applied sciences

MDPI

Article

An Intelligent Real-Time Object Detection System on Drones

Chao Chen [1], Hongrui Min [1,2], Yi Peng [1,2], Yongkui Yang [1] and Zheng Wang [1,*]

[1] Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China
[2] University of Chinese Academy of Sciences, Beijing 101408, China
* Correspondence: zheng.wang@siat.ac.cn

Abstract: Drones have been widely used in everyday life and they can help deal with various tasks, including photography, searching, and surveillance. Nonetheless, it is difficult for drones to perform customized online real-time object detection. In this study, we propose an intelligent real-time object detection system for drones. It is composed of an FPGA and a drone. A neural-network (NN) engine is designed on the FPGA for NN model acceleration. The FPGA receives activation data from an NN model, which are assembled into the data stream. Multiple fetch and jump pointers catch required activation values from the data stream, which are then filtered and sent to each thread independently. To accelerate processing speed, multiple processing elements (PEs) deal with tasks in parallel by using multiple weights and threads. The image data are transferred from the drone host to the FPGA, which are tackled with high speed by the NN engine. The NN engine results are returned to the host, which is used to adjust the flying route accordingly. Experimental results reveal that our proposed FPGA design well utilizes FPGA computing resources with 81.56% DSP and 72.80% LUT utilization rates, respectively. By using the Yolov3-tiny model for fast object detection, our system can detect objects at the speed of 8 frames per second and achieves a much lower power consumption compared to state-of-the-art methods. More importantly, the intelligent object detection techniques provide more pixels for the target of interest and they can increase the detection confidence score from 0.74 to 0.90 and from 0.70 to 0.84 for persons and cars, respectively.

Keywords: neural network accelerator; FPGA; object detection; intelligent system; machine learning

Citation: Chen, C.; Min, H.; Peng, Y.; Yang, Y.; Wang, Z. An Intelligent Real-Time Object Detection System on Drones. *Appl. Sci.* **2022**, *12*, 10227. https://doi.org/10.3390/app122010227

Academic Editors: Yujin Lim and Hideyuki Takahashi

Received: 19 August 2022
Accepted: 3 October 2022
Published: 11 October 2022

1. Introduction

Drone technology has undergone rapid development in recent decades, which has resulted in a dramatic cost reduction for commercial drones, especially quadcopter drones. As a result, more drones are used in everyday life for different kinds of jobs, ranging from search and rescue to photography [1–4]. In addition, by equipping various sensors, drones are able to provide information with different features, which makes it feasible to work as an intelligent system by merging outside information.

Traditionally, drones are controlled and operated by drone pilots to achieve specified goals, such as searching certain targets, etc. Drone pilots play several roles throughout the flying process, including analyzing data in real time, and avoiding obstacles for safety requirements. However, several problems need to be addressed for drone pilots. On the one hand, drone pilots cannot continuously work without a rest and it is necessary to have a rest after working for a long time. On the other hand, even if drone pilots can help analyze data such as images, it is difficult for them to perform high-speed image processing, and some small targets may even not be spotted. Therefore, it is necessary to design an intelligent system for drones to address these problems.

Various components and devices have been used to build intelligent systems, where the Field Programmable Gate Array (FPGA) is a well-known acceleration device for artificial intelligence (AI) tasks [5–8]. In this research, we propose an intelligent real-time object detection system on drones. We first design an FPGA accelerator for real-time object classification and detection. The drone sends image data to the FPGA via the host, which are

assembled into the data stream for access by multiple fetch and jump pointers. Activation values are fetched for each thread and then filtered. Processing elements tackle tasks in parallel by using multiple weights and threads, which significantly speeds up the running of deep-learning neural networks (NNs).

Then, we build an intelligent system using the FPGA on drones, such that the drone can flexibly adjust flying routes to generate more pixels for objects which helps detect objects more accurately based on FPGA results. Experimental results reveal that our proposed FPGA design well utilizes FPGA computing resources, with 81.56% DSP and 72.80% LUT utilization rates, respectively. By using the Yolov3-tiny model for fast object detection, our system can detect objects at the speed of 8 frames per second (FPS) and achieve a much lower power consumption compared to state-of-the-art methods. More importantly, the intelligent object detection techniques provide more pixels for the target of interest and it can increase the detection confidence score from 0.74 to 0.90 and from 0.70 to 0.84 for persons and cars, respectively.

The rest of the paper is organized as follows. Section 2 introduces related work, materials and methods are demonstrated in Section 3, results and discussion are presented in Section 4 and Section 5 concludes the paper.

2. Related Work

Researchers have proposed plenty of object detection algorithms, especially with the help of AI techniques. AI algorithms have been proven to be effective in object detection in terms of detection accuracy and speed. In this section, we review existing related work with AI techniques in different aspects, i.e., FPGA acceleration, AI object detection, and intelligent systems.

2.1. FPGA Acceleration

FPGAs are widely used for AI task acceleration. We find three studies that are closely related to our proposed design and they are summarized in Table 1.

Table 1. Summary of state-of-the-art object detection using FPGA accelerators.

	Virtex-Yolo [9]	**Zynq-Yolo** [10]	**Req-Yolo** [11]
Advantages	High speed	Multiple NN support	High speed
Disadvantages	Yolov3 not supported	Yolov3 not supported	Yolov3 not supported
	Few operator support	Few operator support	Few operator support
	No post-proc.	No post-proc.	No post-proc.
	High power	High power	High power
		Low speed	Compressed Yolov2

Ma et al. [9] propose an FPGA accelerator for Yolov2-tiny, where the Xilinx Virtex-7 485T FPGA is used for development. They support operators of convolution, pooling, ReLU, and fully connected layers. The input size is 448×448 and the system generates $7 \times 7 \times 2$ bounding boxes. The essential contribution is that they optimize the hardware implementation of the fully connected layer and the detection speed is fast. It is worth mentioning that in the new Yolov3, the fully connected layer is not used.

Guo et al. [10] develop an FPGA accelerator using Xilinx Zynq 7020 FPGA board. This design can be applied to various neural networks and it supports Yolov2, whose operators are the same as those in [9]. Its detection speed is low compared to the work in [9].

Ding et al. [11] achieve high-speed detection using a Xilinx Virtex-7 690T FPGA. They accelerate the convolution by utilizing FFT. The operators supported include convolution, max pool, leaky ReLU, etc. The main contributions are that they implement the fully-

connected layer using convolution and compress the model weight to 4.95 MB. This is much smaller than the original and it comes with a cost of accuracy

In addition, the related studies all accelerate Yolov2—which consists of nine convolutional layers—and do not support Yolov3. The operators in Yolov2 are limited. Our design can accelerate Yolov3, which includes 13 convolutional layers and new operators such as up-sampling and concatenation. The previous studies generate bounding boxes which are then processed by CPUs and our design includes a post-processing module in the FPGA, which greatly accelerates the post-processing. Since the previous studies all include SoCs that run an operating system, the power consumption of these studies is inevitably high and they are not suitable for lower power-consumption scenarios.

2.2. Object Detection

Object detection aims to locate and classify objects and we introduce some classic object detection algorithms in this section. With the development of deep-learning technology, the performance of neural networks in object detection far exceeds traditional algorithms, and the state-of-the-art models on public datasets such as MS-COCO [12] are all based on neural networks. The object detectors of the current mainstream object detection algorithms can be divided into two-stage object detectors and single-stage object detectors. The R-CNN family, such as Fast R-CNN [13], employs two-stage object detectors. Yolo [14] and Swin transformer [15] use a single-stage object detector. These models can achieve quite a high accuracy and real-time inference speed in object detection, but the number of parameters is large and the training time is long. Due to limited resources in some scenarios, we cannot deploy these models on edge devices, so there are many lightweight models such as MobileNet [16].

R-CNN [17] is the first application of convolutional neural network (CNN) in the field of object detection, where CNN is mainly used for feature extraction. R-CNN uses the selective search method to select 2000 object candidate regions, then uses CNN to extract a 4096-dimension feature vector from each region and constructs SVM classifiers to classify the feature vectors, and, finally, uses non-maximum suppression (NMS) to adjust the position of the candidate regions.

Compared with R-CNN, Fast R-CNN [13] uses a convolutional neural network as the output to classify and predict the position size of the box, replacing the original SVM classifier. In order to improve the calculation speed, the network finally uses SVD instead of the fully connected layer, so the training speed of the latter is faster than the former.

Yolo [14] treats detection problems as a regression task to directly predict pixels as targets instead of a classification task. Firstly, Yolo divides the input image into SXS grid cells. Each grid cell will detect whether the center of the object falls into it, and if it falls, the grid cell is responsible for detecting the object. Then we can obtain multiple bounding boxes (BBoxes) and the confidence score of the boxes from each cell. If there is no target, the confidence score is zero.

Backbone architectures are an important component in the object detector to extract the input image features. The ResNet [18] proposed by Kaiming He et al. is one of the widely used backbones in object detection. ResNet avoids the vanishing gradient phenomenon through the cross-layer connection of the neural network, and this connection method does not bring additional computational complexity. There are two basic blocks in ResNet-50, Conv Block and Identity Block. The input dimension and output dimension of Identity Block are the same, which are used to deepen the network depth. There are 16 bottleneck modules and a 7X7 convolutional kernel, each bottleneck module has three layers of convolution. The ResNet-50 network contains 49 convolutional layers, and a fully connected layer.

2.3. Intelligent Systems

With the rapid development of technology, researchers have investigated various methods to build intelligent systems and we introduce some typical intelligent systems in this section. Many sensors are equipped in various systems to help gather surrounding information that can be used to make intelligent decisions. Tactile sensors are used in advanced intelligent systems–such as robotics and human-machine interfaces–to achieve safe and precise operations in uncertain environments [19]. Vehicle visual sensors are adopted by Li et al. to perform real-time semantic understanding and segmentation of urban scenes [20]. Qiu et al. [21] build intelligent human motion capture systems using wearable sensors. Haseeb et al. [22] propose a sensor-cloud architecture for intelligent Internet of Things to provide data security and network scalability.

In addition, machine learning (ML) is becoming increasingly important in intelligent systems [23,24]. A large amount of data have been generated from equipped sensors and the Internet, which offers routes to training systems becoming more intelligent using these data together with ML. He et al. [25] summarize sensing applications and investigate how machine-learning algorithms are used to develop intelligent sensor systems. The standard extreme learning machine achieves the best diagnosis result with an accuracy of 80%. Prencipe et al. [26] propose a system that utilizes a 2.5D convolutional neural network (CNN) with the multi-class focal Dice loss, such that liver segments can be classified for surgical planning. An intelligent system is developed to manage and control plant leaf diseases using feature fusion and PCA-LDA classification [27].

Haq et al. [28] propose an intelligent system that takes advantage of decision trees to classify healthy and diabetic subjects using clinical data. González et al. [29] build an intelligent system to solve a closed-loop supply-chain problem using fuzzy logic with machine learning. Deepa et al. [30] perform healthcare analysis with an AI-based intelligent system using a Ridge-Adaline Stochastic Gradient Descent (RASGDC) classifier. Zekić et al. [31] utilize deep neural network (DNN) and tree-partitioning methods to predict energy consumption in smart cities such that an intelligent system is developed for efficient energy management.

3. Materials and Methods

The aim of this paper is to develop an intelligent system on drones that performs real-time object detection. We first describe the design of the FPGA that takes ML methods for object detection, then we explain the system design in detail.

3.1. FPGA Design

To perform real-time object detection using different NN models, we take advantage of an FPGA and design a reconfigurable accelerator architecture with it. The FPGA that we adopt is Kintex-7 410T, as shown in Figure 1. It contains 406,720 logic cells, 1540 DSP slices, 16 high-speed serial transceivers, 12.5 Gb/s transceiver aggregate bandwidth, 28 Mb block RAM, 500 I/O pins, and 800 Gb/s bandwidth. The Kintex-7 FPGA is ideal for applications including 3G and 4G wireless, flat panel displays, and video over IP solutions. We design an NN engine to accelerate NN models on it.

The architecture of the NN engine on the FPGA is displayed in Figure 2. Data are first transferred to the FPGA. Then the NN engine assembles the data into the data stream, which are used for the following processing. Specifically, data streams are sent to multiple fetch and jump pointers. Note that we adopt a multi-threading strategy such that parallel computation can be achieved using multiple weights and buffers.

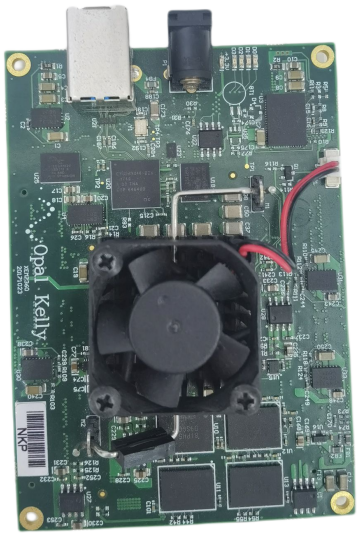

Figure 1. The Xilinx Kintex-7 410T FPGA.

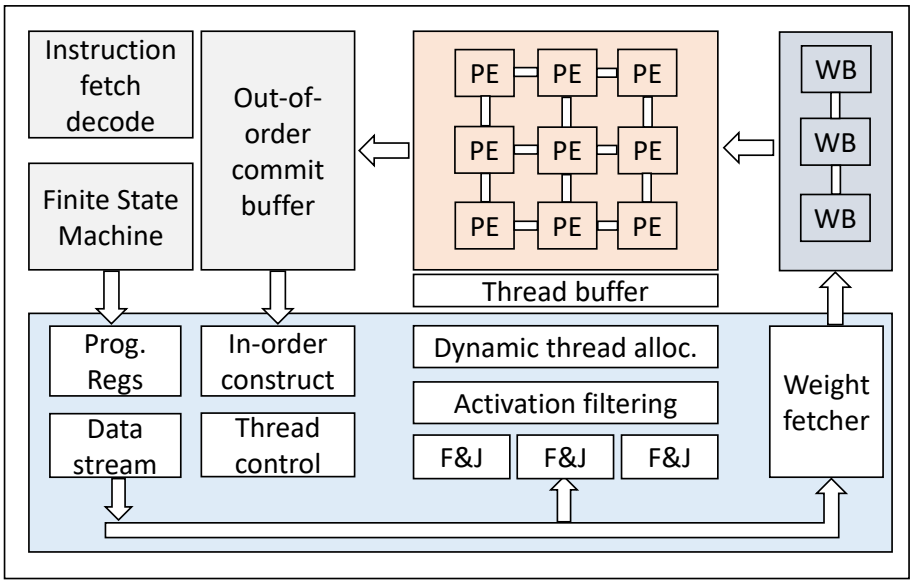

Figure 2. The FPGA NN engine accelerator architecture for neural networks.

A fetch and jump (F&J) mechanism is used to independently obtain required activation data for each thread from the data stream. Fetch pointers—which are programmed in accordance with the kernel structure—access the data stream in parallel, and fetch the activation data into specific L1 buffers. Together with a low-cost jump logic, a fetch pointer catches the required data individually. Then, we filter the activation data with sliding windows and use a dynamic thread-allocation module to mark thread buffer and thread status. At the same time, neural network weights are put into weight buffers. The activation data and weights are used to process image data using multiple processing elements (PEs) for NN acceleration, whose outputs are written to the out-of-order commit buffer. The out-of-order data are reconstructed by the in-order construction module and they are sent to the external data sink.

3.2. System Design

To build an intelligent system on drones, we first select the AMOVLAB Z410 drone model as the base platform. It is composed of an Intel Realsense T265 tracking camera, a flight controller, and a Raspberry Pi 4 Model B board host. The host includes a quad-core

Cortex-A72 (ARM v8) SoC @ 1.5GHz and the operating system is Ubuntu 20.04. We equip the FPGA to the drone for NN model acceleration and connect it to the host. The overview of the proposed system is shown in Figure 3.

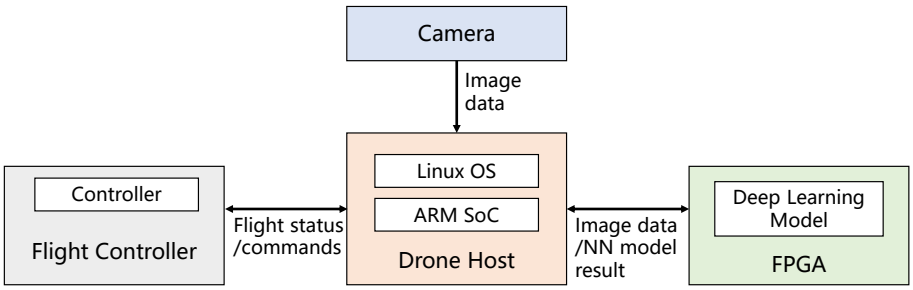

Figure 3. The architecture of the proposed intelligent system.

From Figure 3, we can see that the drone host is the central unit of the intelligent system and it is in charge of the system management. The camera sends image data to the drone host for processing. The host forwards the image data to the FPGA for AI processing using ML models, and the result is returned to the host. Meanwhile, the host sends flight status and commands to the flight controller—a separate module for flight control and adjustment—throughout the flight journey, such that the drone can fly smoothly and safely.

The intelligent system picture is shown in Figure 4. As explained in previous sections, the system includes four key components, i.e., the host, the camera, the flight controller, and the FPGA. In addition, other components are necessary to support the system, including the battery for power supply and GPS for outdoor positioning.

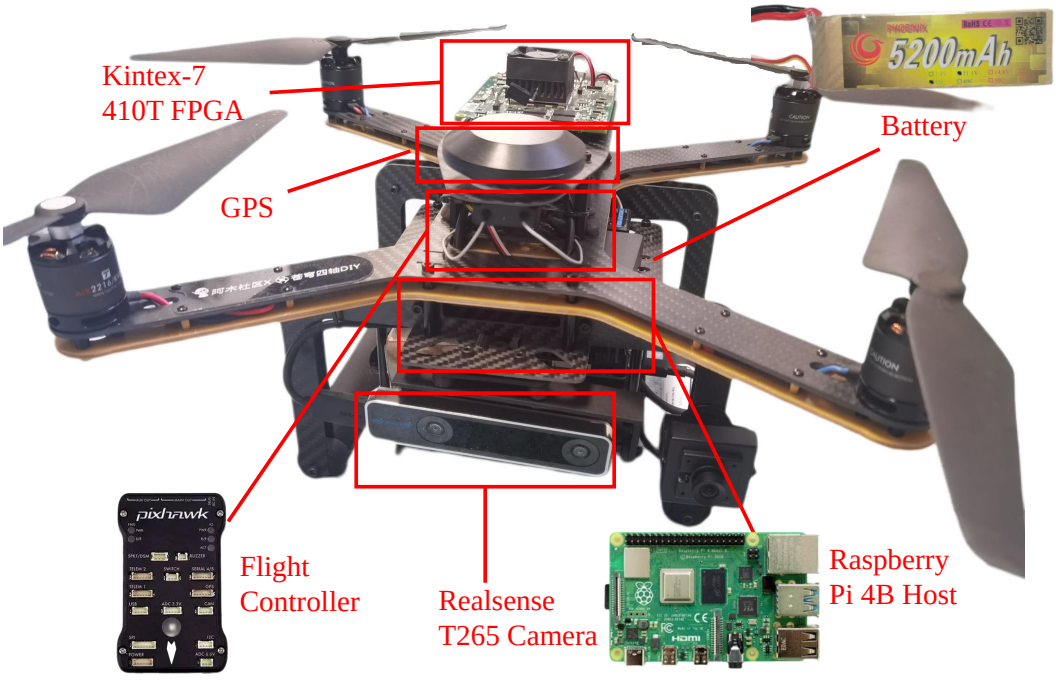

Figure 4. The proposed intelligent system.

3.3. Intelligent Object Detection

We note that when performing object detection, the objects may be misclassified in some cases. Therefore, we propose intelligent object detection techniques, as displayed in Figure 5. It generates the image and performs object detection on it. Only when the object detection is higher than or equal to the preset threshold, will the system report the image remotely. This way, there is no need to send every image to the remote machine, which significantly reduces the transferred image data.

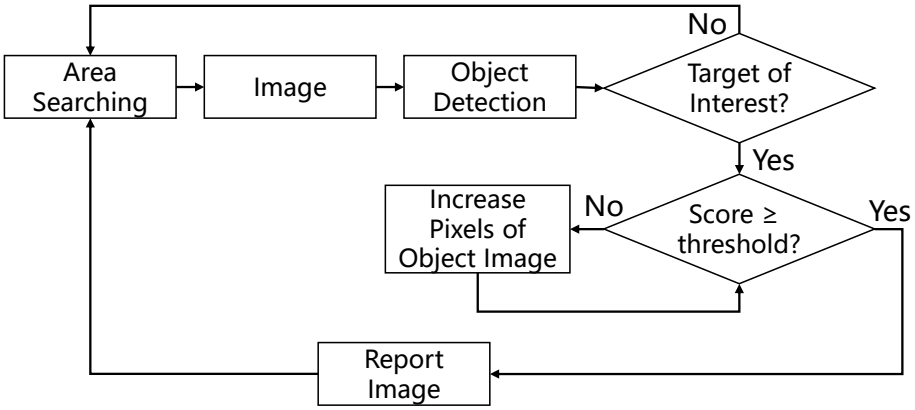

Figure 5. The workflow of the proposed intelligent object detection.

Specifically, in the workflow of Figure 5, the drone first starts flying in an area of interest and then looks around to search for different types of targets. Depending on the battery, a drone can fly typically between 10 to 30 minutes and power consumption is a concern for drones.

While flying, the drone captures images using equipped sensors, including pressure sensors, humidity sensors, accelerometer sensors, image cameras, etc. With the help of deep learning, it is feasible to detect objects using images. Therefore, we use the camera to capture images during the flight.

Upon the image data generation, the host transfers the image data to the FPGA for object detection. The FPGA contains the NN engine that can run various NN models, including ResNet-50, Yolov3, and Yolov3-tiny models. These models can be used to perform object classification and detection tasks with low power consumption.

Based on the object detection results, the system evaluates if there is any target of interest (TOI) in the search area. The TOI is the target that the user specifies in advance, and it may be a person, a car, or other objects. If no TOI is found, the drone moves to the next area of interest.

When TOI is found, our system looks into the confidence score of the object detection. We use a confidence-score threshold as the tuning hyperparameter for the user to adjust. When the confidence score is greater than or equal to the threshold, the system is confident that the image provides enough information and then sends the image to the remote machine for the report. The drone can then start searching the next area.

If the confidence score is below the threshold value, the drone flies around and attempts to take images of the object from different angles. By approaching the target, the system can generate images that may contain more pixels for the TOI. In this case, the confidence score will be effectively increased.

Note that the confidence score denotes the uncertainty of the object detection result from Yolo. By using a preset threshold, the uncertainty in decision-making can be reduced. When the threshold is set to a low value (e.g., 0.60), the uncertainty is high and the object may not be that clear in the image; when it is set to a high value (e.g., 0.95), the uncertainty decreases but the system may not be able to reach the threshold, which results in a deadlock.

Figure 6 illustrates the principle of intelligent object detection. The target of interest is the person, and in the first image, the object detection confidence score is 0.60 and it is below the threshold (e.g., 0.8). By providing more pixels of the person, the system can produce a more effective object detection with a higher confidence score—i.e., 0.90—and report the image accordingly.

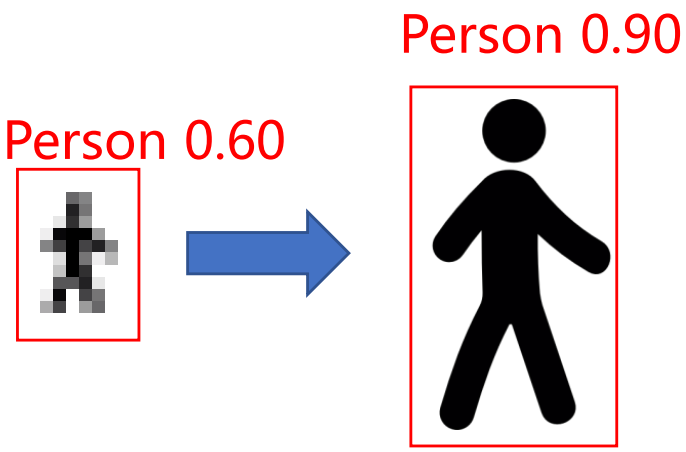

Figure 6. The illustration of intelligent object detection by providing more pixels for the target of interest, which effectively increases the object detection confidence score.

4. Results and Discussion

In order to test the effectiveness of the proposed intelligent system, we first evaluate the NN engine of the FPGA. Then, we perform experiments on the whole system for a full evaluation.

4.1. FPGA Resource Utilization

We first evaluate the resource consumption of the FPGA to see how many resources our NN engine has utilized, as listed in Table 2. Lookup table (LUT) blocks and DSP blocks are important computing resources. LUT blocks are the programmable fabric that can be used to implement arithmetic circuits. DSP blocks are a scarce resource on an FPGA chip and they are more efficient to implement arithmetic circuits compared to LUT blocks. Thus, it is important to use DSP blocks as efficiently as possible. We can see that 185,046 of 254,200 LUT blocks are utilized, i.e., with a 72.80% utilization rate. A total of 1256 of 1540 DSP blocks are utilized, i.e., with an 81.56% utilization rate. This means that LUT and DSP blocks are used efficiently, i.e., we have well utilized the computing resources of the FPGA.

Table 2. NN engine resource consumption of the FPGA.

Resource	Available	Utilization	Utilization %
LUT	254,200	185,046	72.80
LUTRAM	90,600	1297	1.43
FF	508,400	136,483	26.85
BRAM	795	304	38.24
DSP	1540	1256	81.56
IO	400	119	29.75
BUFG	32	5	15.63
MMCM	10	2	20.00
PLL	10	1	10.00

4.2. Power

Figure 7 illustrates the power consumption of the FPGA. It is shown that the total on-chip power is 3.529 W, where dynamic power is the dominant one with a portion of 93%. The on-chip power is not large and can be supplied with portal DC power sources. Thus, we adopt a battery bank to supply a 5 V voltage for the FPGA.

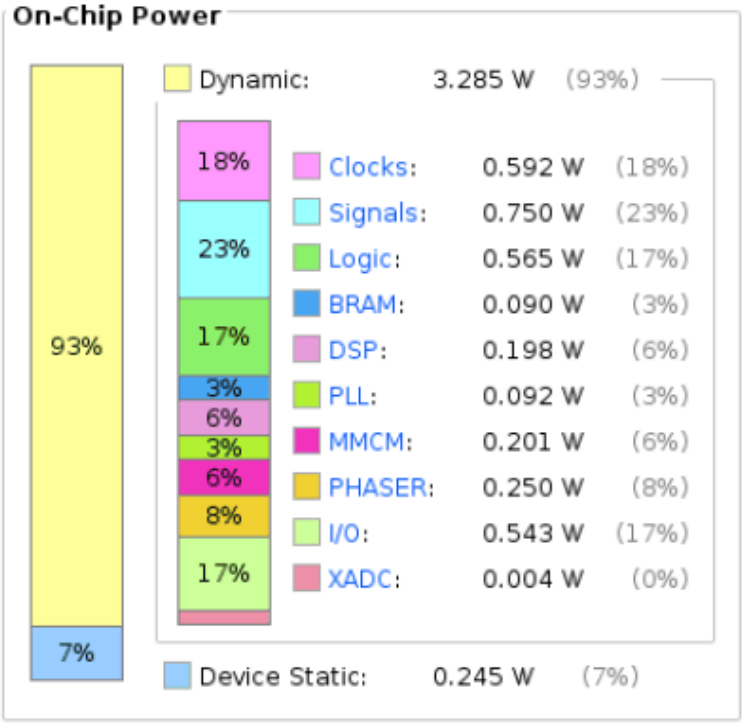

Figure 7. The power consumption of the FPGA.

There are two power sources for the intelligent system, i.e., a battery for the flying of the drone and a battery for the FPGA. Table 3 lists the battery-life duration of the drone and the FPGA. We can see that although the FPGA exhibits low-power characteristics that can work for 2 hours, the system's duration is determined by the battery duration of the drone, which is limited to the duration of 11 minutes.

Table 3. Battery duration for drone and the FPGA.

System Component	Battery Duration
Drone	11 min
FPGA	2 h

4.3. NN Model Speed

The NN engine is an accelerator for NN models and it can leverage various NN models for object classification and detection. We validate three models in our experiments, i.e., ResNet-50, Yolov3, and Yolov3-tiny models. Table 4 summarizes the NN models and their corresponding processing time. In general, NN models require a large number of weights that are trained using numerous labeled images. For ResNet-50 and Yolov3, the weight sizes are 147 MB and 384 MB, respectively. The weights need to be transferred from the host to the FPGA and, thus, it takes a long time—16.01 s for ResNet-50 and 39.31 s for Yolov3—to load the weights.

Though it is time-consuming to load the weights, the weight loading happens only once and it does not cost extra time after loading. On average, the processing time of each image is only 1.17 s and 0.8 s for Yolov3 and ResNet-50, respectively.

It is worth mentioning that Yolov3-tiny is a widely used NN model for object detection, which dramatically improves the processing speed. Compared to Yolov3, the weight size is reduced from 384 MB to 51 MB and the processing time decreases from 1.17 s to 0.13 s, i.e., 8 FPS. Therefore, we adopt the Yolov3-tiny NN model for real-time object detection.

Table 4. NN engine speed for different NN models.

NN model	Weight Size	Weight Loading Time	Processing Time
ResNet-50	147MB	16.01 s	0.8 s
Yolov3	384 MB	39.31 s	1.17 s
Yolov3-tiny	51 MB	5.88 s	0.13 s

Figure 8 displays the detailed time decomposition of different NN models, where weight loading denotes the time to load the NN model weight and i_m represents the time spent for object classification and detection for the ith image. Ten images are tested for each model. We find that Yolov3-tiny significantly outperforms Yolov3 and ResNet-50.

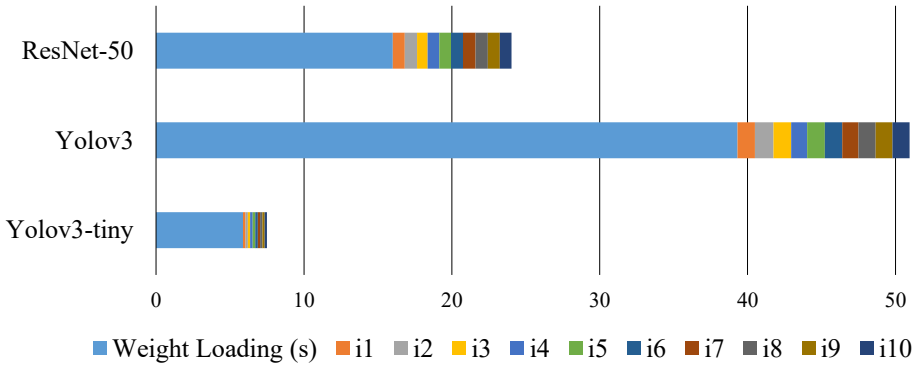

Figure 8. The detailed time decomposition of various NN models using ten images.

4.4. Comparison with Other Approaches

To evaluate the performance of our design, we compare our work with other approaches, as illustrated in Table 5. We randomly select 500 images from the public COCO dataset [12] for performance evaluation. It is evident that our work produces object detection with extremely low power at 3.525 W, which is significantly lower than the power of 23 W from work [11]. This is because we implement neural networks without SoC, which eliminates the operating system and, thus, dramatically reduces power consumption.

In addition, the runtime latency of our work is 130 ms, and it is much smaller than the runtime of work [9]. It is also comparable to the latency of work [10]. However, the FPGA of work [10] runs at 150 MHz, whiles ours runs at 100 MHz, which means that our design outperforms the work of [10] at the same operating frequency.

The object detection speed of our design is only 8 FPS, which is a disadvantage compared to other approaches. This is because our design is limited by the I/O property of the FPGA board, which is an inherent limitation, as indicated by the roofline model. We can improve the speed by using an FPGA board with a high I/O bandwidth in future designs.

Table 5. Benchmarking with state-of-the-art object detection using FPGA accelerators.

Implementation	Virtex-Yolo [9]	Zynq-Yolo [10]	Req-Yolo [11]	Our Work
Device Type	Xilinx Virtex-7 485t	Zynq 7020	ADM-7V3 FPGA	Kintex-7 410T
Clock Freq. (MHz)	140 MHz	150 MHz	200 MHz	100 MHz
Speed (FPS)	21	8	208	8
Power (W)	N/A	N/A	23	3.529
Latency (ms)	970	125	N/A	130

4.5. Intelligent Object Detection

We perform object detection and display some detection results in this section. We aim to find the targets of interest in an area using the drone and report detected objects in

real time. Note that due to the characteristics of the Yolo model, it is inevitable to produce wrong object detection results.

We randomly select some objects to evaluate if the system can increase the confidence score when it is below the specified value. Note that Figures 9–12 are from the camera and are not publicly available. Figure 9 shows an ordinary person-detection result, where some bikes are misclassified as a person.

Figure 9. The ordinary person-detection result.

With our intelligent object detection techniques, we empirically set the confidence score threshold to 0.8 and the drone approaches the detected target when the score is less than the threshold. On the one hand, it can generate more detection results, such that the wrong detection results may be spotted. On the other hand, it produces more pixels as the drone approaches, which helps improve the accuracy of object detection.

Figure 10 displays the person-detection result using intelligent object detection techniques. We can see that the confidence score of the person has improved from 0.74 to 0.90. In addition, the misclassified bikes have been corrected from person to bicycle with a confidence score of 0.50.

In addition to a person, we also demonstrate the effectiveness of the system for cars. Figure 11 illustrates the ordinary car detection and most confidence scores are between 0.50 and 0.70. Figure 12 shows the intelligent object detection mechanism for the car and the confidence scores have been varied, where the top score has been increased from 0.7 to 0.84.

From the experimental results, we can see that the system is intelligent in two ways. On the one hand, Table 5 validates that our system can intelligently process the image detection and consumes low power. On the other hand, Figures 9–12 indicates that when the TOI contains few pixels, the resulting confidence score is low and our system intelligently approaches the TOI to increase the object detection score, which leads to a larger TOI in the image.

Figure 10. The intelligent person-detection result.

It is worth mentioning that since our system can run various neural network models and signal processing tasks, the system is not limited to person and car detection and it can also be used for other object detection or even other types of tasks. For example, it can be adopted to accelerate the CNN model that monitors littering activities [32], and deal with thermographic fault diagnosis tasks using a feature-extraction technique [33].

Figure 11. The ordinary car-detection result.

Figure 12. The intelligent car-detection result.

5. Conclusions

In this research, we propose an intelligent object detection system on drones. We implement a power-efficient neural network engine on the FPGA and use it as the NN model accelerator. The image data are transferred by the host from the camera to the FPGA. To improve the accuracy of object detection, our intelligent detection approaches the target and generates images with more pixels for the target. Experimental results indicate that our system can achieve real-time detection at the speed of 8 FPS. Compared to other approaches, our work achieves low power consumption at 3.529 W with a lower latency of 130 ms. In addition, it can increase the detection confidence score from 0.74 to 0.90 and from 0.70 to 0.84 for persons and cars, respectively. In future work, we plan to apply our approach to various intelligent applications—such as smart city monitoring and fault diagnosis using AI techniques—and combine our design with state-of-the-art edge-computing and blockchain technologies to deal with data security issues while capturing images in real-time [34,35].

Author Contributions: Conceptualization, C.C. and Z.W.; methodology, C.C.; software, H.M.; validation, Y.P.; data curation, Y.Y.; writing—original draft preparation, C.C.; writing—review and editing, Z.W. and Y.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Key-Area Research and Development Program of Guangdong Province (grant number 2019B010155003), National Natural Science and Foundation of China (NSFC 61902355), and the Guangdong Basic and Applied Basic Research Foundation (grant number 2020B1515120044).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Shahmoradi, J.; Talebi, E.; Roghanchi, P.; Hassanalian, M. A comprehensive review of applications of drone technology in the mining industry. *Drones* **2020**, *4*, 34. [CrossRef]
2. Krul, S.; Pantos, C.; Frangulea, M.; Valente, J. Visual SLAM for indoor livestock and farming using a small drone with a monocular camera: A feasibility study. *Drones* **2021**, *5*, 41. [CrossRef]
3. Moshref-Javadi, M.; Winkenbach, M. Applications and Research avenues for drone-based models in logistics: A classification and review. *Expert Syst. Appl.* **2021**, *177*, 114854. [CrossRef]
4. Daud, S.M.S.M.; Yusof, M.Y.P.M.; Heo, C.C.; Khoo, L.S.; Singh, M.K.C.; Mahmood, M.S.; Nawawi, H. Applications of drone in disaster management: A scoping review. *Sci. Justice* **2022**, *62*, 30–42. [CrossRef] [PubMed]
5. Rapuano, E.; Meoni, G.; Pacini, T.; Dinelli, G.; Furano, G.; Giuffrida, G.; Fanucci, L. An fpga-based hardware accelerator for cnns inference on board satellites: Benchmarking with myriad 2-based solution for the cloudscout case study. *Remote Sens.* **2021**, *13*, 1518. [CrossRef]
6. Wang, Z.; Zhou, L.; Xie, W.; Chen, W.; Su, J.; Chen, W.; Du, A.; Li, S.; Liang, M.; Lin, Y.; et al. Accelerating hybrid and compact neural networks targeting perception and control domains with coarse-grained dataflow reconfiguration. *J. Semicond.* **2020**, *41*, 022401. [CrossRef]
7. Wang, J.; Gu, S. FPGA Implementation of Object Detection Accelerator Based on Vitis-AI. In Proceedings of the 2021 11th International Conference on Information Science and Technology (ICIST), Chengdu, China, 21–23 May 2021; pp. 571–577. [CrossRef]
8. Li, W.; Liewig, M. A survey of AI accelerators for edge environment. In Proceedings of the World Conference on Information Systems and Technologies, Budva, Montenegro, 7–10 April 2020; pp. 35–44.
9. Ma, J.; Chen, L.; Gao, Z. Hardware Implementation and Optimization of Tiny-YOLO Network. In *Proceedings of the Digital TV and Wireless Multimedia Communication*; Zhai, G., Zhou, J., Yang, X., Eds.; Springer Singapore: Singapore, 2018; pp. 224–234.
10. Guo, K.; Sui, L.; Qiu, J.; Yao, S.; Han, S.; Wang, Y.; Yang, H. From model to FPGA: Software-hardware co-design for efficient neural network acceleration. In Proceedings of the 2016 IEEE Hot Chips 28 Symposium (HCS), Cupertino, CA, USA, 21–23 August 2016; pp. 1–27. [CrossRef]
11. Ding, C.; Wang, S.; Liu, N.; Xu, K.; Wang, Y.; Liang, Y. REQ-YOLO: A Resource-Aware, Efficient Quantization Framework for Object Detection on FPGAs. In Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Seaside, CA, USA, 24–26 February 2019; Association for Computing Machinery: New York, NY, USA, 2019. FPGA '19, pp. 33–42. [CrossRef]
12. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollar, P.; Zitnick, L. Microsoft COCO: Common Objects in Context. In Proceedings of the ECCV, European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014.
13. Girshick, R. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1440–1448.
14. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 779–788.
15. Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 11–17 October 2021; pp. 10012–10022.
16. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
17. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
18. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
19. Liu, Y.; Bao, R.; Tao, J.; Li, J.; Dong, M.; Pan, C. Recent progress in tactile sensors and their applications in intelligent systems. *Sci. Bull.* **2020**, *65*, 70–88. [CrossRef]
20. Li, Y.; Shi, J.; Li, Y. Real-Time Semantic Understanding and Segmentation of Urban Scenes for Vehicle Visual Sensors by Optimized DCNN Algorithm. *Appl. Sci.* **2022**, *12*, 7811. [CrossRef]
21. Qiu, S.; Zhao, H.; Jiang, N.; Wu, D.; Song, G.; Zhao, H.; Wang, Z. Sensor network oriented human motion capture via wearable intelligent system. *Int. J. Intell. Syst.* **2022**, *37*, 1646–1673. [CrossRef]
22. Haseeb, K.; Almogren, A.; Ud Din, I.; Islam, N.; Altameem, A. SASC: Secure and Authentication-Based Sensor Cloud Architecture for Intelligent Internet of Things. *Sensors* **2020**, *20*, 2468. [CrossRef] [PubMed]
23. Injadat, M.; Moubayed, A.; Nassif, A.B.; Shami, A. Machine learning towards intelligent systems: Applications, challenges, and opportunities. *Artif. Intell. Rev.* **2021**, *54*, 3299–3348. [CrossRef]
24. Janiesch, C.; Zschech, P.; Heinrich, K. Machine learning and deep learning. *Electron. Mark.* **2021**, *31*, 685–695. [CrossRef]
25. Ha, N.; Xu, K.; Ren, G.; Mitchell, A.; Ou, J.Z. Machine Learning-Enabled Smart Sensor Systems. *Adv. Intell. Syst.* **2020**, *2*, 2000063. [CrossRef]

26. Prencipe, B.; Altini, N.; Cascarano, G.D.; Brunetti, A.; Guerriero, A.; Bevilacqua, V. Focal Dice Loss-Based V-Net for Liver Segments Classification. *Appl. Sci.* **2022**, *12*, 3247. [CrossRef]
27. Ali, S.; Hassan, M.; Kim, J.Y.; Farid, M.I.; Sanaullah, M.; Mufti, H. FF-PCA-LDA: Intelligent Feature Fusion Based PCA-LDA Classification System for Plant Leaf Diseases. *Appl. Sci.* **2022**, *12*, 3514. [CrossRef]
28. Haq, A.U.; Li, J.P.; Khan, J.; Memon, M.H.; Nazir, S.; Ahmad, S.; Khan, G.A.; Ali, A. Intelligent machine learning approach for effective recognition of diabetes in E-healthcare using clinical data. *Sensors* **2020**, *20*, 2649. [CrossRef]
29. González Rodríguez, G.; Gonzalez-Cava, J.M.; Méndez Pérez, J.A. An intelligent decision support system for production planning based on machine learning. *J. Intell. Manuf.* **2020**, *31*, 1257–1273. [CrossRef]
30. Deepa, N.; Prabadevi, B.; Maddikunta, P.K.; Gadekallu, T.R.; Baker, T.; Khan, M.A.; Tariq, U. An AI-based intelligent system for healthcare analysis using Ridge-Adaline Stochastic Gradient Descent Classifier. *J. Supercomput.* **2021**, *77*, 1998–2017. [CrossRef]
31. Zekić-Sušac, M.; Mitrović, S.; Has, A. Machine learning based system for managing energy efficiency of public sector as an approach towards smart cities. *Int. J. Inf. Manag.* **2021**, *58*, 102074. [CrossRef]
32. Husni, N.L.; Sari, P.A.R.; Handayani, A.S.; Dewi, T.; Seno, S.A.H.; Caesarendra, W.; Glowacz, A.; Oprzędkiewicz, K.; Sułowicz, M. Real-Time Littering Activity Monitoring Based on Image Classification Method. *Smart Cities* **2021**, *4*, 1496–1518. [CrossRef]
33. Glowacz, A. Thermographic Fault Diagnosis of Ventilation in BLDC Motors. *Sensors* **2021**, *21*, 7245. [CrossRef] [PubMed]
34. Gadekallu, T.R.; Pham, Q.V.; Nguyen, D.C.; Maddikunta, P.K.R.; Deepa, N.; Prabadevi, B.; Pathirana, P.N.; Zhao, J.; Hwang, W.J. Blockchain for edge of things: Applications, opportunities, and challenges. *IEEE Internet Things J.* **2021**, *9*, 964–988. [CrossRef]
35. Khan, A.A.; Laghari, A.A.; Gadekallu, T.R.; Shaikh, Z.A.; Javed, A.R.; Rashid, M.; Estrela, V.V.; Mikhaylov, A. A drone-based data management and optimization using metaheuristic algorithms and blockchain smart contracts in a secure fog environment. *Comput. Electr. Eng.* **2022**, *102*, 108234. [CrossRef]