Proceedings of the 7th International Conference on Trends in Electronics and Informatics (ICOEI 2023)
IEEE Xplore Part Number: CFP23J32-ART; ISBN: 979-8-3503-9728-4

# Implementation of Real-Time Object Detection on FPGA

Gayatri Rathod
Electronics and Communication Engineering Department
Institute of Technology, Nirma University
Ahmedabad - 382481, Gujarat, India.
Email: gcrathod987@gmail.com

Parin Shah
Electronics and Communication Engineering Department
Institute of Technology, Nirma University
Ahmedabad - 382481, Gujarat, India.
Email: parinjshah0526@gmail.com

Ruchi Gajjar
Electronics and Communication Engineering Department
Institute of Technology, Nirma University
Ahmedabad - 382481, Gujarat, India.
Email: ruchi.gajjar@nirmauni.ac.in

Manish I. Patel
Electronics and Communication Engineering Department
Institute of Technology, Nirma University
Ahmedabad - 382481, Gujarat, India.
Email: manish.i.patel@nirmauni.ac.in

Nagendra Gajjar
Electronics and Communication Engineering Department
Institute of Technology, Nirma University
Ahmedabad - 382481, Gujarat, India.
Email: nagendra.gajjar@nirmauni.ac.in

2023 7th International Conference on Trends in Electronics and Informatics (ICOEI) | 979-8-3503-9728-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICOEI56765.2023.10125958

***Abstract*—Object detection is a common and challenging problem in today's world. Also due to the rapid development of deep learning employing underlying deep models over the past ten years, many researchers have investigated and contributed to improving the performance of object identification and associated tasks such as object classification, localization, and segmentation. Inference time and detection accuracy are used to rate the effectiveness of any object detector. In terms of detection accuracy, it is observed that the two-stage detectors outperform the single-stage detector however, single-stage detectors outperform their rivals in terms of inference time. Additionally, You Only Look Once (YOLO) and its versions have improved detection accuracy, sometimes surpassing two-stage detectors. In this study, we performed a co-design methodology for hardware and software (HW/SW) aimed at Central Processing Unit (CPU) + Field Programmable Gate Array (FPGA) based heterogeneous platform which is Xilinx FPGA Kria KV260. A CNN-based algorithm is first extended to the YOLOv4, v5L, and v5s framework before being implemented on the Kria KV260, an out-of-the-box platform for developing advanced vision applications. A reduced computation time is observed for the deployment of the Machine Learning model on Kria KV260.**

*Index Terms - Object Detection, Edge Computing, Machine Learning, Field Programmable Gate Array, Kria KV260*

## I. Introduction

Due to the rapid growth of deep learning over several years, researchers have tried to upgrade the process of object identification, its categorization, and utilization of the underlying deep learning models. Every object class has certain particular characteristics that help identify the class. Efficiently detecting the object class is a challenging task for object detection models. For real-time object detection, machine learning algorithms are not only completely responsible to categorise and localize significant objects as well they should be fast enough regarding the time for a prediction. Major challenges for object detection include dual priorities, the first is of categorizing an object and the second is of figuring out where it is (this is known as the object localization task). The majority of models are developed and tested in ideal environments, making it difficult for detectors to identify objects from various angles. The varying object size and aspect ratio make it hard for object detection algorithms to recognize different objects. A partially visible object can sometimes be challenging to find.

Researchers most frequently employ a multi-task loss function for object localization tasks or to handle this problem in order to penalize both incorrect classifications and localization errors. These methods involve generating suggestions for regions where items are mostly discovered before using Convolutional Neural Network (CNN) to classify and fine-tune object placements. While accuracy also increases as a result of the activities including classification and location being improved utilizing one multi-task loss function, models such as fast R-CNN offers a significant speedup. These algorithms are now substantially faster than they were previously, from R-CNN's (0.02) fps to YOLO's (155) fps [1]. As there are many applications for Object detection, important objects may be present in different sizes and ratios. People use a variety of strategies, such as anchor boxes, multiple feature maps, and feature pyramid networks, to guarantee that detection algorithms can catch objects at

different scales and angles.

When compared to real-time object detectors, YOLO has the inherent advantage of speed in addition to better Intersection over Union in bounding boxes and increased forecast accuracy. YOLO is a far quicker algorithm than its rivals, operating at up to 45 frames per second (fps) [2]. YOLO executes all of its predictions using a single fully connected layer, in contrast to Faster R-CNN and comparable algorithms, which employ the Region Proposal Network to find prospective regions of interest before doing recognition on each of those regions separately. Because YOLO is open-source, the community has been constantly enhancing the model. A subsequent version of YOLOv3 is YOLOv4. The three primary innovations in YOLOv4 are cross-mini-batch normalization, self-adversarial training, and mosaic data improvement. The typical inference timings, for YOLOv4, is 12ms while for YOLOv3 it is 29ms. The two-stage Mask R-CNN method has been greatly outperformed by the one-stage detector known as YOLO (one-stage detector) which has an inference time of 333ms [3].

In this work, we deployed YOLO-based Object detection models on an FPGA i.e. Kria KV260 for investigating its performance in terms of higher speed and fps. The remaining paper follows as follows. A survey of our approach to object detection using machine learning algorithms and its implementation on FPGA can be found in Section II. The approach for object detection is covered in Section III, the results based on various YOLO frameworks are covered in Section IV, followed by Conclusion in Section V.

## II. Literature Survey

The fundamental objective of object detection is to locate instances of each object in digital images or real-world situations, separate them and analyze their necessary properties for in-the-moment predictions. The foundation of other crucial AI vision techniques like image classification, image retrieval, or object co-segmentation, which extract meaningful information from actual objects, is provided by object detection. These methods are being used by engineers and developers to create cutting-edge devices for human ease.

### *A. Object Detection using YOLO*

YOLO v4 offers a faster object detection inference speed for production systems. It is applied to parallel computations in optimization. A single GPU is used by YOLO v4's effective and powerful object detection algorithm to swiftly produce an accurate object detector. YOLOv4 operates twice as rapidly while performing as well as EfficientNet boosts YOLOv3's AP and fps by 10 and 12 %, respectively. YOLOv4 outperforms the fastest and most accurate detectors in terms of both speed and precision. Using a self-made novel dataset called SLS(Students Lab Safety), the YOLOv5 (YOLOv5l, YOLOv5m, YOLOv5n, YOLOv5s, and YOLOv5x) and YOLOv7 models were trained on SLS. The collection includes 481 photos with a resolution of 835 x 1000 and four types, including gloves, helmets, masks, and goggles. Based on the number of instances, the performance of the various YOLOv5 and YOLOv7 versions are evaluated using evaluation measures such as precision, F1 score, recall, and mAP (mean average precision). The experimental findings showed that all of the models performed admirably in identifying PPE(Personal Protective Equipment) in educational labs. For both small and large instances, the mAP for the YOLOv5n method was the highest at 77.40%, followed by the mAP of 75.30% for the YOLOv5m model [4].

### *B. Object Detection using YOLO on Hardware*

FPGAs are efficient for deploying machine learning models, given their ability of reconfigurability, power efficiency, and throughput [5]. One of the time-efficient mechanisms for detection is FPGA-based object detection [7]. Our experimentation is to deploy an ML model built on the YOLO framework on one of the recent Xilinx FPGAs and analyze the outcomes. We have deployed the YOLO algorithm on Xilinx's Kria KV260 which consists of a Kria system-on-module (SOM), a carrier card, and a cooling fan. The SOM has the capability of AI/ML applications and is very appropriate for real-time applications as well as helps in delivering complete applications with low latencies and low power consumption [6].

In general FPGA-based hardware, accelerators offer the advantage of power efficiency and low latency as compared to GPU-based hardware accelerators. One of the main reasons for the selection of Kria KV 260 is its python-based PYNQ framework which supports a large number of custom overlays [8]. In [9] YOLO v3 has been deployed on DE10-Nano for Object Detection obtaining an fps of 28-33 and an mAP of 29. The authors of [10] deployed three models namely YOLO, Single Shot Detector (SSD), and Faster Region CNN (FRCNN) on Xilinx Pynq Z2 and Movidius NCS. In [11], YOLO v3-Tiny was deployed on Xilinx's Zynq 7020 for Facial detection with 1.88 fps and 30.9 mAP. YOLO v4-tiny and v4 were deployed on Jetson Xavier NX for mobile phone and Human detection which achieved an fps of 27.4 & 15 respectively in [12] and [13]. In [13], YOLO v4 was deployed on Kria KV260 for Human detection. Also, YOLO v5 was deployed on Xilinx Ultrascale for real-time object detection [14]. YOLO v6-nano & v6 was deployed on Tesla's GPU for Object detection [15] with an fps of 520 and mAP of 43.1. In [16], YOLO v7 obtained an fps of 161 and an mAP of 69.7. We have deployed YOLO v3 & v5 on KV260 to improve fps. These are the details of the different models on different hardware. Table I presents the summary of the survey of implementation of different object detection models on various hardware.

For all the papers referred to above, very limited work has been done on the proposed FPGA, so this motivated us to work on the Kria KV 260. Object Detection by using YOLO v3 and v5 models on Kria KV260 has not been reported

TABLE I
LITERATURE SURVEY OF OBJECT DETECTION ON HARDWARE

| Object Detection Models | | | | | |
|---|---|---|---|---|---|
| Reference Models | | Hardware | Target | fps | mAP |
| [9] | YOLOv3 | DE10-Nano | Object | 28-33 | 29.1 |
| [10] | YOLO, SSD, FRCNN | Pynq Z2 | Person | 0.84, 1.85, 0.75 | - |
| [11] | YOLOv3-Tiny | Zynq 7020 | Face | 1.88 | 30.9 |
| [12] | YOLOv4-Tiny | Jetson Xavier NX | Mobile Phhone | 27.4 | 62.57 |
| [13] | YOLO v4 | Jetson Xavier NX | Human | 15 | 85.9 |
| [13] | YOLO v4 | Kria KV260 | Human | - | 82.9 |
| [14] | YOLO v5 | Xilinx Ultra-scale | Real-Time Object | - | - |
| [15] | YOLOv6-nano | Tesla GPU | Object | 1242 | 36 |
| [15] | YOLO v6-s | Tesla GPU | Object | 520 | 43.1 |
| [16] | YOLO v7 | - | Real-Time Object | 161 | 69.7 |

before.

## III. METHODOLOGY

In this paper, we have deployed Object Detection models on FPGA. The model we have used is YOLOv3 and YOLOv5 and is deployed on Kria KV260, by deploying this model we have tried to compare the fps of both models.

As shown in Fig 1, which presents the block diagram, initially we trained the model by providing the coco dataset of 80 classes as input to the YOLO Model architecture and tuned some hyper-parameters so that we can get the best accuracy and minimum loss. After satisfying accuracy we, compiled the model as per the hardware requirement and then deployed it on the FPGA hardware Kria KV260 and ran the inference on it.

The dataset and the object detection models that we mentioned in Fig. 1 are explained briefly in the following subsection

### A. Dataset and Object Detection Models

- Dataset Details
  YOLO v3 & YOLO v5 the popular object detection algorithm are trained on the COCO (Common Objects in Context) [17] dataset, which is a large-scale object detection, segmentation, and captioning dataset. The COCO dataset includes 330,000 images and 80 object categories, with each image containing several object annotations. The dataset is appropriate for training object detection models since the objects are diverse and vary in size, orientation, and shape. Additionally, to strengthen the robustness of the model, it employs a method known as data augmentation to fictitiously expand the dataset.
- YOLO v3
  It uses the technique of image separation into grids. This grid has a definite number of boundary boxes around the objects which matches the predefined object classes. One boundary box predicts one object and gives a score of how accurate the prediction can be. YOLOv3 is better as compared to YOLOv2 in terms of speed, accuracy, and specificity of classes. Here we have deployed YOLOv3 on Kria KV260. In YOLOv3 the backbone used is Darknet 53, which means it has 53 convolution layers Model architecture consist of a stride size of 2, kernel size of 3x3, residual network (skip connections), and is trained on pixel size is 256 [18] .
- YOLO v5
  It is a compound-scaled object detection model based on the coco dataset. YOLO v5 comes in four versions: small (s), medium (m), large (l), and extra large (e), each providing better accuracy rates. The main two procedures of v5 are Data Augmentation and Loss Calculations. The main benefit of v5 is that it trains extremely quickly and is optimized compared to other YOLO versions. We used the YOLO v5s model and deployed it on Kria KV260. The model uses Darknet 53 as its backbone, it consists of 53 Convolution Networks, Pooling Layers, and Softmax as a part of its architecture. YOLOv5s has a pixel size of 640, and 7.2M parameters [19].

Thus we have deployed both YOLO v3 and v5 models on FPGA in order to obtain considerable fps.

### B. FPGA Based Detection

We have decided to go for FPGA as compared to GPU because of the low power consumption and its better speed. FPGA gives a better sense of parallelism for the execution of all the concurrent operations providing a better speed [20]. AMD Xilinx's Kria KV260 [21] is a Vision AI Starter Kit to provide designers/ researchers to evaluate their target applications like Machine Vision, Security Camera, and other industrial applications. This System on Module (SoM) consists of an evaluation carrier card for vision and machine learning models. It provides a Linux or Yocto-based environment for AI-enabled applications. The best advantage of deploying the machine learning-based models on FPGA is to improve the inference time and expect better fps. By looking at the possibilities and advantages of this SoM we decided to deploy our object detection models, both YOLO v3 and v5 on the Kria KV260 FPGA board.

## IV. RESULTS

In this paper, we have shown that Kria KV260 can efficiently handle object detection algorithms quite efficiently and with better fps.

### A. Training results

The model was trained on the coco dataset, which was further divided into training, validation, and testing dataset and the model was trained for 150 epochs.

Fig. 2 shows different graphs plotted after the training and validation of the model. All the graphs are plotted with respect

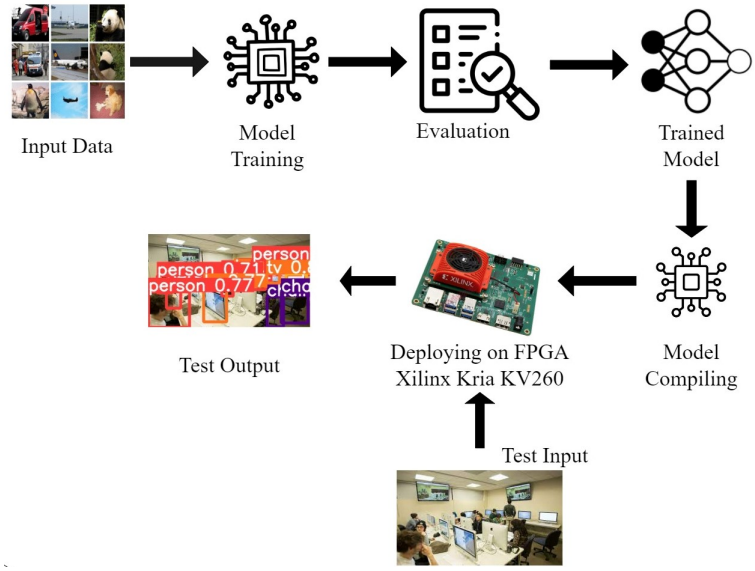


Fig. 1. Object Detection Block Diagram

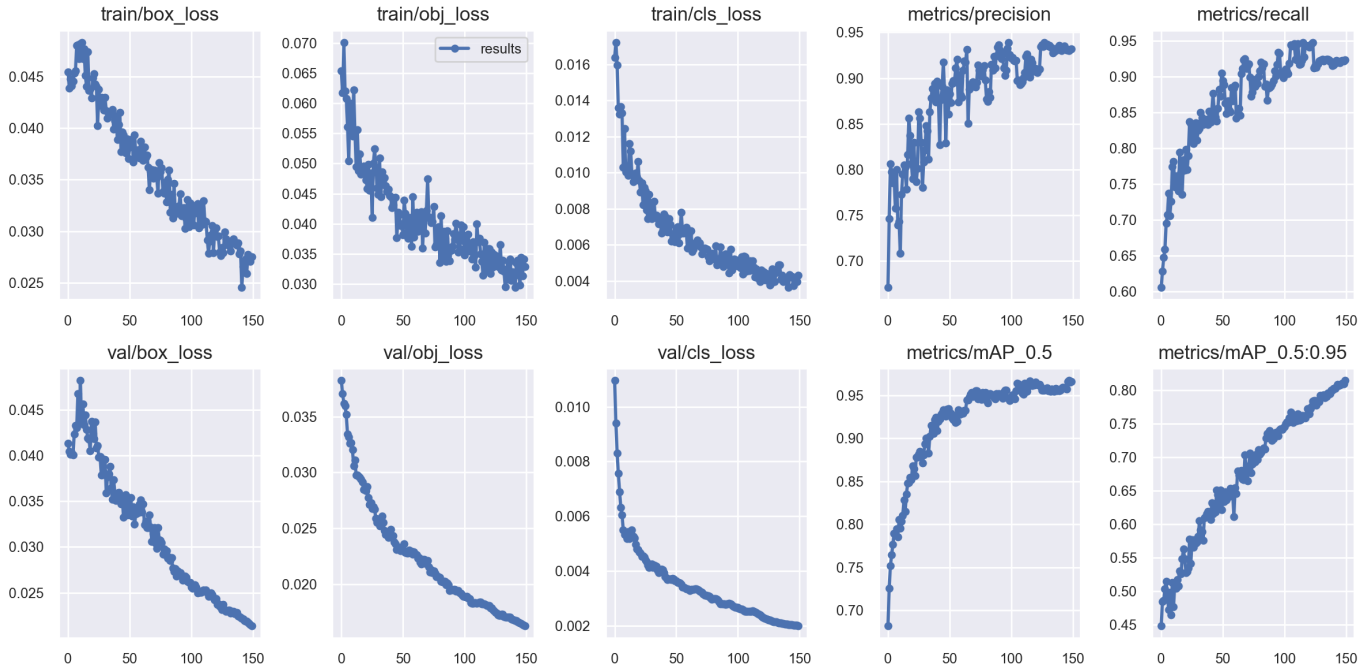


Fig. 2. Starting from Top Left; (a) epoch vs train/ box loss (box_loss), (b) epoch vs train/ object loss (obj_loss), (c) epoch vs train/ class loss (class_loss), (d) epoch vs precision, (e) epoch vs recall, (f) epoch vs valid/ box loss (box_loss), (g) epoch vs valid/ object loss (obj_loss), (h) epoch vs valid/ class loss (class_loss), (i) epoch vs mAP_0.5, (j) epoch vs mAP_0.5:0.95

to epoch as a common parameter. The graphs are plotted for 3 different losses i.e., bounding box loss, object loss, and class loss during the training of the model. After training, the validation graphs are plotted for those three losses as mentioned above. After this, the matrices graphs are plotted for Precision, Recall, mAP_0.5, and mAP_0.5:0.95.

### *B. Hardware results*

After deploying the model we ran the inference on the test dataset. The test data contained different images and videos to make the prediction.

*1) Object Detection Implementation on Hardware:* After achieving satisfactory results on the host computer, we deployed the Object Detection Models on the AMD Xilinx Kria KV260 hardware. However, deploying the CNN on an

FPGA technology like the KV260 required several steps, as the limited resources of the FPGA presented some constraints.

- Converting Model to FPGA-compatible format: Convert the object detection model to a format that is compatible with the FPGA board.
- Optimize the model for FPGA: Optimize the object detection model for FPGA to achieve the best performance. This involves reducing memory usage, pruning unnecessary layers, and reducing the precision of the weight.
- Design the hardware: Create a hardware design for the object detection model on the FPGA. This could involve designing the input/output interface, creating hardware blocks for the layers in the model, and integrating the optimized model into the FPGA board.
- Test the implementation: After completing the implementation, the final step was to test the model on the FPGA to verify its correct functioning. This required running the FPGA-converted model on a test dataset.


Fig. 3. Hardware Setup

In Fig. 3, the setup of the hardware is shown in which the hardware is connected to the screen with the help of an HDMI cable, an Ethernet wire is connected, and a mouse and a keyboard are also attached to the hardware. The hardware is given the power supply with the help of a 12 Volt AC/DC wall mount adapter and is booted with the help of an SD card.

TABLE II
HARDWARE RESULTS

| Object Detection Models on FPGA | | |
|---|---|---|
| Model | Hardware | fps |
| YOLOv3 | Kria KV 260 | 45 |
| YOLOv5 | Kria KV 260 | 30 |

The result of the inference run on FPGA is shown below in Fig. 4 - 7. We have also calculated the fps, which is briefed in Table II.

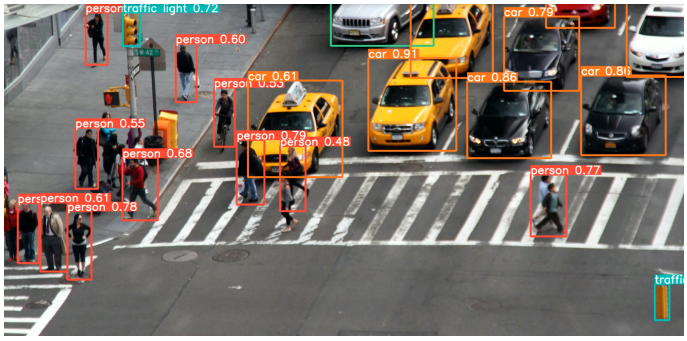


Fig. 4. Detecting Objects in Moving Traffic on a Video

A video of a crossroad with moving pedestrians was fed as input to check the performance of object detection with videos as data. A snapshot of the detection output on FPGA is presented in Fig. 4. It can be observed that the model is accurately detecting vehicles, traffic signals, and moving pedestrians at a higher fps.

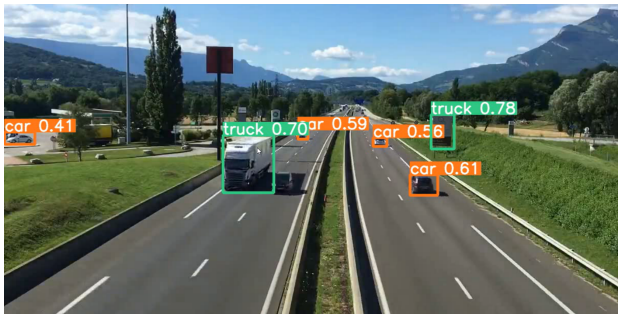


Fig. 5. Object Detection Results on Video

The video of Moving Vehicles on the highway was given as test input. In Fig. 5 the detection results of the model for predicting the moving vehicles are shown. Kria KV260 was able to process the data and detect the moving vehicles quite accurately.

In order to check the detection of the model on still images with multiple classes, a collage of boat images was given as test input. In Fig. 6 the prediction of the model detecting the Boats is shown. The collage of a Bird seating on a branch of a tree was taken for the next prediction. As shown in Fig. 7 the model is able to predict that the provided collage is of Birds with high accuracy.

It is evident from the results that even the model is efficient in detecting different objects in images, videos as well as in real-time video, giving a comparatively higher fps as compared to other implementations reported in the literature.

## V. Conclusion

Vision based real-time object detection is required in many applications. Instead of its software implementation, its hardware-based implementation improves the speed of the inference. This work presents implementation of Object

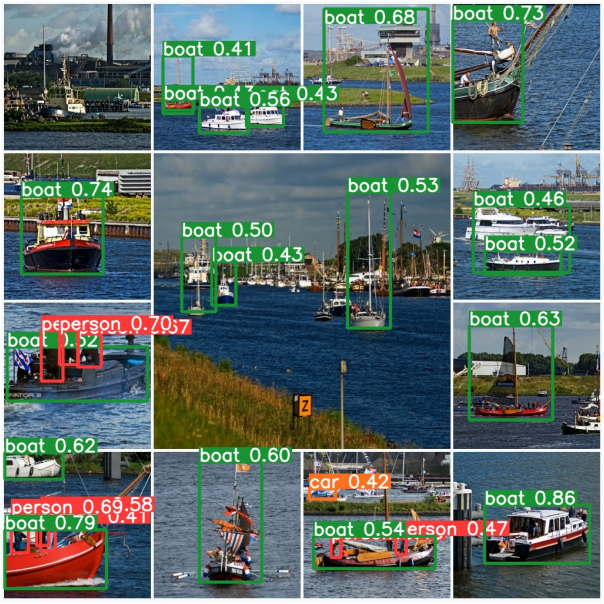


Fig. 6. Detection Results on Collage of Boats

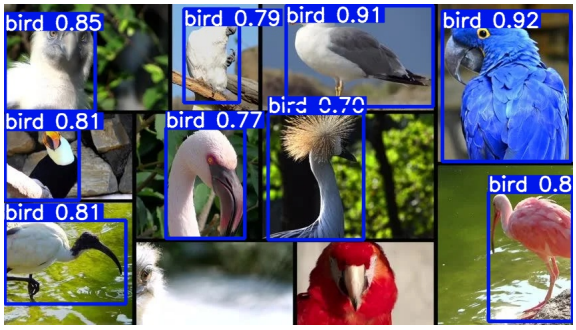


Fig. 7. Detection Results on Collage of Birds

Detection on FPGA. The object detector YOLOv3 & YOLOv5 are implemented on FPGA Kria KV260. The implementation achieved fps in the range of 30 to 45. From the experimental results it is evident that implementation of such high-performing Object Detection on FPGA has potential in many real-time applications.

## REFERENCES

[1] Du, Juan., "Understanding of object detection based on CNN family and YOLO " *I*n Journal of Physics: Conference Series, vol. 1004, no. 1, p. 012029. IOP Publishing, 2018.

[2] R. Gandhi, "R-CNN, fast R-CNN, faster R-CNN, YOLO — object detection algorithms" Towards Data Science, 09-Jul-2018. [Online]. Available: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-YOLO-object-detection-algorithms-36d53571365e. [Accessed: 05-Jan-2023].

[3] S. Wang, "textquotedblleft Research Towards YOLO-Series Algorithms: Comparison and Analysis of Object Detection Models for Real-Time UAV Applications" textquotedblright emph In Journal of Physics: Conference Series, vol. 1948, no. 1

[4] Solawetz, J. (2020, June 15). How to train A custom object detection model with YOLO v5. Towards Data Science. https://towardsdatascience.com/how-to-train-a-custom-object-detection-model-with-YOLO-v5-917e9ce13208

[5] Kolosov, Dimitrios, Vasilios Kelefouras, Pandelis Kourtessis, and Iosif Mporas of Deep Learning Image Classification and Object Detection on Commercial Edge Devices: A Case Study on Face Mask Detection" *I*EEE Access 10 (2022): 109167-109186.

[6] Vision, A. I."Kria k26 som: The ideal platform for vision ai at the edge" *(*2021).

[7] Seng, Kah Phooi, Paik Jen Lee, and Li Minn Ang. "Embedded intelligence on FPGA: Survey, applications and challenges " *E*lectronics 10, no. 8 (2021): 895.

[8] Kalapothas, Stavros, Georgios Flamis, and Paris Kitsos. "Efficient edge-AI application deployment for FPGAs" Information 13, no. 6 (2022): 279.

[9] Rzaev, Edward, Anton Khanaev, and Aleksandr Amerikanov. "Neural Network for Real-Time Object Detection on FPGA" *I*n 2021 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM), pp. 719-723. IEEE, 2021.

[10] Kaarmukilan, S. P., and Soumyajit Poddar. "FPGA based deep learning models for object detection and recognition comparison of object detection comparison of object detection models using FPGA" *I*n 2020 Fourth international conference on computing methodologies and communication (ICCMC), pp. 471-474. IEEE, 2020.

[11] Günay, Bestami, Sefa Burak Okcu, and Hasan Şakir Bilge. "LPYOLO: Low Precision YOLO for Face Detection on FPGA" *a*rXiv preprint arXiv:2207.10482 (2022).

[12] Wang, Guanbo, Hongwei Ding, Bo Li, Rencan Nie, and Yifan Zhao. "Trident-YOLO: Improving the precision and speed of mobile device object detection" *I*ET Image Processing 16, no. 1 (2022): 145-157.

[13] Rizk, Mostafa, Dominique Heller, Ronan Douguet, Amer Baghdadi, and J-Ph Diguet. "Optimization of deep-learning detection of humans in marine environment on edge devices" *I*n 2022 29th IEEE International Conference on Electronics, Circuits and Systems (ICECS), pp. 1-4. IEEE, 2022.

[14] Liu, Mingshuo, Shiyi Luo, Kevin Han, Bo Yuan, Ronald F. DeMara, and Yu Bai. "An Efficient Real-Time Object Detection Framework on Resource-Constricted Hardware Devices via Software and Hardware Co-design" *I*n 2021 IEEE 32nd International Conference on Application-specific Systems, Architectures and Processors (ASAP), pp. 77-84. IEEE, 2021.

[15] Barazida, Nir. "YOLOv6: Next-generation Object Detection — Review and Comparison" Medium, 5 July 2022, towardsdatascience.com/YOLOv6-next-generation-object-detection-review-and-comparison-c02e515dc45f.

[16] Wang, Chien-Yao, Alexey Bochkovskiy, and Hong-Yuan Mark Liao., "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors" *a*rXiv preprint arXiv:2207.02696 (2022).

[17] "COCO - common objects in context" Cocodataset.org. [Online]. Available: https://cocodataset.org/. [Accessed: 14-Mar-2023].

[18] V. Meel, "YOLOv3: Real-time object detection algorithm (guide)" viso.ai, 25-Jun-2022. [Online]. Available: https://viso.ai/deep-learning/YOLOv3-overview/. [Accessed: 05-Jan-2023].

[19] J. Solawetz, "How to train A custom object detection model with YOLO v5," Towards Data Science, 15-Jun-2020. [Online]. Available: https://towardsdatascience.com/how-to-train-a-custom-object-detection-model-with-YOLO-v5-917e9ce13208. [Accessed: 05-Jan-2023].

[20] Magalhães, Sandro Costa, Filipe Neves dos Santos, Pedro Machado, António Paulo Moreira, and Jorge Dias. "Benchmarking edge computing devices for grape bunches and trunks detection using accelerated object detection single shot multibox deep learning models" Engineering Applications of Artificial Intelligence 117 (2023): 105604.

[21] "System-on-modules (SOM) boards" Xilinx. [Online]. Available: https://www.xilinx.com/products/som/Kria/kv260-vision-starter-kit.html. [Accessed: 05-Jan-2023].