Vision Guided Crop Detection in Field Robots using FPGA-based Reconfigurable Computers

Cyrus Wing-Hei Chan
University of Hong Kong
Email: u3527884@connect.hku.hk

Philip H.W. Leong
University of Sydney
Email: philip.leong@sydney.edu.au

Hayden Kwok-Hay So
University of Hong Kong
Email: hso@eee.hku.hk

***Abstract*—A case study in applying modern FPGAs as a platform to accelerate intelligent vision-guided crop detection in agricultural field robots is presented. A state-of-the-art YOLOv3 object detection neural network was adapted to detect broccoli and cauliflower in image dataset obtained from autonomous agricultural robots. A baseline floating point implementation achieved 96 % mAP, and an efficient, quantized implementation suitable for FPGA implementation achieved 92 % mAP. The proposed FPGA solution achieved 136.86 ms inference latency while consuming 12.43 W in a low latency setup, and 28.48 frames per second while consuming 17.78 W in a high throughput setup. Compared to an embedded GPU implementation of the same task, the FPGA solution is 4.12 times more power-efficient and offers 6.85 times higher throughput, translating to faster and longer operation of a battery-powered field robot.**

I. Introduction

Machine learning approach based on imaging techniques have been widely adopted in modern agriculture for tasks ranging from crop detection to soil management [1]. Many of these data analytic tasks are computationally demanding. In applications where data can be processed offline with no real-time constraints, such computational needs can be met with reasonably high performance computing systems, either on-site or in the cloud with acceleration using GPUs or FPGAs. However, in real-time and mobile applications, such as when used in autonomous field robots that must cover large areas between each recharge, in-situ, high-performance but energy-efficient embedded computing platforms that are also robust and easy-to-program can improve utility.

In this work, a case study of accelerating real-time vision-based crop detection with modern integrated FPGA for use in autonomous field robot is presented. The state-of-the-art YOLOv3 [2] neural network was modified and retrained to identify broccoli and cauliflower in raw images taken from a field robot that surveyed a growing field over a course of 10 weeks. Despite varying lighting conditions and growth stages of the crops, the retrained neural network was capable of detecting and classifying the 2 crops with high accuracy.

For deployment in autonomous field robots with real-time requirements, inference of the retrained neural network was accelerated with an FPGA MPSoC. While researchers have previously demonstrated a hand-optimized binarized YOLOv2 inference accelerated with FPGA [3], we chose instead to implement our neural network using the vendor-provided high-level compilation framework DNNDK [4]. The use of high-level compilation framework allows rapid prototyping and deployment of deep neural network on FPGAs, which is a necessary workflow for multidisciplinary research teams in precision agriculture where not all members can be FPGA hardware design expert.

We also implemented the same network on an embedded GPU platform, which similarly provided high-performance neural network inference with easy to program interfaces. Both platforms were fabricated with 16 nm technologies and were both tightly integrated to a host ARM processor. A comparison of their throughput, power consumption, energy efficiency, and development effort is presented.

II. Related Work

Computer vision has long been utilized in industrialized agriculture for tasks ranging from navigation in the field [5], [6] to real-time crop detection [7] and weed detection. Recently, there is an increasing interest in leveraging modern machine learning algorithms to prepare these tasks to the next generation [1]. For instances, a custom convolutional neural network (CNN) was developed in [8] to classify carrot in field images after a custom segmentation preprocessing step. Using a custom encoder-decoder neural architecture, researchers have also demonstrated real-time performance in pixel-wise semantic segmentation for isolating weed from crops [9]. Similarly, a custom CNN was designed to classify weeds in segmented images for robot control in [10]. In all of the above cases, in order to maintain real-time performance, GPU has been employed as computer accelerators.

On the other hand, with the promise of superior power-efficiency, numerous researchers have turned to the use of FPGA to accelerate deep machine learning inference using FPGA both in large-scale datacenters and in high-performance embedded systems [11]. In the area of precision agriculture, Lammie *et al.* have recently performed an extensive comparison and demonstrated the benefit of accelerating a weed classification task using FPGAs for power-efficiency [12].

In this work, we similarly explore the use of FPGA for real-time crop detection in autonomous field robots. We focus particularly on high-performance embedded systems with a goal to improve power-efficiency of the system while maintaining real-time performance.

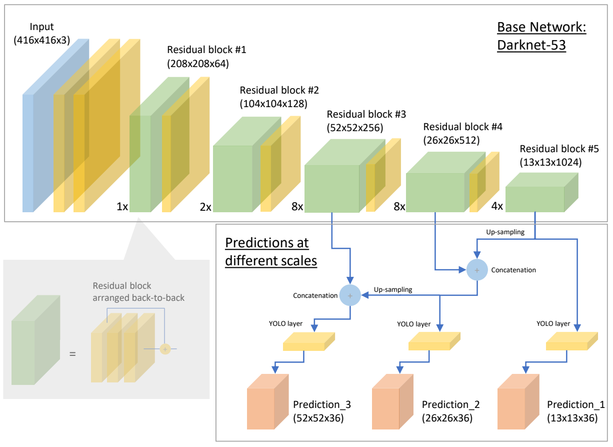


Fig. 1. Architecture of the adopted YOLOv3 network

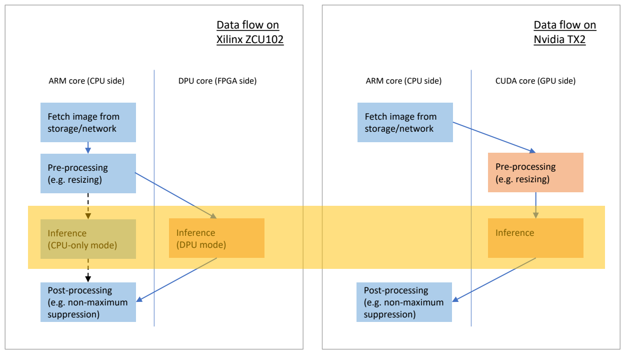


Fig. 2. Timings of the different stages, with inference highlighted.

III. System Design

Our system was designed to be deployed in field robots that autonomously patrol crop beds and perform necessary actions such as weed control. Specifically, we studied the task of identifying valuable crops from their bed using visible spectrum imaging in real-time on autonomous robots.

A. Dataset

For this study, the Ladybird Cobbitty 2017 Brassica Dataset was used [13], [14]. The dataset contains 1248 annotated images that were collected by the Ladybird autonomous agricultural robot while monitoring 4 crop beds over a period of 10 weeks. The dataset was annotated with bounding boxes on the 2 crops at various growth stages, as well as 5 color checkers and calibration panels, namely, “Spectralon 15 %”, “Spectralon 30 %”, “Spectralon 60 %”, “Xrite ColorChecker Classic” and “Xrite ColorChecker Grayscale”. These calibration panels were put in the field to allow calibration of the camera during different lighting condition. While the sensing elements on the Ladybird included a hyperspectral camera, a thermal camera, as well as a pair of high-resolution stereo camera, only the left image from the stereo camera were used in this work.

B. Network Architecture

Our crop detection neural network was based on the original full size YOLOv3 network [2]. The YOLO family of neural network is currently one of the fastest object detection network with high-accuracy real-time implementations. The YOLOv3 network calculates feature maps at multiple scales, and hence improves the detection of object with varying input sizes. The feature extractor in YOLOv3, termed “Darknet-53”, is made up of 23 $(1 + 2 + 8 + 8 + 4)$ residual blocks, using feature extractors composed of 1×1 and 3×3 convolution filters (see Figure 1), and with batch normalization and Leaky ReLU applied to each layer. The feature map is extracted at 3 scales, and pushed through the detection block to obtain the predictions. These prediction results are then combined and go through Non-Maximum Suppression (NMS) to produce the final predicted bounding boxes. To adapt to our object detection task, we shrink the number of detection classes from the original 20 down to 7, corresponding to the 2 crops and the 5 calibration panels. As shown in Figure 1, the output of the detection block (layer 81, 93 and 105) have the dimension of $N \times N \times 36$, containing 3 bounding box predictions, each with 1 objectness score, 4 parameters for bounding box coordinates and 7 class confidence.

C. System Configuration

Our hardware prototype was based on the Xilinx ZCU102 development board, featuring a Zynq UltraScale+ XCZU9EG-2FFVB1156 MPSoC [15]. The chip combined a quad-core ARM Cortex-A53 processor system (PS) with user programming logic (PL). The task of neural network inference was offloaded to the PL, while the image pre- and post-processing were performed on the ARM cores. Three (3) deep learning processing units (DPUs) from the vendor (Xilinx, V1.4.0) with the B4096 architecture running at 325 MHz were configured in the PL under the control of the PS (Figure 2).

The neural network inference accelerator was developed using the DNNDK framework [4] based on our modified YOLOv3 network. The original dataset was split into a training set with 739 images and a validation set with 509 images. Using the Darknet framework [16], the YOLOv3 network was re-trained using pre-trained weight on the PASCAL VOC dataset [17]. During the transfer learning phase, we initialized the feature extractor (top half in Figure 1) with the weights from the aforementioned model, and randomly initialized the classification layers (lower half in Figure 1), which were shrunk down to 36 filters. The input filter size of the YOLOv3 filter was kept at 416×416.

The trained network with floating-point (`fp32`) weights was then converted and quantized to 8-bit integers (`int8`) for FPGA implementation. It was accomplished using the min-diffs method with the DNNDK framework using only samples from the training set. The quantization only affected the truncated bits and did not retrain the weights of the other parts of the network.

IV. Experimental Results

A. Detection Quality

Our retrained network with floating point weights achieved 96 % mAP on the validation set using the default PASCAL VOC metrics [17]. The VOC2007 metrics specifies that the intersection over union (IoU) of the detection and the ground truth must be larger than 0.5 to be counted as a successful detection. After quantization of weights as 8-bit integers (`int8`) weights, the accuracy dropped to about 92 % mAP with NMS threshold set to 0.5.

As shown in Figure 3, the network was able to detect crops with different sizes at different growth stages over the span of 10 weeks in the dataset. High accuracy was achieved even for cases where the source images were underexposed. Common detection errors (Figure 4) included duplicates and missing detection, especially when calibration panels were involved. We theorize that this may be caused by the small number of panels originally presented in the training dataset.

B. Performance & Power

Table I summarizes the performance and power consumption of our FPGA implementation. Latency, throughput, and power measurements are defined as follow:

Latency was obtained by measuring the end-to-end delay of processing 1 image by the 2 stages of our *processing pipeline*. The first stage of this processing pipeline was responsible for pre-processing the input images, which included resizing and quantizing the input images from 848×565 to 416×416 for inference. In the second stage, inference and post-processing of the results, including NMS computation were performed. Processing latency for each of the image in the dataset were measured and the average value is reported here. Throughput was obtained by dividing the size of the image dataset (1248 images) by the total time required to perform inference on the dataset. Power consumption values were obtained from the on board INA226 power IC. We monitored the current drawn from the power rails to obtain the average power over a 10 s interval.

Two different hardware/software setups were experimented. In the low-latency setup, a single processing pipeline was used in the host software. In the high-throughtput setup, 3 processing pipelines executing as parallel threads on the host CPU were used. As controlled by the runtime environment, 1 DPU was activated in the low-latency setup and 3 DPUs were activated in the high-throughput setup.

The results in Table I indicate that the lowest processing latency was achieved when a single processing pipeline was used. Highest throughput was achieved when 3 processing pipelines were used, which caused all 3 available DPUs to be activated. However, the increased I/O contention, as indicated by underutilized DPUs, also resulted in increased per-image processing latency.

C. Comparison with Embedded CPU & GPU

In this section we compare the performance of the network against an embedded GPU platform and the embedded ARM CPU on the FPGA. The Jetson TX2 is an embedded GPU platform that integrates a quad-core ARM Cortex-A57 CPU with a 256-core NVIDIA Pascal GPU fabricated using TSMC 16 nm technology [18]. We ran the customized floating-point YOLOv3 network using the Jetson TX2 GPU and the Darknet framework [16] with CUDA and cuDNN support. For the embedded CPU only case, we compiled Darknet with OpenMP support. In all cases we set the batch size to 1.

As shown in Table I, while the the floating point inference on GPU results in the best accuracy, the quantized implementations on FPGA result in the best inference latency and throughput. The FPGA setup achieves the highest power efficiency, reaching 1.6 fps/W or 0.624 J/frame, which is 6.85 times higher than the embedded GPU solution. On the other hand, the embedded GPU has the lowest power consumption, both when idle and during inference.

We also tested with larger batch sizes on the Jetson TX2, and only observed a minimal throughput increase. At a batch size of 3, the setup achieved a latency of 0.65 s, throughput of 4.92 fps and drew 12.78 W of power.

V. Conclusion and Future Work

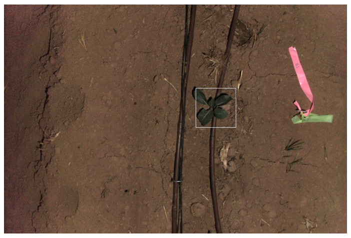
This work presents the feasibility to use FPGA-based machine learning solution for object detection in precision agriculture. After quantization, the network experience a sight accuracy drop but offers performance comparable to using GPU accelerated solution. The low and consistent latency also benefits real-time robot control. Our FPGA implementation achieved 6.8 times better power-efficiency than our embedded GPU solution. This translates to longer operation time on field robots.

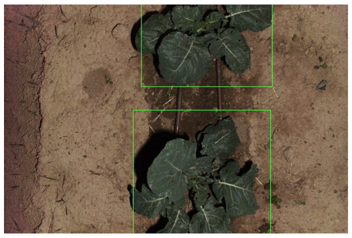
The FPGA-based solution is more suitable for use in harsh conditions in farmlands, where extreme temperature and humidity may arise throughout the growing season. Its high power-efficiency allows intelligent data processing at the edge, which significantly improves response time and reliability of the field robot in harsh environment with unreliable network coverage. The Linux OS running on the ARM processor cores enables easy integration with existing robot parts.

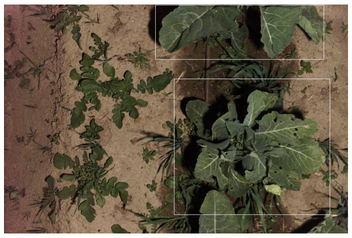
Timing analysis of our current implementation suggests that software resizing of larger images will become the system bottleneck. Therefore in the future, we plan to integrate image preprocessing steps with our hardware inference engine for acceleration. We also plan to explore the accuracy and speed trade-offs using different object detection networks, and will explore dropout and data augmentation to reduce overfitting.

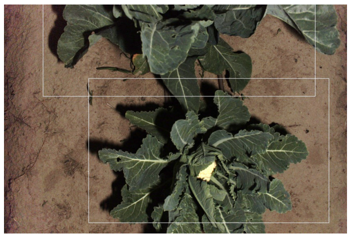
Acknowledgment

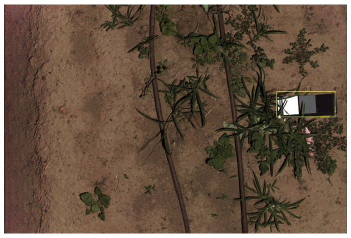
We would like to thank Dr. Asher Bender and Prof. Salah Sukkarieh of Australian Centre for Field Robotics for the comprehensive Ladybird Cobbitty 2017 Brassica Dataset. This work was supported in part by the HKU-USydney Strategic Partnership Fund and by the HKU Undergraduate Research Fellowship Programme.


(a) Cauliflower bed, week 2


(b) Broccoli bed, week 5


(c) Cauliflower bed, week 6


(d) Cauliflower bed, week 8


(e) Weed with calibration panel

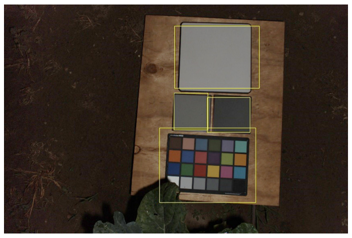
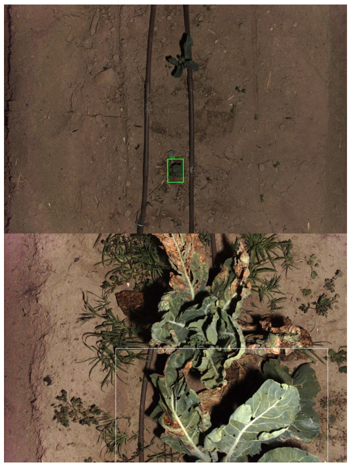
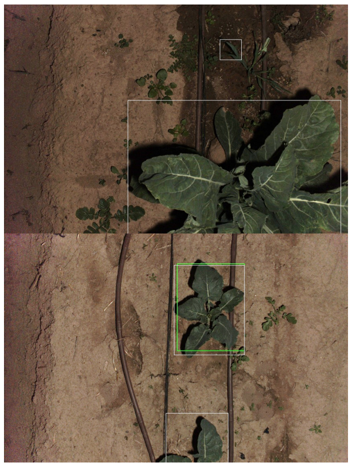

(f) Full calibration panel set

Fig. 3. Examples of successful detection with detection bounding boxes overlay on the original image. Images are brightened for presentation clarity.


(a) Missing detection


(b) Incorrect detection

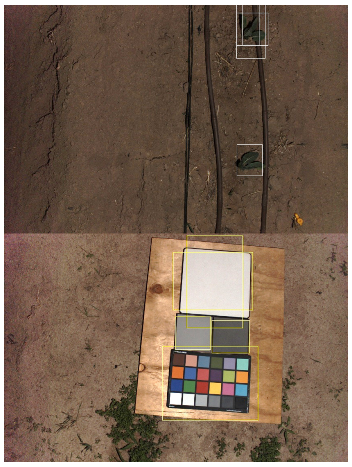

(c) Duplicated detection

Fig. 4. Examples of failed detection. Images are brightened for presentation clarity.

TABLE I
PERFORMANCE COMPARISON BETWEEN FPGA, EMBEDDED CPU, AND EMBEDDED GPU IMPLEMENTATIONS. IN THE LOW LATENCY FPGA SETUP, 1 PROCESSING PIPELINE WAS USED. IN THE HIGH THROUGHPUT FPGA SETUP, 3 PRODESSING PIPELINES WERE USED.

	Unit	**FPGA (1 pipeline)**	**FPGA (3 pipelines)**	**CPU**	**GPU**
Latency (Total)	ms	136.86	160.68	27.09×10^{3}	274.53
▷ preprocess	ms	68.45	74.90	220.49	36.35
▷ inference	ms	68.41	85.78	27.08×10^{3}	238.18
Throughput	fps	14.48	28.48	36.92×10^{-3}	4.16
Power (Idle)	W	6.19	6.19	6.19	2.12
Power (Load)	W	12.43	17.78	7.71	10.71
Power Efficiency	fps/W	1.16	1.60	4.79×10^{-3}	388.42×10^{-3}

References

[1] K. G. Liakos, P. Busato *et al.*, "Machine learning in agriculture: A review," *Sensors*, vol. 18, no. 8, 2018.

[2] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018.

[3] H. Nakahara, H. Yonekawa *et al.*, "A lightweight YOLOv2: A binarized CNN with a parallel support vector regression for an FPGA," in *Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '18. New York, NY, USA: ACM, 2018, pp. 31–40.

[4] *DNNDK User Guide*, UG1327 (v1.6), Xilinx, Aug 2019.

[5] V. Dworak, M. Huebner, and J. Selbeck, "Precise navigation of small agricultural robots in sensitive areas with a smart plant camera," *Journal of Imaging*, vol. 1, no. 1, pp. 115–133, 2015.

[6] T. Bak and H. Jakobsen, "Agricultural robotic platform with four wheel steering for weed detection," *Biosystems Engineering*, vol. 87, no. 2, pp. 125 – 136, 2004.

[7] U. Weiss and P. Biber, "Plant detection and mapping for agricultural robots using a 3D LIDAR sensor," *Robotics and Autonomous Systems*, vol. 59, no. 5, pp. 265 – 273, 2011, special Issue ECMR 2009.

[8] F. J. Knoll, V. Czymmek *et al.*, "Real-time classification of weeds in organic carrot production using deep learning algorithms," *Computers and Electronics in Agriculture*, p. 105097, 2019.

[9] A. Milioto, P. Lottes, and C. Stachniss, "Real-time semantic segmentation of crop and weed for precision agriculture robots leveraging background knowledge in CNNs," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 2229–2235.

[10] A. dos Santos Ferreira, D. M. Freitas *et al.*, "Weed detection in soybean crops using ConvNets," *Computers and Electronics in Agriculture*, vol. 143, pp. 314 – 324, 2017.

[11] M. P. Véstias, "A survey of convolutional neural networks on edge with reconfigurable computing," *Algorithms*, vol. 12, no. 8, 2019.

[12] C. Lammie, A. Olsen *et al.*, "Low-power and high-speed deep FPGA inference engines for weed classification at the edge," *IEEE Access*, vol. 7, pp. 51 171–51 184, 2019.

[13] A. Bender, B. Whelan, and S. Sukkarieh, "A high-resolution, multimodal data set for agricultural robotics: A Ladybird's-eye view of Brassica," *Journal of Field Robotics*, vol. 37, no. 1, pp. 73–96, 2020.

[14] ——, "Ladybird Cobbitty 2017 Brassica dataset," https://doi.org/10.25910/5c941d0c8bccb, The University of Sydney, Mar. 2019.

[15] *Zynq UltraScale MPSoC Data Sheet: Overview*, DS891 (v1.8), Xilinx, Oct 2019.

[16] J. Redmon, "Darknet: Open source neural networks in C," http://pjreddie.com/darknet/, 2013–2016.

[17] M. Everingham, L. Van Gool *et al.*, "The pascal visual object classes (VOC) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun 2010.

[18] D. Franklin. (2017, Mar) NVIDIA Jetson TX2 delivers twice the intelligence to the edge. https://devblogs.nvidia.com/jetson-tx2-delivers-twice-intelligence-edge/.