electronics

MDPI

Article

Real-Time On-Board Deep Learning Fault Detection for Autonomous UAV Inspections †

Naeem Ayoub * and Peter Schneider-Kamp

Department of Mathematics and Computer Science, University of Southern Denmark, 5230 Odense M, Denmark; petersk@imada.sdu.dk

* Correspondence: nomi@imada.sdu.dk; Tel.: +45-71449089

† Proceedings of the 1st International Conference on Deep Learning Theory and Applications, 8–10 July 2020, Lieusaint, Paris (online stream), France.

Abstract: Inspection of high-voltage power lines using unmanned aerial vehicles is an emerging technological alternative to traditional methods. In the Drones4Energy project, we work toward building an autonomous vision-based beyond-visual-line-of-sight (BVLOS) power line inspection system. In this paper, we present a deep learning-based autonomous vision system to detect faults in power line components. We trained a YOLOv4-tiny architecture-based deep neural network, as it showed prominent results for detecting components with high accuracy. For running such deep learning models in a real-time environment, different single-board devices such as the Raspberry Pi 4, Nvidia Jetson Nano, Nvidia Jetson TX2, and Nvidia Jetson AGX Xavier were used for the experimental evaluation. Our experimental results demonstrated that the proposed approach can be effective and efficient for fully automatic real-time on-board visual power line inspection.

Keywords: unmanned aerial vehicles; edge computing; deep learning; object recognition; fault detection

Citation: Ayoub, N.; Schneider-Kamp, P. Real-Time On-Board Deep Learning Fault Detection for Autonomous UAV Inspections. *Electronics* **2021**, *10*, 1091. https://doi.org/10.3390/electronics10091091

Academic Editors: Byung-Gyu Kim and Eva Cernadas

Received: 29 March 2021
Accepted: 3 May 2021
Published: 5 May 2021

1. Introduction

Unmanned aerial vehicles (UAVs), commonly known as drones, are aircraft platforms that fly without direct human pilot interaction. These aircraft are operated indirectly by a pilot from the ground control station or by using autonomous operating systems and sensors mounted on it [1]. To inspect power lines, to date, most companies have used manual inspection methods involving humans, helicopters, or manually piloted UAVs. These types of inspection are rather expensive and slow, with some of them even being outright dangerous. To overcome these issues, research projects at some companies, as well as in the academic world are focusing on the development of artificial intelligence (AI)-based autonomous power line inspection and fault detection methods. In recent years, much focus has been on inspection architectures that partially automate the visual inspections by utilizing drones or climbing robots [2]. The increase in the computational capabilities of SBDs and the extended capabilities of UAV technologies have led many researchers to focus on and develop UAV-based autonomous object detection systems. There is a stream of research on vision-based applications for UAVs [3], which provides the potential to become game-changers in the inspection of power lines and other linear infrastructures.

Based on recent advances in drone technology, in this paper, we addressed a number of challenges regarding traditional power line inspection methods and develop autonomous algorithms by utilizing deep learning (DL) technology. We also created a medium-sized dataset of different classes based on normal and faulty components (Figures 1 and 2) for training the supervised classification models. During the development of the proposed architecture, we identified that there can be different factors that influence the inspection process: training data, the relatively small size of components and faults, unidentified

faults, and cluttered backgrounds in different lighting conditions. Figures 1 and 2 were modified and regenerated from [4] by adding new component and faults.

Figure 1. Sample images of normal component classes used to train the DL model (from top left to bottom right): Insulator Type 1, Insulator Type 2, Insulator Type 3, normal top pad [2], normal nut-bolt, antenna, Vibration Dampener Type 1, Vibration Dampener Type 2, Vibration Dampener Type 3, Vibration Dampener Type 4, normal electric transmission wires, and pylon, respectively.

Figure 2. Sample images of the most common faulty components used to train the DL model (from top left to bottom right): Insulator Rust Type 1, Insulator Rust Type 2, damaged vibration dampener, rusted vibration dampener, broken vibration dampener, missing vibration dampener, rusted nut-bolt, missing top pads [2], Pylon Corrosion Type 1, Pylon Corrosion Type 2, bird's nest, and broken wire, respectively.

1.1. Power Line Inspection Methods

Power lines are traditionally inspected at regular intervals by different inspection methods such as the human-centered power line inspection, semi-automated power line inspection, and UAV-based power line inspection methods [2].

1.1.1. Human-Centered Power Line Inspections

Human-centered power line inspection methods rely on human involvement in the form of inspectors. In these methods, the inspection of power lines is conducted by foot patrols or by helicopter-assisted surveys. When using the foot patrol method, a team typically consists of two or more inspectors traveling along the power lines by foot and inspecting the power lines with the help of binoculars or infrared cameras. Where a closer look is required, the power line is shut down, and one of the inspectors climbs the power tower and along the power line secured by a rope.

When using the helicopter-assisted inspection method, a team of inspectors travels by helicopter along power lines to take pictures of different components of the pylons. These images are further sent to inspectors for off-line inspection. The inspectors then identify the faults such as rusty components, bird's nests, broken wires, faulty and broken insulators, missing top pads, and other malformed or missing components.

These two inspection methods are human-centered and are still widely used by inspection companies in spite of a number of disadvantages such as high cost, extensive time consumption, lack of safety in harsh terrains, and even the impossibility of application in less than optimal weather conditions. The accuracy is rather low due to many components being hard to reach by humans or helicopters. The biggest disadvantage of helicopter-based inspection in addition to the very high costs is that flying close to power lines poses a life-threatening security risk as contact with the cables during the survey usually has fatal consequences [5].

1.1.2. Semi-Automated Power Line Inspections

Human-centered inspections are performed less frequently because of their high monetary and time costs. As an alternative to these manual inspections, semi-automated inspection methods are slowly being used by a few first-moving inspection companies. These methods provide a moderate boost to the speed of the inspection process, improve the accuracy, and moderately reduce the inspection costs. The most common techniques are semi-automated helicopter-assisted and climbing robot inspections. Automated helicopter-assisted inspections differ from manual helicopter-assisted inspections, because, in this method, vision-based object detection techniques are used after the collection of the inspection images and videos. For example, power mast detection can be applied for guiding cameras to automatically film the conductors, pylons, power components, and objects around the pylons and under the power lines [6]. Although this technique has reduced the dependency on human inspectors for visual observation and sped up the inspection process, the inspection costs are still high, and the safety issues remain challenging. To overcome these challenges, climbing robot inspection techniques have been adopted by different companies. In this method, a climbing robot carrying many sensors and cameras travels on the power-lines for inspection. Zhou et al. [7] indicated that climbing robots can lead to new challenges during the inspection such as damaging the power wires while traveling, difficulties while crossing obstacles on and around wires, and large time costs compared to the automated helicopter-assisted inspection method.

1.1.3. UAV-Based Power Line Inspections

UAV-based power line inspections are the most promising inspection method. Developing the technology to be more robust is one of the main challenges for many researchers and companies. In this method, UAVs are equipped with multiple cameras and sensors to travel along the power lines to inspect them and detect faults. This technology has made

some progress during the last few years because it has overcome most of the inspection challenges such as the cost of inspections, as well as inspection safety and speed.

1.2. DL Models for Object Detection

During the last two decades, many researchers have proposed different machine learning (ML)- and deep learning (DL)-based computer vision algorithms by considering supervised, semi-supervised, and unsupervised methods. DL-based object detection techniques are very popular these days. These advances in vision-based techniques are encouraging the power industry to consider replacing the traditional inspection methods and develop autonomous power line monitoring systems based on the use of UAVs. The main reason for building the autonomous inspection systems is the property of being able to detect a wide range of components and faults in a single inspection [8]. Reviews of different data sources for vision-based inspection and existing vision-based inspection systems can be found in [9].

1.3. Components to Build An Autonomous Powerline Inspection System

There are five main components to build an autonomous powerline inspection system.

- Data collection and data analysis
- Autonomous vision systems for UAVs to perform real-time inspection
- Suitable SBDs with a sufficiently strong GPU to run vision-based DL models for real-time on-board inspection
- Communication and mission control systems for BVLOS UAV systems
- Deep integration of path planning and control systems in a visual UAV-based inspection system

In this paper, we focus on the first three challenges, which are essential for designing and implementing an autonomous visual real-time on-board inspection system. We will discuss the remaining two challenges in future work. Note, though, that in particular, the fifth point depends on the ability to run DL models on-board and in real time to achieve a deep integration, e.g., by reusing component and fault detection results for adapting the path planning and the further collection of images. To address the first three challenges, we collected image data for different components and faults. Then, we trained DL models on the collected and partially human-labeled images. Experiments on different hardware during the field test phase showed that the proposed architecture can provide a solid base for building autonomous power line inspection systems. Figure 3 shows the flowchart of our proposed architecture for autonomous power line inspection.

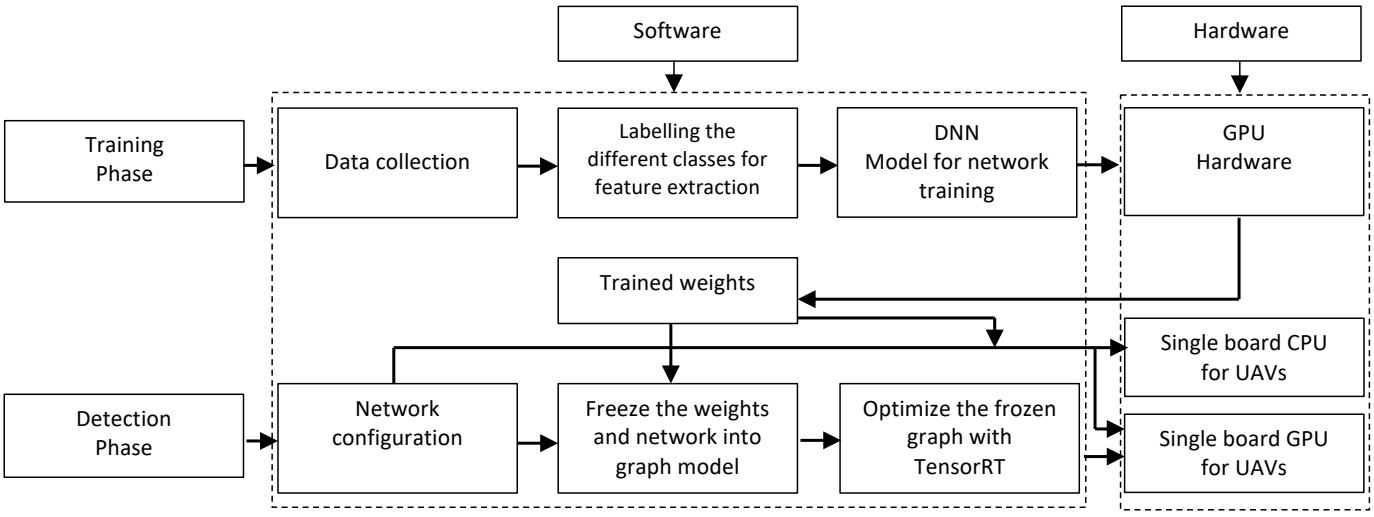


Figure 3. Proposed architecture for UAVs based on-board inspection and an autonomous vision system (Adapted from ref. [4]).

This study extends previous results published in conference proceedings [4]. The extended version contains the following additional contributions:

- We added additional types of components to the training dataset showing both normal and faulty components (Figures 1 and 2).
- We demonstrated that YOLOv4-tiny provides a promising platform for building vision-based fault detection system, as it showed higher accuracy and minimum loss (Table 1) with better FPS after optimization (Table 3) compared to previous (YOLO) architectures used in [4].
- We added extensive experimental evaluations of the object and fault detection systems (Section 3.3 and Figure 13) and compared with previous results (Figures 10 and 11).

The remainder of this paper is structured as follows: Section 2 presents relevant related work on different DL-based classification techniques for object detection. Our proposed autonomous power line inspection architecture and experimental results are described in Section 3. Finally, we conclude our results and further work in Section 4.

Table 1. Accuracy and average loss results of different DL models on different iterations.

Evaluation ⟶	**10,000 Iterations**		**20,000 Iterations**	
↓ DL Models	**Accuracy**	**avg. Loss**	**Accuracy**	**avg. Loss**
YOLOv3-tiny	74%	1.08	78.3%	0.97
YOLOv4-tiny	81.2 %	0.23	84.4%	0.15
YOLOv3-full	79.9 %	0.29	84.3%	0.11
YOLOv4-full (Mish)	82.4 %	0.38	83.6%	0.44
YOLOv4-full (leaky)	81.6 %	0.35	82.8 %	0.21

2. Related Work

In this section, we discuss some state-of-the-art single-stage and two-stage DL models for object detection and their architecture.

2.1. DL-Based Objects Classification and Detection Models

During the last few years, convolutional neural networks (CNNs) have been used for different computer vision techniques such as object detection [10], as well as image classification and semantic segmentation [11].

The CNN model has been improved in a variety of ways, and state-of-the-art object detection algorithms based on CNNs are flourishing. As CNN models are computationally rather expensive, it is not straightforward to use them in real-time image processing, in particular in situations where computational resources are limited such as SBDs onboard UAVs. In this section, we briefly discuss some CNN frameworks developed for the classification and detection of objects relevant to the development paper.

ResNet: The training of CNNs remained challenging in the above-mentioned variants of CNNs. To ease the training of neural networks, He et al. [12] introduced the deep residual network (ResNet) model. In this method, connections between the standard CNN layers allow the gradient signal to travel back directly from later layers to early layers. During the learning phase, the connection establishment technique of the ResNet layers allows the network model to be successfully trained even with 152 layers. Figure 4 shows the residual learning model (reproduced from [12]).

R-CNN: Girshik et al. [13] proposed the R-CNN method for object detection. In this method, region proposals obtained by selective search methods and then features are extracted with a CNN. A classifier based on support vector machines is used to classify these features. Finally, patches are optimized by bounding box regression. Figure 5 shows the flowchart of an R-CNN model (reproduced from [13]).

R-FCN: Dai et al. [14] introduced a new method based on region-based fully convolutional networks (R-FCNs) to address existing issues in R-CNN-based network architectures. As an alternative to applying region-level feature extraction, R-FCN adopts a FCN (fully convolutional network) architecture to share the computations across the image. In this method, position-sensitive score maps are obtained for classification and detection in a single evaluation. R-FCN performs 2.5–20 times faster and achieves higher accuracy than Faster R-CNN. Figure 6 shows the architecture of R-FCN (reproduced from [14]).

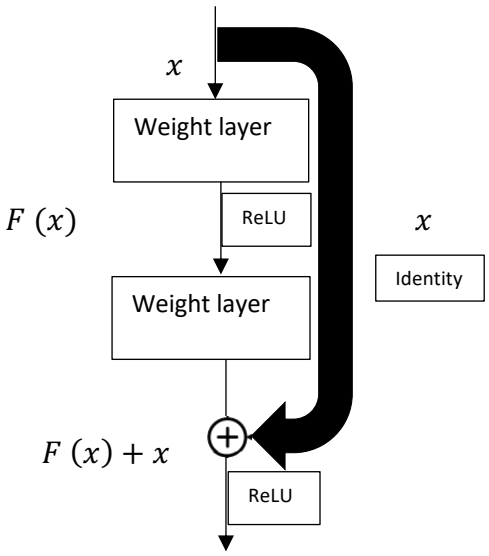


Figure 4. Residual learning model (Adapted from ref. [4]).

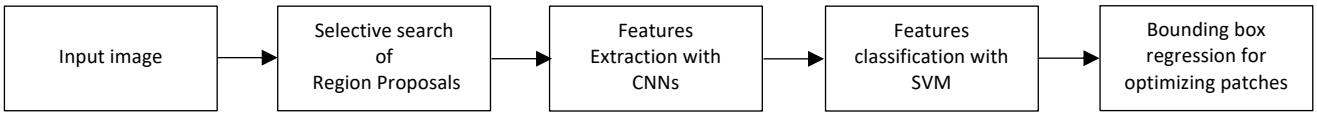


Figure 5. Flowchart of the R-CNN model (Adapted from ref. [4]).

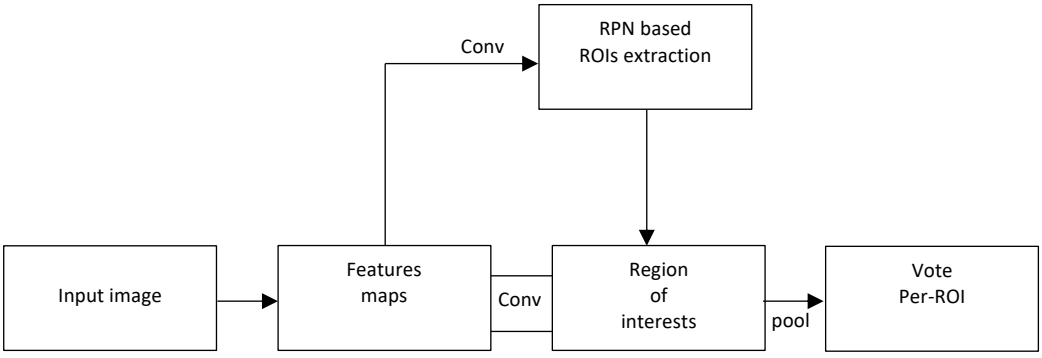


Figure 6. Region-based fully convolutional network model (Adapted from ref. [4]).

SSD: Liu et al. [15] proposed a single-shot detector (SSD) designed for real-time object detection. The SSD, as compared to two-stage object detectors, speeds up the process by eliminating the region proposal network (RPN) step and predicts the location with class scores using small convolution filters. The SSD has lower accuracy, as it uses a single-stage object detection method. In the SSD network, VGG16 was used to extract image features, and then, 3×3 convolution filters were added to make four predictions per cell regardless of the depth of the feature maps. To improve the performance, multi-scale features and

default box steps were added where lower resolution layers were used to detect larger scale objects. Each prediction contained the information of a boundary box and 21 scores per class. The SSD also contained "no object" information considering it as the 0 class to indicate the background. These improvements allowed the SSD to detect objects in real-time environments with reasonable accuracy. Figure 7 shows the SSD model architecture (reproduced from [15]).

YOLO: Redmon et al. [16] proposed a real-time object detection algorithm YOLO (you only look once). This model unifies region classification proposals into a single neural network to predict the bounding boxes and class probabilities. A single image is divided into $S \times S$ grid cells, and detection is performed into a single evaluation. This unique network structure makes YOLO much faster than the aforementioned algorithms. To improve accuracy, the YOLOv2 model was proposed by the same author, in which features such as direct location prediction, a high-resolution classifier, fine gradients, and dimension clustering were added to the YOLO network [17]. The authors introduced batch normalization and direct location prediction and replaced the fully connected layer with a convolution layer to speed up the training and detection process. In 2018, Redmon et al. introduced YOLOv3-darknet53 [18] to further improve the accuracy of YOLOv2 and added more layers and features to the network. In YOLOv3 models, darknet-53 is used as the backbone where 53 convolutional layers are used for feature extraction. Recently, Bochkovskiy et al. [19] introduced a new version of the YOLO series by adding some extra features to the YOLO models to further speed up detection and improve accuracy. In YOLOv4, instead of using darknet-53 layers for feature extraction, a modified version of CSPdarknet-53 is used as a backbone where cross-stage-partial connections (CSP) are used to separate the feature extraction connection into two parts. The YOLOv4-CSPdarknet version uses the Mish activation function instead of the leaky ReLU function used in YOLOv3-darknet53 and YOLOv4-tiny. Figure 8 shows the general architecture of YOLOv4-CSPdarknet53 (reproduced from [19]).

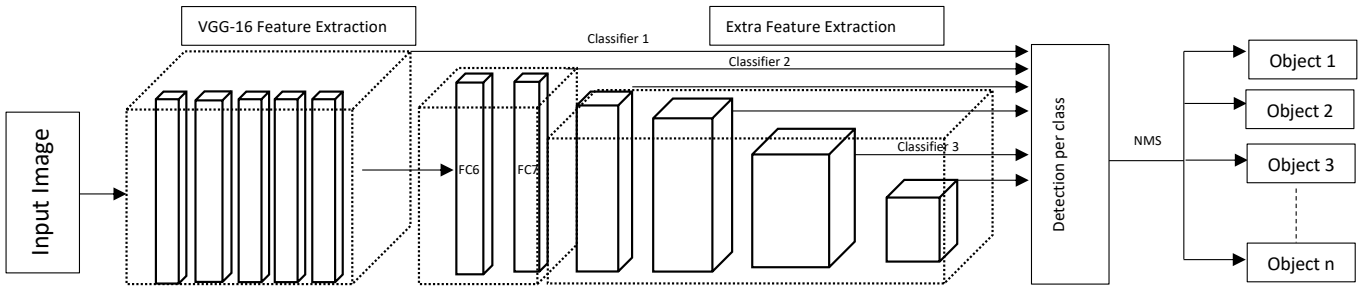


Figure 7. Architecture of the SSD model.

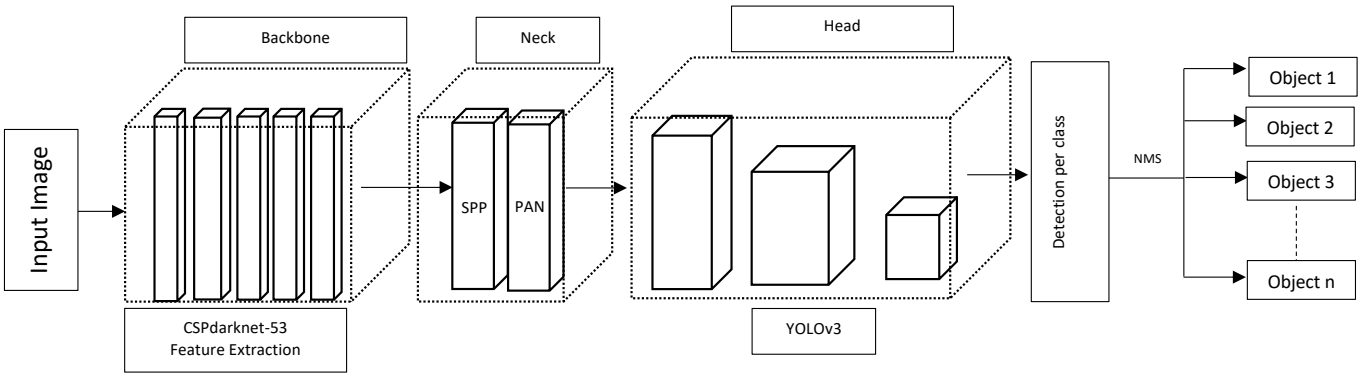


Figure 8. Architecture of the YOLOv4 model.

In this paper, to build the autonomous vision part of the proposed UAV-based inspection architecture, we took into consideration both speed and accuracy. We used a YOLOv4-tiny [19] DL model for developing the real-time autonomous fault detection system.

3. Proposed Real-Time On-Board Visual Inspection Model

Our architecture for the autonomous vision system for power line inspection was based on the three following main components:

- Collection and pre-analysis of a dataset.
- Application of DL algorithms for the training, testing, and analysis of the dataset.
- Selection of suitable SBDs for running the inference in real time on-board the UAV.

In this section, we discuss and summarize relevant data and classes of components and faults for training the autonomous detection algorithms. Then, we discuss the DL model for autonomous inspection. We summarize different SBDs and their compatibility issues with respect to their application for UAV-based real-time inspection. We also highlight the advantages of using SBDs with GPUs embedded within UAVs.

3.1. Data Collection and Pre-Analysis

Data collection and labeling the components of different classes for training the DL model are challenging because there are no publicly available datasets. After comprehensively reviewing different data sources and the structure of the components, we built a custom dataset with the help of an inspection company that is a collaborator in the Drones4Energy project. While collecting the image dataset, we considered different types of components and faults as separate classes according to their appearance in different backgrounds, angles, lighting, and weather conditions. Figure 1 shows the different components of the power lines and their related faults reproduced from [2]. In our dataset, we concentrated on five relevant classes to keep the time consumption for labeling by human experts at a reasonable level.

3.2. Suitable SBDs for UAV-Based Real-Time On-Board Inspections

Real-time on-board autonomous monitoring systems for power lines consist of two main components: DL algorithms and UAVs with embedded SBDs to run the DL algorithms. Taking into consideration the required computational power, we trained the DL models using more powerful hardware than available for the detection process, as discussed in the following.

Nvidia Tesla V100: Training DL models in a reasonable time requires rather powerful GPUs. We used an Nvidia Tesla V100 having 32GB RAM through the CUDA 10 framework. In this GPU, each streaming multiprocessor is partitioned into four processing blocks. Each block consists of two Tensor Cores, 16 FP32 cores, 8 FP64 cores, 16 INT32 cores, and one special function unit (SFU). This GPU has a total of 640 Tensor Cores, which jointly can accelerate the DL framework up to 125 TFLOPs [20].

We used a variety of different SBDs to perform inference with the DL model in order to assess and compare the real-time performance. Figure 9 shows the different SBDs that were considered as candidates to be embedded in UAVs.

Raspberry Pi 4: The Raspberry Pi 4 is the least expensive option in terms of price among the SBDs considered. This board is built with a 64 bit Broadcom Videocore VI GPU and a quad-core Cortex-A72 (ARM v8) CPU and has 4 GBs of RAM (see Figure 9a).

Nvidia Jetson Nano: The Jetson Nano is a small, more powerful SBD developed by Nvidia (see Figure 9b). It features a Maxwell architecture-based GPU and a quad-core ARM Cortex-A57 CPU and, as the Raspberry Pi 4, has 4 GBs of RAM. The GPU comes with a total of 128 CUDA cores, which can accelerate the DL framework up to 0.5 TFLOPs.

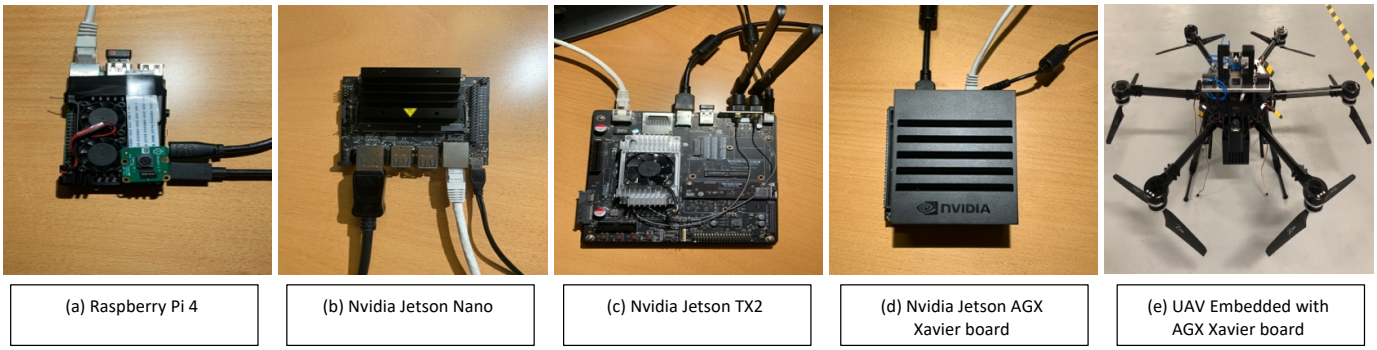


Figure 9. Single-board hardware for UAVs to run the real-time autonomous algorithm. From left to right, (**a**) Raspberry Pi 4, (**b**) Nvidia Jetson Nano, (**c**) Nvidia Jetson TX2, (**d**) Nvidia AGX Xavier, and (**e**) UAV embedded with the AGX Xavier board.

Nvidia Jetson TX2: The Jetson TX2 is an even more powerful SBD. Nvidia has equipped it with a more modern Pascal architecture-based GPU (see Figure 9c). The Jetson TX2 has two CPUs: a dual-Core Nvidia Denver and a quad-core ARM Cortex-A57, sharing 8 GBs of RAM. The GPU comes with a total of 256 CUDA cores, which can accelerate the DL framework up to 1.3 TFLOPs.

Nvidia AGX Xavier: Nvidia's flagship SBD, the AGX Xavier, is one of the most powerful SBDs on the market (see Figure 9d). It has a Volta architecture-based GPU with a total of 512 CUDA cores and 64 Tensor Cores. It also has an 8-core Nvidia Carmel ARM v8.2 64 bit CPU. The AGX Xavier can reach up to 32 TFLOPs and is specifically designed for running inference on DL models in real-time environments. The board can work in a number of different power modes, which gives the user the possibility to select the number of working CPU cores and, thereby, to control the power consumption of the SBD.

3.3. Autonomous DL Algorithm for Real-Time Inspection

For the development of the autonomous detection algorithm, we used the DL model of YOLOv4-tiny proposed by Alexey et al. [19].

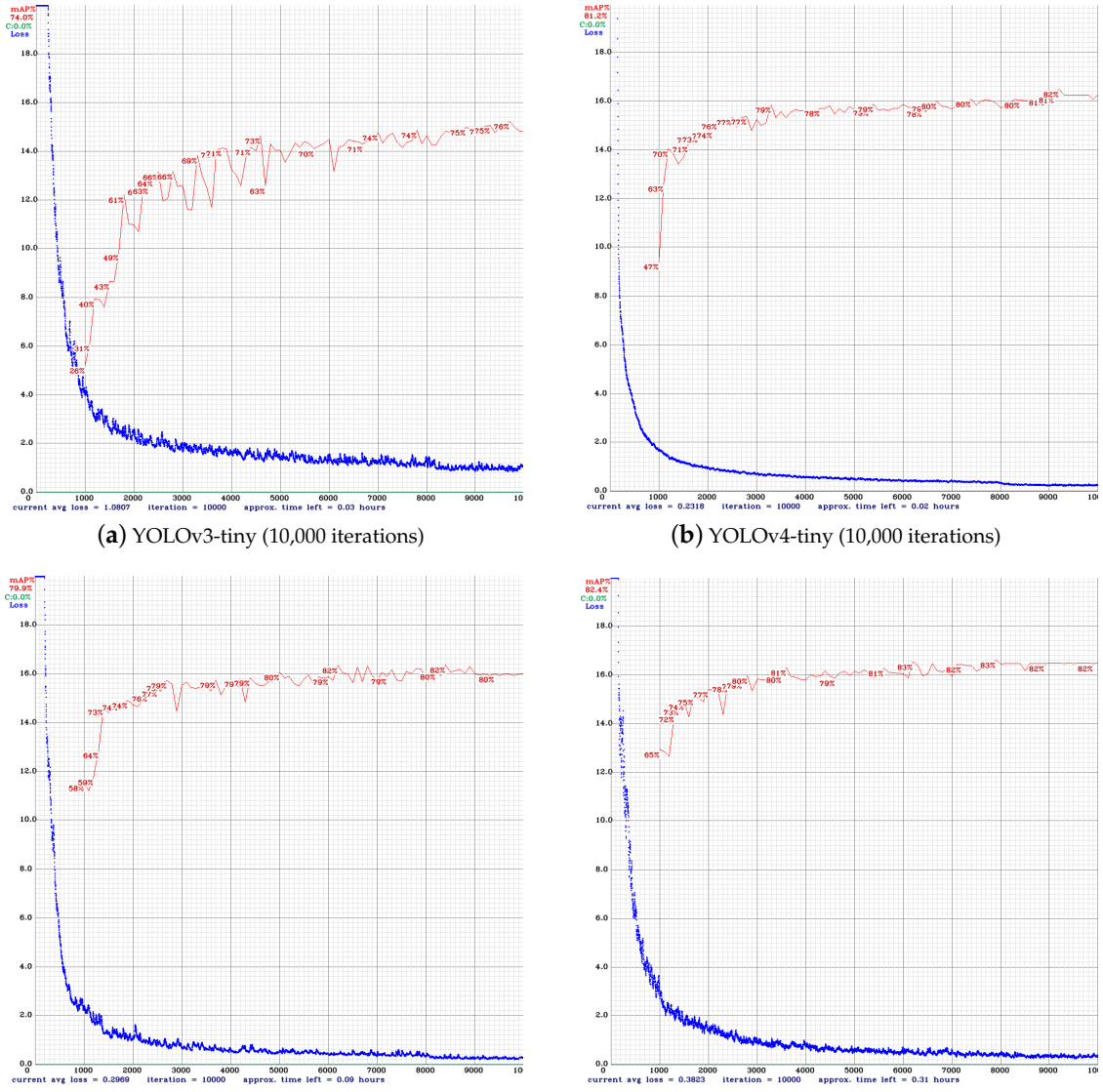
3.3.1. Training

We trained four different DL models YOLOv4-CSPdarknet53, YOLOv4-tiny, YOLOv3-darknet53, and YOLOv3-tiny to build the autonomous algorithm and compared the performance in the real-time environment. We tested the influence of different training improvement techniques after 10,000 and 20,000 iterations (Figures 10 and 11) on our proposed custom dataset. We also tested the impacts of different activation functions to activate neurons such as leaky ReLU and Mish on accuracy and loss.

YOLOv4-tiny and YOLOv3-tiny have fewer convolutional layers than YOLOv4-CSPdaknet53 and YOLOv3-darknet53, which improves their suitability for real-time processing, but in theory, reduces the accuracy somewhat. Concerning the parameters, we used the default configurations of these DL models and compared the training results with the previous findings [4] based on YOLOv3-darknet53 and YOLOv3-tiny. The momentum of the stochastic gradient descent was set to 0.9 and the learning rate 0.001. The weight decay was set to 0.005. Concerning the scaling of the images for the training of the network, we set the image size to 608 (standard). The batch size was set to 64 to improve the utilization of the GPU and its memory. During the training, we ignored the anchors that dropped below the threshold value. As training required a rather powerful GPU, we used the Nvidia Tesla V100 GPU to train the DL models.

It can be realized from Table 1 that, after completing 10,000 iterations, YOLOv3-tiny yielded a 74% accuracy with a loss of 1.08, YOLOv4 a 81.2% accuracy with a loss of 0.23, YOLOv3-darknet53 a 79.9% accuracy with a loss of 0.29, YOLOv4-Mish a 82.4% accuracy with a loss of 0.38, and YOLOv4-leaky with the leaky ReLU activation function less accuracy (81.6%) with minimum loss (0.35) when compared to YOLOv4-Mish. As the number of iterations was increased to 20,000, YOLOv4-Mish saw a drop in loss to 0.44 as compared to

YOLOv4-leaky with 0.21. Among all four models, YOLOv4-tiny showed the best training results with an accuracy of 84.4% and a loss of 0.15. YOLOv3-darknet53 also yielded an even lower loss of 0.11 with 84.3% accuracy, but used 53 convolution layers as compared to the 29 convolution layers used in YOLOv4-tiny, which made it computationally significantly more expensive. Figures 10 and 11 show the graphical representation of training accuracy and loss during 10,000 and 20,000 iterations.



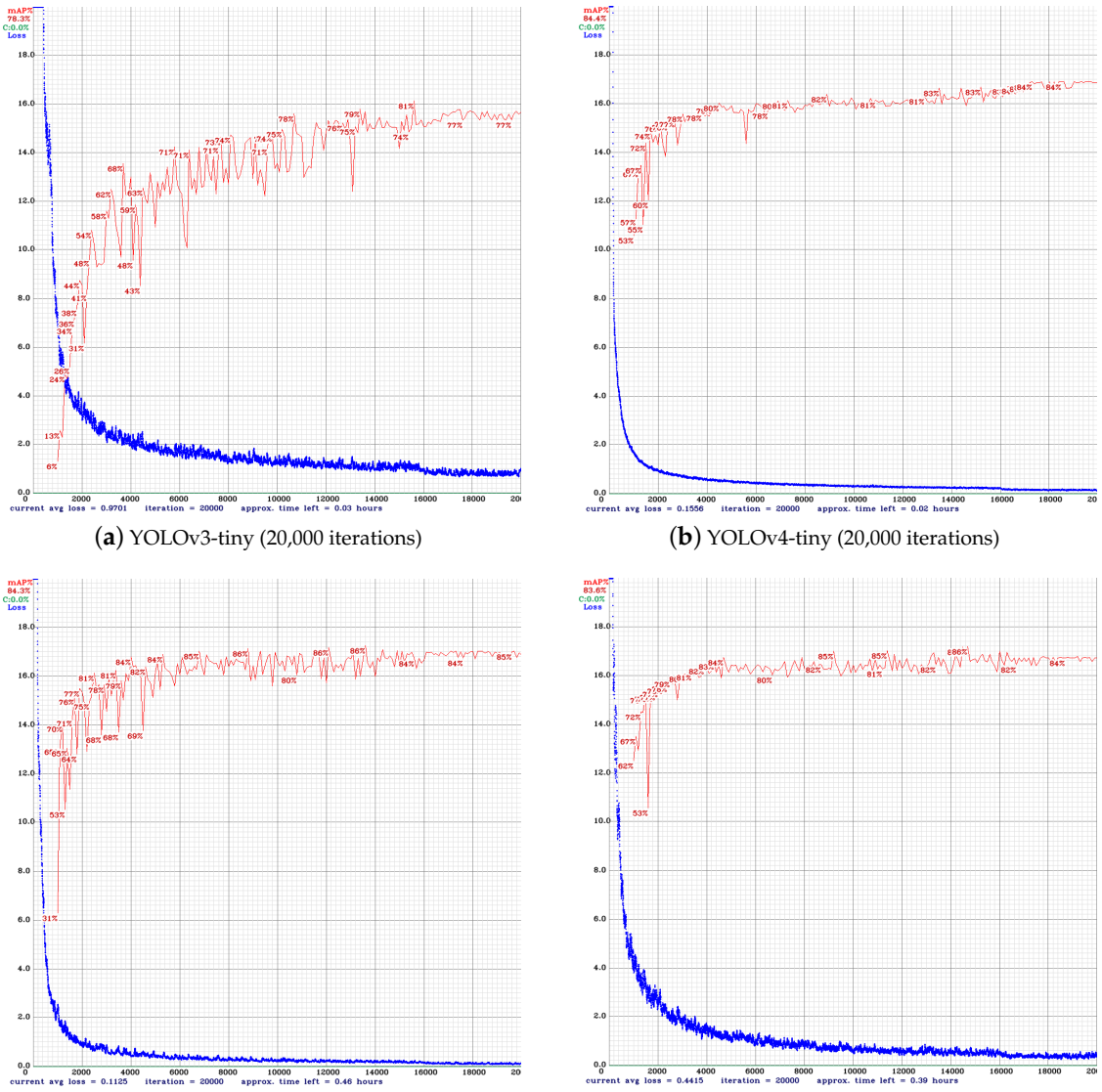
(**a**) YOLOv3-tiny (10,000 iterations)

(**b**) YOLOv4-tiny (10,000 iterations)

(**c**) YOLOv3 (10,000 iterations)

(**d**) YOLOv4 (10,000 iterations)

Figure 10. YOLO training results after 10,000 iterations: YOLOv4-CSPdarknet53 shows higher loss with the Mish function as compared to YOLOv4-tiny and YOLOv3-darknet53 because the leaky ReLU activation performs better in terms of minimizing the loss, though at the cost of some accuracy.



(**a**) YOLOv3-tiny (20,000 iterations)

(**b**) YOLOv4-tiny (20,000 iterations)

(**c**) YOLOv3 (20,000 iterations)

(**d**) YOLOv4 (20,000 iterations)

Figure 11. YOLO training results after 20,000 iterations, where we observe that YOLOv4-CSPdarknet53 shows higher loss as compared to YOLOv4-tiny and YOLOv3-darknet53 because ReLU performs better in terms of minimizing the loss rather than maximizing accuracy.

3.3.2. Testing

To run the inference faster on the SBDs, we accelerated the DL algorithm using the TensorRT library. TensorRT is a DL library introduced by Nvidia that optimizes the trained weights. For optimization, the trained weights are frozen, and then, these frozen weights are optimized with TensorRT. The optimized weights run much faster than non-optimized weights (compare Tables 2 and 3.)

3.4. Experimental Results and Discussion

For the evaluation of the real-time image processing, we used different SBDs, which can be embedded as part of the UAVs. Tables 2–4 show the experimental evaluation in terms of frames per second (FPS) using different hardware for running real-time inference with the DL algorithms. Table 2 shows the FPS during real-time processing with YOLOv4-tiny on different hardware. YOLOv4-tiny ran much faster than the YOLOv4 model (compare Tables 3 and 4), but the accuracy dropped after weight optimization. We can clearly see that the processing of the algorithm with non-optimized weights was too slow for a real-world

implementation (see Table 2), even when only using YOLOv4-tiny. We could not run the inference with the YOLOv4-CSPdarknet53 model on the Raspberry Pi due to too large memory demands.

Table 2. FPS on different scales of images during real-time detection (YOLOv3-tiny and YOLOv4-tiny without weight optimization).

DL Model	YOLOv3-Tiny (Non-Optimized)			YOLOv4-Tiny (Non-Optimized)		
Input Size →	288	416	608	288	416	608
Raspberry Pi 4	3	1	0.2	–	–	–
Nvidia Jetson nano	3.4	1.2	0.5	3	2.5	1.1
Nvidia Jetson TX2	20	17	10	8	7	5.4
Nvidia AGX Xavier	30	21.6	14	16	15	12

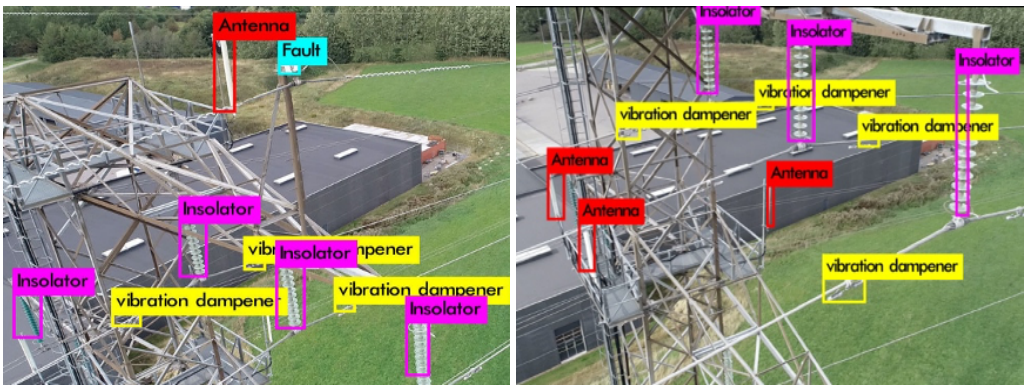
Table 3. FPS of YOLOv3-tiny and YOLOv4-tiny models on different scales of images during real-time detection with optimized weights.

DL Model	YOLOv3-Tiny Optimized			YOLOv4-Tiny Optimized		
Input size →	288	416	608	288	416	608
Nvidia Jetson Nano	22	15	4.5	9.2	7.8	6
Nvidia Jetson TX2	25	19	12	14	11.5	4.3
Nvidia AGX Xavier	50	32	22	26	22	19

Table 4. FPS for YOLOv3 (darknet53) and YOLOv4 (CSPdarknet-53) on different scales of images during real-time detection with optimized weights.

DL Model	YOLOv3-darknet53 Optimized			YOLOv4-CSPdarknet53 Optimized		
Input size →	288	416	608	288	416	608
Nvidia Jetson Nano	5.28	3	1.45	3.7	2.3	1.2
Nvidia Jetson TX2	11.4	6.4	3	7.2	6.6	4.3
Nvidia AGX Xavier	24	17	11	20	13	9.6

After optimizing the weights with the TensorRT library, the performance on the Nvidia Jetson TX2 and the Jetson AGX Xavier improved significantly (see Tables 3 and 4). We know that a DL algorithm for an autonomous vision system should have both properties: accuracy and real-time suitability. Hence, during the experimental evaluation, we chose the YOLOv4-tiny model and Nvidia Jetson AGX Xavier for the real-world implementation of our vision system in the inspection architecture. Jetson AGX Xavier can run the YOLOv4-tiny algorithm up to 22 FPS on images scaled to 416 with an accuracy 84.4% and a loss of 0.15 (see Table 1 and 3). Such a frame rate is acceptable for real-time processing and allows the drone control software to react to the results of the detection without undue delay. The experiments were performed by setting the AGX Xavier board to maximum performance mode (MAXN0), i.e., all CPU and GPU cores were enabled at full speed. Figure 12 shows the detection results of different powerline components with optimized weights of YOLOv3-darknet53. Figure 13 shows the detection results of powerline components with optimized YOLOv4-tiny weights.



(**a**) (**b**)

Figure 12. Detection results of different components of power line pylons based on YOLOv3-darknet53.

(**a**) (**b**)

Figure 13. Detection results of different components of power line pylons based on YOLOv4-tiny.

4. Conclusions

In this study, we proposed and evaluated the use of autonomous DL algorithms running in real time on-board UAVs with the purpose of visual power line inspection. We also compared the results between different SBDs that can be embedded in UAVs for running inference for powerful DL models. We further compared real-time results among YOLOv3-tiny, YOLOv3-darknet53, YOLOv4-tiny, and YOLOv4 (CSPdarknet-53) and investigated the real-world impact of weight optimization using TensorRT. We gave the theoretical description of different algorithms and SBDs and then practically implemented each algorithm on these SBDs. We realized that YOLOv4-tiny showed higher accuracy with a better frame rate during real-time inspection; YOLOv3-darknet53 also showed an acceptable level in terms of accuracy, but it showed lower FPS during detection. The experiments were performed on Nvidia Jetson AGX Xavier with optimized weights, which therefore will constitute the SBD for the Drones4Energy project.

In the future, we will extend our work by considering the remaining two challenges of performing multiple tasks on a single UAV regarding path planning, communication, and control, as well as autonomously sending extracted components and faults images to the cloud database for further processing the results of component and faults through a cloud service.

Author Contributions: Conceptualization, N.A.; data curation, N.A. and P.S.-K.; funding acquisition, P.S.-K.; methodology, N.A.; project administration, P.S.-K.; software, N.A.; supervision, P.S.-K.; validation, N.A. and P.S.-K.; writing—original draft, N.A.; writing—review and editing, P.S.-K. All authors read and agreed to the published version of the manuscript.

Funding: The presented research was supported by the Innovation Fund Denmark, Grand Solutions, under Grant Agreement No. 8057-00038A Drones4Energy project.

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

References

1. Mohamed, N.; Al-Jaroodi, J.; Jawhar, I.; Idries, A.; Mohammed, F. Unmanned aerial vehicles applications in future smart cities. *Technol. Forecast. Soc. Chang.* **2020**, *153*, 119293. [CrossRef]
2. Jenssen, R.; Roverso, D. Automatic autonomous vision-based power line inspection: A review of current status and the potential role of deep learning. *Int. J. Electr. Power Energy Syst.* **2018**, *99*, 107–120.
3. Al-Kaff, A.; Martín, D.; García, F.; de la Escalera, A.; Armingol, J.M. Survey of computer vision algorithms and applications for unmanned aerial vehicles. *Expert Syst. Appl.* **2018**, *92*, 447–463. [CrossRef]
4. Ayoub, N.; Schneider-Kamp, P. Real-time On-board Detection of Components and Faults in an Autonomous UAV System for Power Line Inspection. In Proceedings of the 1st International Conference on Deep Learning Theory and Applications—Volume 1: DeLTA, INSTICC, Paris, France, 8–10 July 2020; SciTePress: Setúbal, Portugal, 2020; pp. 68–75. [CrossRef]
5. Takaya, K.; Ohta, H.; Kroumov, V.; Shibayama, K.; Nakamura, M. Development of UAV System for Autonomous Power Line Inspection. In Proceedings of the 2019 23rd International Conference on System Theory, Control and Computing (ICSTCC), Sinaia, Romania, 9–11 October 2019; pp. 762–767.
6. Bühringer, M.; Berchtold, J.; Büchel, M.; Dold, C.; Bütikofer, M.; Feuerstein, M.; Fischer, W.; Bermes, C.; Siegwart, R. Cable-crawler—robot for the inspection of high-voltage power lines that can passively roll over mast tops. *Ind. Robot. Int. J.* **2010**, *3*. [CrossRef]
7. Zhou, G.; Yuan, J.; Yen, I.L.; Bastani, F. Robust real-time UAV based power line detection and tracking. In Proceedings of the 2016 IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, USA, 25–28 September 2016; pp. 744–748.
8. Zhou, X.; Fang, B.; Qian, J.; Xie, G.; Deng, B.; Qian, J. Data Driven Faster R-CNN for Transmission Line Object Detection. In *Cyberspace Data and Intelligence, and Cyber-Living, Syndrome, and Health*; Springer: Singapore, 2019; pp. 379–389. [CrossRef]
9. Contreras-Cruz, M.A.; Ramirez-Paredes, J.P.; Hernandez-Belmonte, U.H.; Ayala-Ramirez, V. Vision-Based Novelty Detection Using Deep Features and Evolved Novelty Filters for Specific Robotic Exploration and Inspection Tasks. *Sensors* **2019**, *19*, 2965. [CrossRef]
10. Zou, Z.; Shi, Z.; Guo, Y.; Ye, J. Object Detection in 20 Years: A Survey. *arXiv* **2019**, arXiv:1905.05055.
11. Li, Y.; Hao, Z.; Lei, H. Survey of convolutional neural network. *J. Comput. Appl.* **2016**, *36*, 2508–2515.
12. 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 (CVPR), Las Vegas, NV, USA, 27–30 June 2016.
13. 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.
14. Dai, J.; Li, Y.; He, K.; Sun, J. R-FCN: Object Detection via Region-based Fully Convolutional Networks. *arXiv* **2016**, arXiv:1605.06409.
15. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*; Springer: Amsterdam, The Netherlands, 2016; pp. 21–37.
16. 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 (CVPR), Las Vegas, NV, USA, 27–30 June 2016.
17. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. *arXiv* **2016**, arXiv:1612.08242.
18. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:1804.02767.
19. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv* **2020**, arXiv:2004.10934.
20. Markidis, S.; Der Chien, S.W.; Laure, E.; Peng, I.B.; Vetter, J.S. Nvidia tensor core programmability, performance & precision. In Proceedings of the 2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), Vancouver, BC, Canada, 21–25 May 2018; pp. 522–531.