Implementation of the Object Detection Algorithm (YOLOV3) on FPGA

1st Fahri Esen
Electrical and Electronics Engineering
Ankara Yıldırım Beyazıt University
Ankara, Turkey
fahri9esen@gmail.com

2nd Ali Degirmenci
Electrical and Electronics Engineering
Ankara Yıldırım Beyazıt University
Ankara, Turkey
adegirmenci@ybu.edu.tr

3rd Omer Karal
Electrical and Electronics Engineering
Ankara Yıldırım Beyazıt University
Ankara, Turkey
karal@ybu.edu.tr

2021 Innovations in Intelligent Systems and Applications Conference (ASYU) | 978-1-6654-3405-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ASYU52992.2021.9599073

Abstract—**From past to present, traffic has always been an important problem in society. Loss of life and material losses due to traffic accidents are quite high. The vast majority of these accidents are closely related to people who drive incorrectly and unconsciously. Therefore, researchers propose various methods to solve this problem. Recently, deep learning-based models have been widely used for this purpose. However, these models such as single shot detection (SSD), regions convolutional neural network (R-CNN) and fast regions convolutional neural network (Fast R-CNN) cannot perform real-time object recognition. Because, image processing time (speed) is too long. In this study, the you only look once version 3 (YOLOV3) model, which is one of the deep learning applications, is introduced in real-time to track cars in traffic. YOLOV3 can track objects in real-time. However, the hardware required for the model to give better and faster results is not sufficient today. For this reason, ZCU102 field programmable gate arrays (FPGA) card is used. It has been observed that faster and more accurate results are obtained with the YOLOV3 model run on the FPGA card.**

Index Terms—**field programmable gate arrays (FPGA), deep learning processing unit (DPU), you only look once version 3 (YOLOV3), convolutional neural network, deep learning**

I. Introduction

Traffic safety is a very important issue for human life. Hundreds of people die every day due to traffic accidents. In addition to the loss of life, there are also some problems caused by not following the traffic rules. For example, traffic congestion can be given as an example of one of these problems. Although some solutions have been applied to prevent such problems, traffic safety has not been fully resolved. In recent years, machine learning (ML) models have been developed to minimize traffic safety problems, and rapidly continue to be improved further. However, most proposed models fail to perform real-time object recognition. Because the developed deep learning models have a complex structure, they demand high-quality technology. For this reason, today's hardware is usually run on a graphics processing unit (GPU). But this equipment becomes insufficient after a while. Because, GPUs with fixed hardware are insufficient to run deep learning algorithms that are getting cumbersome day by day. However, since field programmable gate arrays (FPGA) cards do not have a fixed hardware, they can be redesigned in accordance with deep learning algorithms.

When comparing GPU and FPGA, it is clear that both have advantages over each other. In terms of ease of use, GPU hardware usage is better than FPGA hardware. In FPGA, hardware is created by the user while it is ready-made in the GPU. So, FPGAs are a little more complex and take longer time to create a hardware according to the application which is private to the user's project. Additionally, GPUs are also more advantageous than FPGAs due to the fact that they run depending on the operating system and thus deep learning algorithms are faster and easier to adapt. On the other hand, the GPU hardware produced today has a fixed structure and the desired result cannot be achieved because the delay obtained cannot be reduced to a certain level. In these cases, the need for more than one GPU arises. They also cause an increase in both power consumption and cost. FPGA hardware does not have a fixed structure (creating the hardware according to the user's requirements) provides a great advantage in terms of making the desired hardware. A very reasonable price comparison cannot be made between GPU and FPGA. In the GPU, different models have to be used according to the requirements because that have a fixed structure, causing the price to vary greatly. Since the FPGA allows the creation of hardware suitable for the needs the same FPGA can be used in different applications [1], [2].

Due to the advancement of deep learning and its recent popularity, the use of deep learning algorithms on FPGA [3] cards has begun to rise. Because, the fast and parallel processing of FPGA cards increases its importance in deep learning applications [4], [5].

Some of deep learning models used in object recognition applications to date are regions convolutional neural network (R-CNN), Fast R-CNN, Faster R-CNN [6], single shot detection (SSD) [7] and you only look once (YOLO) [8]. Each of these deep learning algorithms has its own advantages and disadvantages. Although some models run fast, their accuracy is low. Some models have very high accuracy rates, but their speed is very low. Therefore, the model used must be as fast as possible and with high accuracy.

In the R-CNN-based machine learning model, images are processed one by one. A Selective Search mechanism is applied to extract regions of interest (ROI) from the images. Here, ROI is a rectangle that can represent the boundary of any

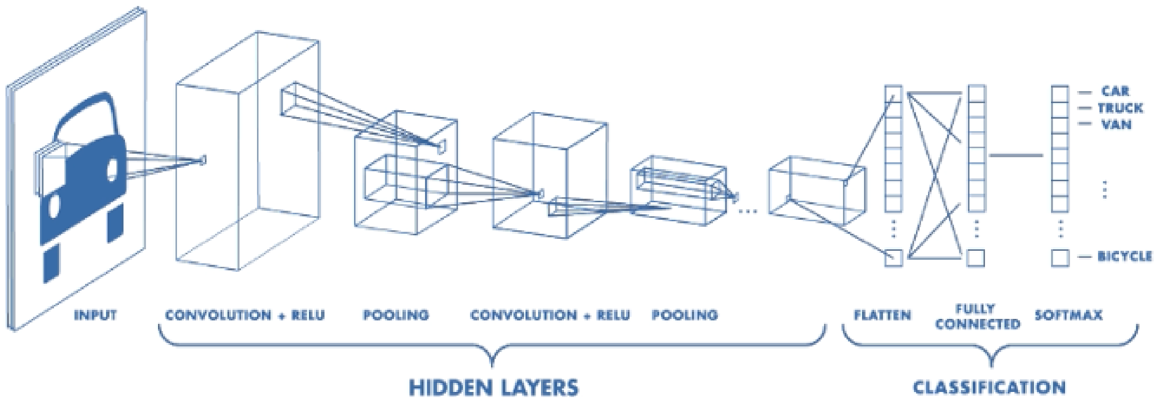

Fig. 1. YOLOV3 model architecture [9]

object in the image. After that, output features are generated for each ROI with a user-designed neural network. With these output features, multiple support vector machine is trained and it is determined what type of object (if any) is in the ROI.

The R-CNN model offers thousands of ROI suggestions from a single input image. The system requires thousands of forward propagated CNNs to handle successive images. This massive computing load makes extensive use of R-CNNs in real-world applications. In order to speed up the original R-CNN, a fast R-CNN is proposed that allows the output features of each ROI to be determined by running the forward propagated CNN once over the entire image. Like the original R-CNN, Fast R-CNN uses also Selective Search to generate ROIs [6].

For more accurate object detection, the fast R-CNN model often generates multiple region suggestions in selective search. In order to reduce region suggestions without loss of accuracy, a faster R-CNN model is developed that proposes replacing selective search with a region recommendation network.

As with other R-CNN models, the faster R-CNN model has two stages: 1) region recommendation, 2) classification of these regions and improvement of position estimation. Although they increase the accuracy, the processing time is longer compared to the SSD model. To compensate for the drop in accuracy, the SSD implements various improvements such as multi-scale features and default boxes. These improvements also increase speed, with results close to (sometimes even surpassing) the accuracy of Faster R-CNN even on lower resolution images.

Finally, the YOLO algorithm uses a completely different approach than the algorithms mentioned above. It can predict the class and coordinates of all objects in the image at once by passing the image through a single neural network. Therefore, it can produce more accurate results in a shorter time

YOLO is the fastest and most accurate model among these deep learning models. Most researchers have tested the YOLO model on the CPU or GPU. With the development of FPGA chips, deep learning models have recently begun to be run on FPGA chips.

Considering all these conditions, in this study, an object recognition algorithm will implement on the FPGA card to solve the traffic safety problem. In addition, the same application will perform on the GPU and compare with the FPGA results.

The remainder of the article is organized as follows, Section 2 describes the method used and followed. Section 3 clearly explains how the experiment was done, in order. Section 4 shows the experimental results of the system. Finally, Section 5 concludes the article.

II. Method

With the development of machine learning algorithms in recent years, object tracking and object detection applications have started to be used in many areas. One of the most important issues to be considered in these applications is to run the application in real time. For this reason, the YOLOV3 method, which is one of the machine learning applications, has been built in a structure suitable for working in real time for object tracking [8]. Inspired by the increasing interest in deep learning applications, in this study, the YOLOV3 method is used for object tracking.

A. *You Only Look Once Version 3 (YOLOV3)*

YOLO is an algorithm that uses artificial neural networks to detect objects in the image with their positions at once. It splits the image into cells and yields output vectors that try to estimate confidence values and bounding boxes for the objects in each cell. It works on the Darknet framework developed using C/Cuda. In this way, it shows a very high performance. Structurally, it is inspired by the GoogleNet structure, which consists of 24 convolutions and 2 dense layers.

The third version of YOLO was released in April 2018 [10]. The main innovation with this version is; it is the determination of bounding boxes for different sizes. The basic layer structure has been increased to 53. The YOLOV3 architecture is shown in Figure 1.

B. Experiment

First of all, before starting the training of the YOLOV3 model on dataset, establishment of the necessary libraries and the environment is completed. Open source car images and labels offered by Google were downloaded from Open Images Dataset website. With the help of the installations and the downloaded images, the training process on the YOLOV3 model is carried out quickly thanks to the GPU hardware. As a result of the training, weights file and configuration file are obtained (X.weights and X.cfg). Then, test images on the trained model are tested with the help of the GTX 950 GPU. The accuracy and frames per second (FPS) values obtained as a result of the test images are recorded and the same model is ready to work on the ZCU102 FPGA card shown in the Figure 2.

In order to run the model on the FPGA card, Petalinux operating system is installed on the FPGA card and the necessary libraries are installed [11]–[13]. In order for the model files obtained with the help of the GPU to be run on the FPGA board, the model files must be in the Protobuf file format. Protobuf file transformation is accomplished. Later pruning operations and bit reduction operations are performed on the obtained X.pb file [14]–[17]. The model file obtained as a result of the improvements is transformed into an .elf file in order to work on the FPGA board. Now the model is ready to be used on the FPGA board. The model file can run directly on the CPU part of the FPGA. But what we want to do is to speed up the object recognition process by working together with processing subsystem (PS) and programmable logic (PL). For this reason, short code is written in C++ to communicate the deep learning processing unit (DPU) block in the PL section with the PS part [18]–[23]. So, the model which is trained in the GPU environment ran on the FPGA board and compared the FPS difference between them.

Fig. 2. ZCU102 FPGA card

III. Experimental Results

The loss values obtained during the training of the model trained with 19.652 images are clearly shown in the Figure 3.

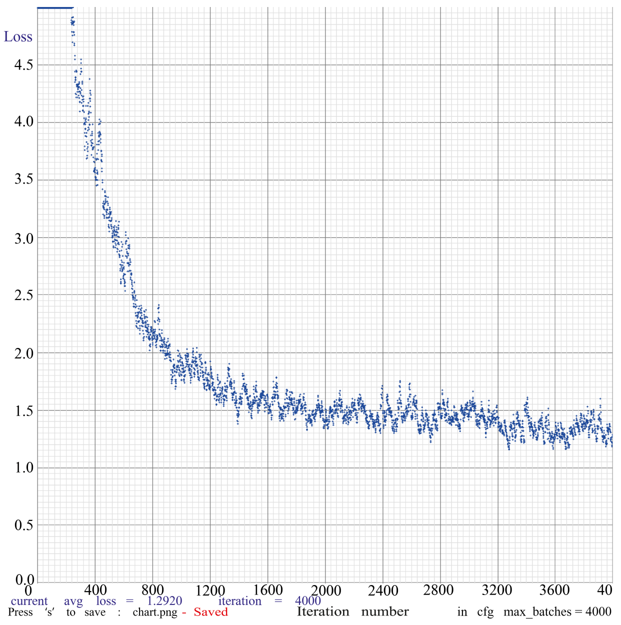

Fig. 3. YOLOV3 training Loss

The video file in the Figure 4 is run on the GPU and FPGA. As a result of the tests, the FPS value on the Nvidia GTX 950 GPU is found to be 15.8. On the other hand, the test on ZCU102 FPGA board, 31.2 FPS value is measured.

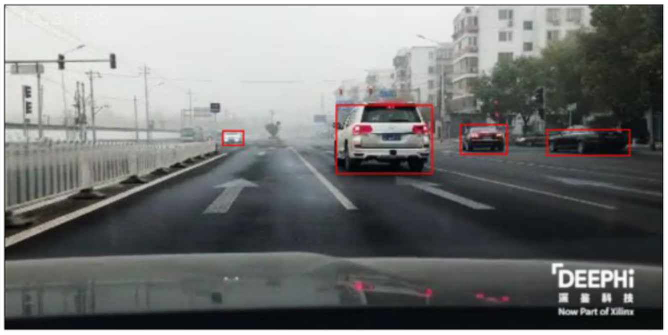

Fig. 4. 512x216 input video

When the video in the Figure 5 is given as an input model, the FPS value received on the GPU is maximum 15.6 while the FPS value on the FPGA board is measured as 16.2.

The input of the YOLOV3 model used in previous experiments takes the incoming image as 416x416. FPS values received should be evaluated accordingly. In this experiment, the input of the YOLOV3 model was set to 224x224.

FPS results of FPGA board and GPU are given in Table I. According to the performance results in Table I, the FPS

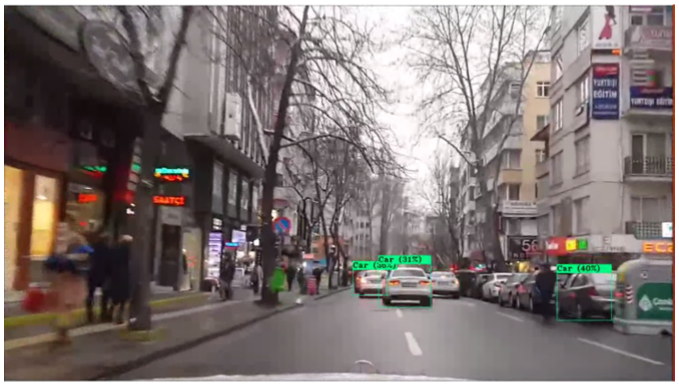

Fig. 5. 1280x720 input video

value taken on the GPU is measured as maximum 34.1 while the FPS value on the FPGA card is measured as 41.1.

TABLE I
PERFORMANCE RESULTS

Hardware	Performance (FPS)			
	Neural Network (YOLOV3 416x416)		Neural Network (YOLOV3 224x224)	
	512x216 Input Video	1280x720 Input Video	512x216 Input Video	1280x720 Input Video
ZCU102 UltraScale + MPSoc FPGA Board	31.2	16.2	41.1	26.8
Nvidia GTX 950 GPU	15.8	15.6	34.1	22.3

As can be seen from the experimental results of these videos, the working speed of the object detection model depends on the size of the image. In addition, the size of the input accepted by the model file also affects performance.

IV. CONCLUSION

The dataset obtained in the study is trained on the YOLOV3 model with the help of GTX950 GPU. In addition, it is provided to work on the ZCU102 FPGA card and the operating speeds of the GPU and FPGA card are compared. According to the observed experimental results, the best results are obtained from the FPGA card.

In future studies, faster running YOLOV3-tiny model can be used. By increasing the amount of dataset, the probability of recognizing objects can be increased. By changing the model input size, the model can be run faster.

REFERENCES

[1] Y. Zheng, Z. Shi, C. He, and Q. Zhang, "Lifting Based Object Detection Networks of Remote Sensing Imagery for FPGA Accelerator," IEEE Access, vol. 8, pp. 200430-200439, 2020.

[2] M. Chen, and P. Liu, "A Deep Learning-Based FPGA Function Block Detection Method With Bitstream to Image Transformation," IEEE Access, vol. 9, pp. 99794-99804, 2021.

[3] TUL ZCU102 [ONLINE] Available at: `https://www.xilinx.com/support/documentation/boards_and_kits/zcu102/ug1182-zcu102-eval-bd.pdf`

[4] I. Martinez-Alpiste, G. Golcarenarenji, Q. Wang, and J. M. Alcaraz-Calero, "A dynamic discarding technique to increase speed and preserve accuracy for YOLOv3," Neural Computing and Applications, 1-13, 2021.

[5] D. Pestana, P. R. Miranda, J. D. Lopes, R. P. Duarte, M. P. Véstias, H. C. Neto, and J. T. De Sousa, "A Full Featured Configurable Accelerator for Object Detection With YOLO," IEEE Access, vol. 9, pp. 75864-75877, 2021.

[6] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," Advances in neural information processing systems, vol. 28, pp. 91-99, 2015.

[7] Z. Chen, K. Wu, Y. Li, M. Wang, and W. Li, "SSD-MSN: an improved multi-scale object detection network based on SSD," IEEE Access, vol. 7, pp. 80622-80632, 2019.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi,"You only look once: Unified, real-time object detection," In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779-788, 2016.

[9] K. Alderliesten (2020, May) YOLOv3 — Real-time object detection [ONLINE] Available at: `https://medium.com/analytics-vidhya/yolov3-real-time-object-detection-54e69037b6d0`

[10] J. Redmon, and A. Farhadi, "Yolov3: An incremental improvement," arXiv preprint arXiv:1804.02767, 2018.

[11] TUL PetaLinux [ONLINE] Available at: `https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_3/ug1144-petalinux-tools-reference-guide.pdf`

[12] TUL DPU [ONLINE] Available at: `https://www.xilinx.com/support/documentation/ip_documentation/dpu/v3_2/pg338-dpu.pdf`

[13] TUL DNNDK [ONLINE] Available at: `https://www.xilinx.com/support/documentation/user_guides/ug1327-dnndk-user-guide.pdf`

[14] Xilinx. (2020, April) AI-Model-Zoo [ONLINE] Available at: `https://github.com/Xilinx/AI-Model-Zoo/tree/7f3456b26724cc649960e3b6924488859eebe489`

[15] Qqwweee. (2018, July) keras-yolo3 [ONLINE] Available at: `https://github.com/qqwweee/keras-yolo3`

[16] Wutianze. (2020, September) dnndk-pynqz2 [ONLINE] Available at: `https://github.com/wutianze/dnndk-pynqz2`

[17] Gewuek. (2020, August) flower_classification_vai_tf_dataset [ONLINE] Available at: `https://github.com/gewuek/flower_classification_vai_tf_dataset`

[18] S. Fang, L. Tian, J. Wang, S. Liang, D. Xie, Z. Chen, ... Y. Wang, "Real-time object detection and semantic segmentation hardware system with deep learning networks," International Conference on Field-Programmable Technology (FPT), pp. 389-392, December 2018.

[19] A. Kojima, and Y. Osawa, "Design and Implementation of Autonomous Driving Robot Car Using SoC FPGA," International Conference on Field-Programmable Technology (ICFPT), pp. 441-444 December 2019.

[20] J. Zhu,L. Wang, H. Liu, S. Tian, Q. Deng, and J. Li, "An efficient task assignment framework to accelerate DPU-based convolutional neural network inference on FPGAs," IEEE Access, vol. 8, pp. 83224-83237, 2020.

[21] R. Kedia, S. Goel, M. Balakrishnan, K. Paul, and R. Sen, "Design space exploration of FPGA based system with multiple DNN accelerators," IEEE Embedded Systems Letters, 2020.

[22] T. Wu, W. Liu, and Y. Jin, "An end-to-end solution to autonomous driving based on xilinx fpga," International Conference on Field-Programmable Technology (ICFPT), pp. 427-430, December 2019.

[23] Y. Lei, Q. Deng, S. Long, S. Liu, and S. Oh, "An Effective Design to Improve the Efficiency of DPUs on FPGA," IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS), pp. 206-213, December 2020.