrated with object detection algorithms to help ensure the quality and sorting of products accurately and efficiently. One of the recent and widely used deep learning-based object detection algorithms is the YOLOv5, also known as "You Only Look Once" version 5. YOLOv5 is supported by machine learning frameworks in deep learning, providing different execution performance. The goal of this paper is to determine the best suited framework for YOLOv5 to achieve the optimal hardware performance on edge based on the inference time and inference speed of the model. The performance comparative results revealed that among all the frameworks that were tested on the YOLOv5 model, TensorRT gave the best overall performance. Its average inference time to process the prepared test images is 133ms which has a percent difference of 22.11%, 41.70%, and 65.87% from the other frameworks such as ONNX, TensorFlow, and PyTorch, respectively. In the TensorRT framework, YOLOv5 was able to increase its hardware performance in terms of inference speed, yielding 8fps and 12fps, respectively, based on its specified precision of FP32 and FP16. Hence, YOLOv5 in TensorRT framework can be deployed in a resource-constrained device such as edge AI with improved inference performance for applications that involve object detection and classification.

Ⅰ. Introduction

Manufacturing industries have allowed innovation in their manufacturing processes to increase their economic productivity that could keep up with the increasing demand of the public. The production and consumption of commodities have increased since the beginning of the industrial revolution, and it is that innovation in the manufacturing industries have helped to transform and to boost their productivity in the services provided. These services have aimed at manufacturing flexibility and rapid product innovation to gain competitive advantages in the manufacturing industry [1]. Recent innovative ideas in the manufacturing industry include hybridizing the tracking system with a Convolution Neural Network (CNN)-based object detection [2]; and integrating an improved Singe Short Detector (SSD) network to a defect detection system in manufacturing [3].

These methods, such as Faster R-CNN and SSD, for object detection applications are popular for their efficient feature extraction ability [3]. The increase in demand for manufacturing automation has been realized with use of these deep-learning methods. It has become the interest in automated factories to develop and improve the production line performing complex and time-efficient tasks. However, object detection methods that have been recently proposed are not all well-suited to cover complex environments and to perform time-efficient tasks. Scenes that have imperfect background can affect the detection performance of a surveillance system [4]. A surveillance system in manufacturing would either need to purchase a better system or use a compact system integrated with the most efficient and accurate object detection model to improve the efficiency of object detection in a complicated background. This brings the need for a more reliable system that performs monitoring in a manufacturing facility and integrated with an object detection algorithm to help ensure the quality and sorting of products accurately and efficiently.

One of the widely used deep learning-based object detection algorithms is known as "You Only Look Once" or YOLO. It is a regression-based object detection system that delivers class probabilities for the discovered images omitting the region proposal process [5]. This process was performed differently and was compared to other state-of-the-art models, the summary of YOLOv5 comparison to state-of-the-art models is presented in Table I. From the table, the methods are analyzed based on the complexity of their model, flexibility, and inference speed. These are the factors that greatly affect the hardware performance of a model when deployed on an edge. In [6], Fast R-CNN was the inspiration of the framework proposed in the

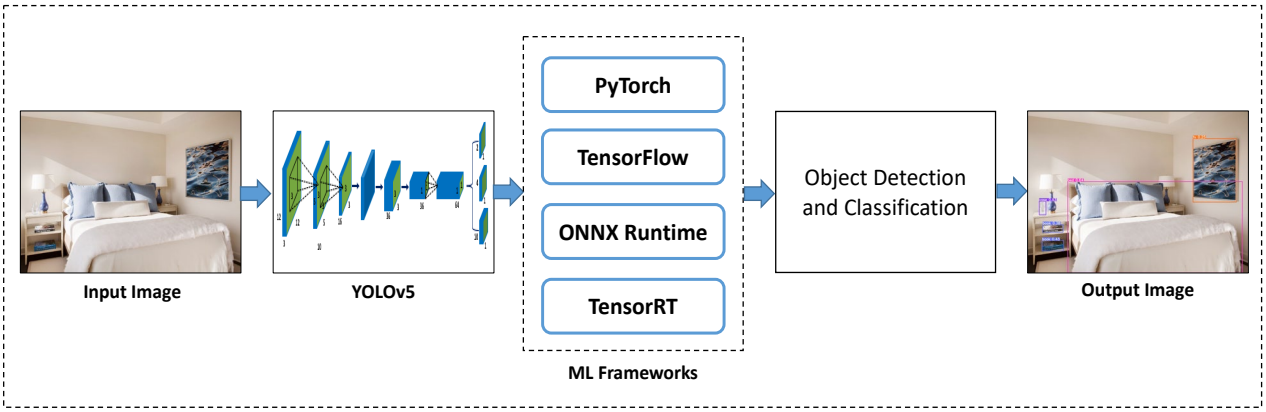

Fig. 1. Overview of the architecture for object detection and classification using YOLOv5 in different ML frameworks.

Table I
Summary of YOLOv5 comparison with state-of-the-art models for Object detection application

Method	Less Complexity	Flexibility	Inference Speed
Fast R-CNN [6]	✔	✘	7 FPS
SSD [7]	✔	✔	85 FPS
YOLOv5 [8]	✔	✔	155 FPS

paper. It is an object detection model that can achieve superior performance because of its CNN features. It is composed of two processes that involves regional proposal and object recognition [7]. Both parts of fast R-CNN can determine the object that appears on the image and specify the region proposal of that image. Which shows that the structure of this model has redundant functions and can be further improved to speedup the computation or processing of object detection. Morever, the region-based network of the model is not optimized which brought false detection results when applied on a complex environment. There are some studies that have tried to improve the outpuHence, it would require a lot of processing power when deployed on an embedded system. To resolve this constraint in an object detection system, the SSD method was introduced. In [8], SSD can handle complex environments by introducing default boxes in different aspect ratios to a larger input image. The object detected by the default boxes will be categorized based on the scores generated by the network. This predicts the object category and adjusts the box to better fit the shape of the object. Moreover, the model is flexible enough to be deployed in an embedded system and even applied for real-time applications. However, the inference performance of this model was inefficient for high quality real-time detection which brought to the idea of developing a YOLO model. Although the SSD has an inference speed of 85FPS, which is faster than the 7FPS of a Fast R-CNN model, the latest version of the YOLO series which is YOLOv5 [9] is capable of achieving 155FPS in real-time applications. Hence, among other models, YOLOv5 has been recorded to achieve faster speed, but lower accuracy. Considering its potential in the improvement of object detection algorithms, YOLO has become the hot topic by researchers to improve the surveillance system in manufacturing automation. The progression from the original version of YOLO up until its 5th version, has outperformed several models especially in terms of real-time inferencing. It is the latest version released for the YOLO series that is reliable, efficient, and simple to use.

In summary, although there are state-of-the-art models suitable for object detection applications, these models have some serious concerns. These concerns are as follows: 1) object detection algorithms produce false detection findings when applied in a complicated environment which indicates the lack of suitability and right information in the study, 2) the analysis in the improvement of the inference performance of a model was inefficient to determine a factor that contributes to the high quality real-time detection, and 3) deep-learning models when deployed on an embedded device would demand a lot of processing power to perform difficult and time-efficient tasks.

To address the aforementioned concerns, our study has focused on benchmarking of YOLOv5 using known frameworks like TensorRT, ONNX Runtime, Pytorch, and Tensorflow on edge AI devices. The performance of the model is analyzed based on the inference performance of the device. These parameters would determine which framework has the highest efficiency of the YOLOv5 on edge AI device. It is, thus, aimed to attain a high-efficiency YOLOv5 model with the lowest possible hardware requirements on deep learning applications. The experimental analysis results will decide and explain which framework is best suitable for YOLOv5 on edge to conduct an accurate and efficient object detection task in a manufacturing environment.

Ⅱ. Methodology

The overview for the architecture of object detection and classification using YOLOv5 on different machine learning frameworks is shown in Fig.1. This pipeline is composed of input images, YOLOv5 model, machine learning frameworks (i.e., PyTorch, TensorFlow, ONNX, and TensorRT), object detection

Fig. 2. NVIDIA Jetson Nano (Edge AI device).

and classification algorithm, and output images. A set of input images is fed into the pretrained YOLOv5 model where different machine learning frameworks are used in performing the object detection and classification algorithm. The output images are expected to have bounding boxes on the objects detected with their classifications. The inference time and speed of detecting objects from each image will also be outputted which varies depending on the ML framework that was used.

A. The Network Architecture

YOLO is a family of deep learning-based object detection architectures and models pretrained on the COCO dataset. Before YOLO, the process of classifying objects in an input image was difficult, slow-moving, and not very energy efficient. Instead of going through the same process, the YOLO algorithm simplifies object detection and classification tasks by treating it as a regression problem and employing a one-stage algorithm to predict object classes and positions immediately. Specifically, it divides the input images into a grid system and makes predictions on all the objects detected in the images by computing all the features at the same time. It can precisely predict and locate the object using bounding box coordinates.

Dataset: To evaluate the performance of the YOLOv5 model based on the inference speed at different frameworks, a benchmark data set is utilized. Fig.3 contains the test images that were picked from the pool of available images online aimed to best demonstrate the object detection and classification capabilities of the model. These images show the possible installation and applications of this study namely, pedestrian, streets, school facilities, backyard/garden, urban park, bus stop, and different parts of a house. Moreover, these images can be found in [10-19]. Hence, the dataset is important when re-training a model for a specific application. The more dataset is used to train the model, the higher the improvement in its accuracy and performance.

YOLOv5: It is the latest version released for the YOLO object detection series which is 3x faster as compared with YOLOv4. By just looking at the image

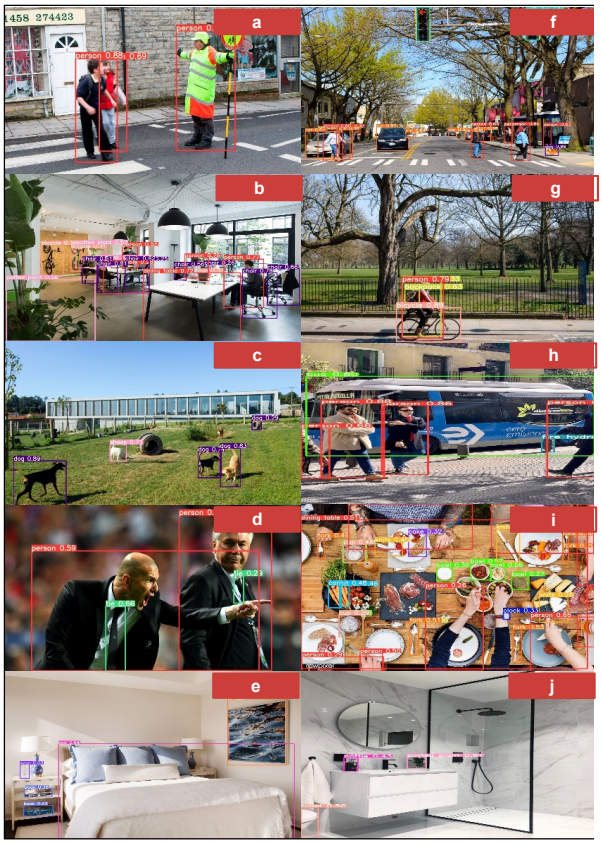

Fig. 3. Test Images. (a)Pedestrian, (b)Office, (c)Park, (d)Zidane, (e)Bedroom, (f)Road, (g)Bike, (h)Bus Stop, (i)Dinning, (j)Bathroom.

once, it can detect the objects with an expected speed of 140 FPS (Frames Per Second). In addition to that, the size of YOLOv5 is 9x smaller making it suitable to efficiently deploy on edge AI devices having low computational capability. YOLOv5 is supported by machine learning frameworks in deep learning, providing different execution performance.

Frameworks: YOLOv5 was introduced with a PyTorch framework. The aspiration of researchers to continuously improve the architecture of the YOLO model has brought to the idea of converting a model defined in PyTorch into a different framework format (i.e., ONNX, TensorFlow, and TensorRT). These frameworks have a corresponding effect on the performance of the YOLOv5 model in terms of inference speed and accuracy. A benchmarking analysis on the results is undertaken and compared statistically and qualitatively with state-of-the-art models to evaluate the effectiveness of each framework and investigate the performance of the YOLOv5 model in an edge AI device.

B. Hardware Platform

The hardware platform used as an edge AI device in the experimental setup is the NVIDIA Jetson Nano device. Fig.2 shows the actual device where the YOLOv5 model is deployed and performs the object detection and classification tasks. It is a resource-constrained device, which means that it has limitations on resources such as computational power and memory

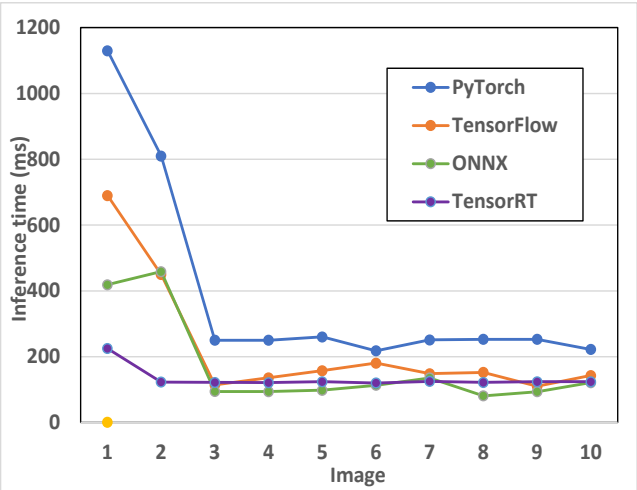

Fig. 4. Inference time of processing test images in different machine learning frameworks.

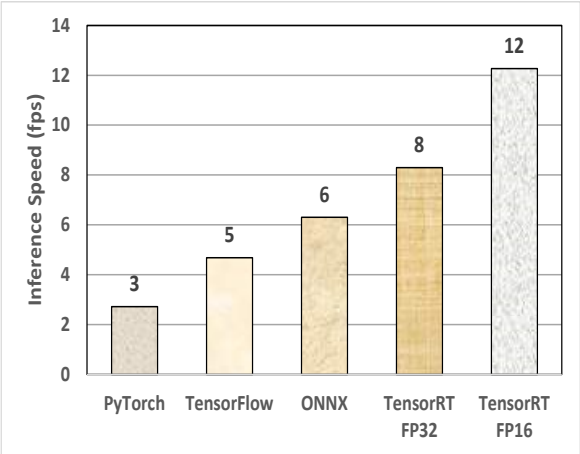

Fig. 5. Inference speed of different ML framework With NVIDIA Jetson Nano.

demand. For the YOLOv5 model, this is the perfect hardware platform to analyze the hardware performance of the model with different frameworks.

It is important to understand the behavior of these models in actual implementation since most applications would require installation setup. It is compact and compatible with deep learning-based models, specifically, YOLO models since the structure have less complexity and less energy requirements.

Ⅲ. Results and Discussion

In this section, the experimental setup and results are discussed to provide a comprehensive benchmarking analysis of the YOLOv5 model with different machine learning frameworks.

Experimental Setup: NVIDIA Jetson Nano was used as the edge AI device to deploy the deep learning network, YOLOv5 model, in different frameworks for object detection and classification tasks. This edge AI device is popular for research that is related to building autonomous machines and complex AI systems because of its compactness and capabilities. Despite being small, it is a powerful edge AI device that is well-suited for real-world application and experimental setup. Deploying the model in this device provides the inference speed and inference time of processing object detection and classification tasks in every input image. The hardware performance of the YOLOv5 model in different frameworks is evaluated with the following parameters: inference speed and inference time of the model design.

A. Deep Learning Inference Performance

The YOLOv5 model was implemented with different frameworks and ran inference on the test images. The results for the inference time of processing the prepared test images in different ML frameworks is shown in Fig.4. Given the trend on the inference time of these ML frameworks, it gives the intuition that Pytorch took the longest time to identify and classify the objects in the test images. On the other hand, TensorRT is the framework that has the fastest average inference time of 133ms to process the test images. Despite that, all the frameworks for YOLOv5 were able to maintain their average inference time for the succeeding test images starting the 3rd image. Moreover, among all the test images, the image (i) for dinning in Fig.3 took the longest for all the frameworks to detect and classify the objects in the image. The resulting percent difference of the different frameworks from the overall average inference time are 7.71%, 37.23%, 37.46%, and 39.22% for TensorRT, ONNX, TensorFlow, and PyTorch, respectively.

For the inference speed of the different frameworks, the YOLOv5 is deployed on edge AI devices and the results are recorded during the performance test. The graph in Fig.5 for inference speed of different ML frameworks with NVIDIA Jetson Nano shows that TensorRT FP16 and TensorRT FP32 got the fastest inference speed among the other frameworks with an inference speed of 12fps and 8fps, respectively. The result also shows that the YOLOv5 model at TensorRT FP16 format is 2x faster than the ONNX framework and 4x faster than the PyTorch framework. In reference to the average inference speed of the YOLOv5 model in different frameworks, the percent difference of their individual results is a 9.95%, 18.46%, 33.20%, 57.14%, and 75.29% for ONNX, TensorRT FP32, TensorFlow, PyTorch, and TensorRT FP16, respectively.

From the overall performance of the YOLOv5 model on edge for object detection and classification tasks, TensorRT framework got the best performance based on the inference time and inference speed results. TensorRT managed to process all the test images within an average of 133ms (inference time) with an inference speed of 8fps for FP32 and 12fps for FP16. To achieve an optimal performance for the YOLOv5 model, the best-suited framework to use is the TensorRT FP32. It might not have the fastest inference speed compared to FP16, but it has the most stable and high output results, which is 9.95% percent difference from the overall average inference speed, compared to all other frameworks.

Ⅳ. Conclusion

In this paper, a performance comparative study is performed in the YOLOv5 model with different ML frameworks to determine which model architecture is best suited for the application of object detection and classification tasks. This paper utilized an edge AI device to observe the hardware performance of the YOLOv5 model through its inference time and inference speed. Its goal is to develop a YOLOv5 model that can run on low-resource devices and give real-time performance. In terms of overall performance for the YOLOv5 model, the acquired average inference speed is 6fps which is 2x slower on PyTorch but 2x faster on the TensorRT FP16 framework. For inference time, TensorRT got the fastest average time of 133ms which has a percent difference of about 22.11%, 41.70%, and 65.87% for ONNX, TensorFlow, and PyTorch, respectively. Therefore, based on the results, TensorRT is the best suited framework for YOLOv5 with an average inference time of 133ms and an inference speed of 8fps and 12fps for TensorRT FP32 and TensorRT FP16, respectively. It was proven to provide the best inference performance for the YOLOv5 model on the application of object detection and classification algorithms. For future works, TensorRT provides INT8 optimization as well which can be used to other Jetson series like NVIDIA Jetson TX2 and NVIDIA Jetson Xavier. This could further optimize the inference performance of the YOLOv5 model by enabling the FP16 in the TensorRT framework.

ACKNOWLEDGMENT

This work was supported by the Ministry of SMEs and Start-ups, S. Korea (S2829065, S3010704), and by the National Research Foundation of Korea (2020R1A4A101777511, 2021R1I1A3056900).

References

[1] Bi Xinhua, Chen Taibo, Yu Baojun and Yu Cuiling, "Manufacturing flexibility and rapid product innovation: Two key manufacturing decision issues in turbulent business environment," 2009 Chinese Control and Decision Conference, 2009, pp. 4963-4967, doi: 10.1109/CCDC.2009.5194921.

[2] M. Jiang, K. Shimasaki, S. Hu, T. Senoo and I. Ishii, "A 500-Fps Pan-Tilt Tracking System With Deep-Learning-Based Object Detection," in IEEE Robotics and Automation Letters, vol. 6, no. 2, pp. 691-698, April 2021, doi: 10.1109/LRA.2020.3048653.

[3] J. Yang, S. Li, Z. Wang and G. Yang, "Real-Time Tiny Part Defect Detection System in Manufacturing Using Deep Learning," in IEEE Access, vol. 7, pp. 89278-89291, 2019, doi: 10.1109/ACCESS.2019.2925561.

[4] Y. Tian, R. S. Feris, H. Liu, A. Hampapur and M. Sun, "Robust Detection of Abandoned and Removed Objects in Complex Surveillance Videos," in IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 41, no. 5, pp. 565-576, Sept. 2011, doi: 10.1109/TSMCC.2010.2065803.

[5] W. Liu et al. (Dec. 2015). "SSD: Single shot multibox detector."[Online].Available:https://arxiv.org/abs/1512.02325

[6] Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: CVPR. (2016)

[7] S. Hsu, C. Huang and C. Chuang, "Vehicle detection using simplified fast R-CNN," 2018 International Workshop on Advanced Image Technology (IWAIT), 2018, pp. 1-3, doi: 10.1109/IWAIT.2018.8369767.

[8] H. Li, L. Deng, C. Yang, J. Liu and Z. Gu, "Enhanced YOLO v3 Tiny Network for Real-Time Ship Detection From Visual Image," in IEEE Access, vol. 9, pp. 16692-16706, 2021, doi: 10.1109/ACCESS.2021.3053956.

[9] X. Wang, H. Ma, X. Chen and S. You, "Edge Preserving and Multi-Scale Contextual Neural Network for Salient Object Detection," in IEEE Transactions on Image Processing, vol. 27, no. 1, pp. 121-134, Jan. 2018, doi: 10.1109/TIP.2017.2756825.

[10] M. Waghorn and (image: Getty), "Children can't safely cross the road until they are over 14 years old, scientists warn," Mirror.co.uk, 20-Apr-2017. [Online]. Available: https://www.mirror.co.uk/news/uk-news/children-cant-safely-cross-road-10259133.

[11] "3d visualization architecture," Service4money.com. [Online]. Available: https://www.service4money.com/3d-visualization-architecture/.

[12] "Dogs-and-cats-hotel-by-raulino-Silva-arquitecto-00," Aasarchitecture.com. [Online]. Available: https://aasarchitecture.com/2020/10/dogs-and-cats-hotel-by-raulino-silva-arquitecto.html/dogs-and-cats-hotel-by-raulino-silva-arquitecto-00/.

[13] "Ancelotti looks forward to Madrid-Bayern showdown with 'icon' Zidane," Loop News, 02-Apr-2017. [Online]. Available: https://tt.loopnews.com/content/ancelotti-looks-forward-madrid-bayern-showdown-icon-zidane.

[14] H. Mendelsohn, "If you've always wanted to sleep on a cloud, copy these white bedroom ideas," Housebeautiful.com, 01-Apr-2012. [Online]. Available: https://www.housebeautiful.com/room-decorating/colors/g1215/white-bedrooms/.

[15] S. Davis, "Seattle takes new steps to fine-tune traffic signals for people walking and rolling during COVID-19 health crisis," Seattle.gov, 07-May-2020. [Online]. Available: https://sdotblog.seattle.gov/2020/05/07/seattle-takes-new-steps-to-fine-tune-traffic-signals-for-people-walking-and-rolling-during-covid-19-health-crisis/.

[16] "Bloomberg," Bloomberg News.

[17] "YOLOv5 environment construction and target detection - Programmer Sought," Programmersought.com. [Online]. Available: https://programmersought.com/article/52126720263/.

[18] "Group of people dining concept," 123Rf.com. [Online]. Available: https://www.123rf.com/photo_67127031_group-of-people-dining-concept.html.

[19] Thetrendspotter.net. [Online]. Available: https://www.thetrendspotter.net/small-bathroom-design-ideas/