electronics

MDPI

Article

An Improved Lightweight Deep Learning Model and Implementation for Track Fastener Defect Detection with Unmanned Aerial Vehicles

Qi Yu, Ao Liu *, Xinxin Yang * and Weimin Diao

School of Electronic Information Engineering, Beihang University, Beijing 100191, China; 16231012@buaa.edu.cn (Q.Y.); diaoweimin@buaa.edu.cn (W.D.)
* Correspondence: buaaliuao@buaa.edu.cn (A.L.); yangxx@buaa.edu.cn (X.Y.)

Abstract: Track fastener defect detection is an essential component in ensuring railway safety operations. Traditional manual inspection methods no longer meet the requirements of modern railways. The use of deep learning image processing techniques for classifying and recognizing abnormal fasteners is faster, more accurate, and more intelligent. With the widespread use of unmanned aerial vehicles (UAVs), conducting railway inspections using lightweight, low-power devices carried by UAVs has become a future trend. In this paper, we address the characteristics of track fastener detection tasks by improving the YOLOv4-tiny object detection model. We improved the model to output single-scale features and used the K-means++ algorithm to cluster the dataset, obtaining anchor boxes that were better suited to the dataset. Finally, we developed the FPGA platform and deployed the transformed model on this platform. The experimental results demonstrated that the improved model achieved an mAP of 95.1% and a speed of 295.9 FPS on the FPGA, surpassing the performance of existing object detection models. Moreover, the lightweight and low-powered FPGA platform meets the requirements for UAV deployment.

Keywords: track; fastener defect detection; model improvement; FPGA; UAV

Citation: Yu, Q.; Liu, A.; Yang, X.; Diao, W. An Improved Lightweight Deep Learning Model and Implementation for Track Fastener Defect Detection with Unmanned Aerial Vehicles. *Electronics* **2024**, *13*, 1781. https://doi.org/10.3390/electronics13091781

Academic Editor: Mahmut Reyhanoglu

Received: 4 April 2024
Revised: 26 April 2024
Accepted: 1 May 2024
Published: 5 May 2024

1. Introduction

Track fasteners are essential components that connect the rails to the sleepers and used to secure the rails and prevent lateral and longitudinal displacement [1]. Due to factors such as wear and tear on train wheels and the irregular deformation of the tracks over prolonged periods of train operation, trains are prone to vibrations during high-speed travel. These vibrations not only affect the trains themselves but are also transmitted to the track fasteners. Coupled with the impact of train loads, this can lead to the fracture and damage of track fasteners, thereby affecting the safe operation of trains. Common abnormalities in track fasteners include fracture, displacement, and dislodgement [2].

The speed and mileage of high-speed trains are gradually increasing, and urban rail transit is also developing gradually. Therefore, the efficient detection of track fastener defects is crucial. Initially, track fastener defect detection relied mainly on manual visual inspection. This method is inefficient, costly in terms of labor, and has unreliable accuracy, making it incapable of meeting modern requirements. This has also been confirmed in railway bridge inspection [3]. In order to improve detection efficiency, researchers have developed non-destructive testing methods, such as detection based on vibration signals [4,5], ultrasonic detection [6], laser detection [7], and machine vision detection [8]. With the rapid development of artificial intelligence, machine vision-based detection methods have emerged in various scenarios, including track fastener defect detection. Currently, machine vision-based detection methods can be categorized into two main types: those based on traditional image processing techniques and those based on deep learning approaches.

Detection methods based on image processing rely on manually designed features. After extracting features, a trained classifier is used for detection and classification. This results in the detection performance being influenced by manually designed features. Therefore, these algorithms often have lower detection accuracy and poor adaptability to variations in factors such as lighting and noise in real-world engineering scenarios, leading to low robustness. Khan et al. [9] utilized Harris–Stephens and Shi–Tomasi feature detectors to extract feature points and feature vectors from images. Subsequently, they matched the features of input images with those of training images to detect track fasteners. Feng et al. [10] proposed a probabilistic structural subject model (STM) to model the fastener. This model can detect the wear state of the fastener and is robust to changes in lighting conditions. Gibert et al. [11,12] proposed a fastener detection algorithm based on a multi-task learning framework. The algorithm uses image-oriented gradient histograms (HOGs) to extract fastener features and uses support vector machine (SVM) classifiers to classify and recognize damaged and missing fasteners, improving detection accuracy. Wang et al. [13] proposed an automated method for detecting defects in track fasteners. Initially, they located track fasteners precisely using the background difference method. Then, they extracted linear features from images based on an improved Canny operator and Hough transform. Subsequently, they extracted feature vectors for track fastener defects by combining local binary patterns (LBP) and HOGs. Finally, they employed an SVM to classify the feature vectors. This method demonstrated higher real-time performance and accuracy. Although the aforementioned image processing-based detection methods have improved detection accuracy to some extent, their complex image-processing and feature extraction processes still cannot improve the efficiency of fastener detection.

Detection methods based on deep learning primarily utilize convolutional neural networks (CNNs) to learn features from images. Compared to image processing-based detection methods, they do not require manual feature design, thus offering better robustness. These algorithms can be classified into two major categories: two-stage detection algorithms based on candidate regions and one-stage detection algorithms based on end-to-end learning. The main representatives of two-stage algorithms include R-CNN [14], Fast R-CNN [15], and Faster R-CNN [16]. Wei et al. [2] applied Faster R-CNN to track fastener detection. Despite the improvement in detection accuracy, the issue of slow detection speed persists. The one-stage algorithms are mainly represented by SSD [17] and the YOLO [18–22] series. Compared to other algorithms, the YOLO series algorithms have significant advantages in both detection speed and accuracy. Therefore, they are widely used in track fastener defect detection. Qi et al. [23] proposed an improved MYOLOv3-Tiny network based on YOLOv3. Depth-wise and pointwise convolution were used, and the backbone network was redesigned. The experiments showed that the network achieved higher detection precision and faster detection speed compared to R-CNN. Fu et al. [24] proposed a MobileNet-YOLOv4 algorithm for track fastener detection. This algorithm replaces the CSPDarknet53 feature extraction network in the YOLOv4 algorithm with MobileNet, which enables the extraction of subtle features of track fasteners while reducing the number of parameters and computational complexity, thus improving detection speed. Li et al. [25] proposed an improved track fastener defect detection model based on YOLOv5s. In this model, a convolutional block attention module (CBAM) is added to the Neck network of YOLOv5s to enhance the extraction of key features and suppress irrelevant features. Additionally, a weighted bi-directional feature pyramid network (BiFPN) is introduced to achieve multi-scale feature fusion. The experimental results demonstrate that the improved model enhances both accuracy and detection speed. Wang et al. [26] introduced the CBAM attention mechanism into the backbone network of YOLOv5, replaced the standard convolution blocks in the neck network with GSConv convolution modules, and integrated BiFPN. Finally, they designed a lightweight decoupled head structure to improve detection accuracy and enhance the robustness of the model. The experimental findings testify to the YOLOv5-CGBD model's ability to conduct real-time detection, with mAP0.5 scores of 0.971 and 0.747 for mAP0.5:0.95, surpassing those of the original YOLOv5 model by

2.2% and 4.1%, respectively. Although the above methods can accurately detect fastener defects, there is still room for improvement in terms of false detection rates. Additionally, the detection speed of the models is relatively slow, making it challenging to apply them in engineering practice.

Due to its low cost, high flexibility, and ease of control, UAV-based detection is widely employed across various fields. For instance, in agriculture, it is utilized for tree detection [27,28]; in transportation, it is employed for vehicle tracking [29,30]; in environmental conservation, it is used for inspections [31]; in industry, it is applied for power facility inspections [32]; and in infrastructure, it is employed for bridge crack inspections [33]. In the field of railway inspection, utilizing UAVs can reduce labor costs, improve efficiency, and enhance safety. Wu et al. [34] proposed the use of UAV vision for detecting surface defects on railway tracks. Similarly, Milan et al. [35] suggested the use of UAVs for inspecting railway infrastructure. The development of autonomous UAVs for analyzing data in real time is an emerging trend in UAV data processing [36]. However, current track fastener defect detection models rely on Nvidia graphics cards, which cannot meet the engineering requirements for lightweight, low-power, and real-time devices.

To address the aforementioned issues, the main contributions of this article are as follows:

1. We converted the YOLOv4-tiny model to output single-scale features, which resulted in improved detection speed. Furthermore, we utilized the K-means++ algorithm to re-cluster anchor boxes, thereby improving the model's detection accuracy.
2. We developed the model using an FPGA development platform and deployed the model on the FPGA platform after transformation [37,38], achieving the lightweight, low-power, and real-time requirements of the track fastener defect detection device.

2. Materials and Methods

2.1. YOLOv4-Tiny Algorithm

YOLO is an end-to-end object detection algorithm. It takes the entire image as input, and after processing through a CNN, it yields the localization and classification results of objects. The core of the YOLO detection algorithm involves segmenting the neural network's input image into an $n * n$ grid, where each grid cell has S predefined anchor boxes. Detection results are obtained by applying non-maximum suppression to remove duplicate and ineffective anchor boxes. Additionally, techniques such as residual network structures, feature fusion, and multi-scale output are employed to improve detection capabilities across various scenarios. The loss function of YOLO is represented as Equation (1).

$$LOSS = loss_{loc} + loss_{obj} + loss_{cls} \tag{1}$$

$loss_{loc}$, $loss_{obj}$, and $loss_{cls}$ represent the position regression loss function, the object confidence loss function, and the target classification loss function, respectively.

YOLOv4-tiny is a lightweight version of YOLOv4 proposed by Bochkovskiy et al. In comparison to YOLOv4, YOLOv4-tiny employs a lighter architecture, enabling it to achieve efficient detection speeds even in resource-constrained environments. Therefore, YOLOv4-tiny is better suited for running on embedded devices. The structure of YOLOv4-tiny is shown in Figure 1.

From Figure 1, it can be seen that the backbone network of YOLOv4-tiny consists of Resblocks. The structure of the Resblock is shown in Figure 2. The backbone network outputs multi-scale features, which are then utilized by the YOLO head for detection.

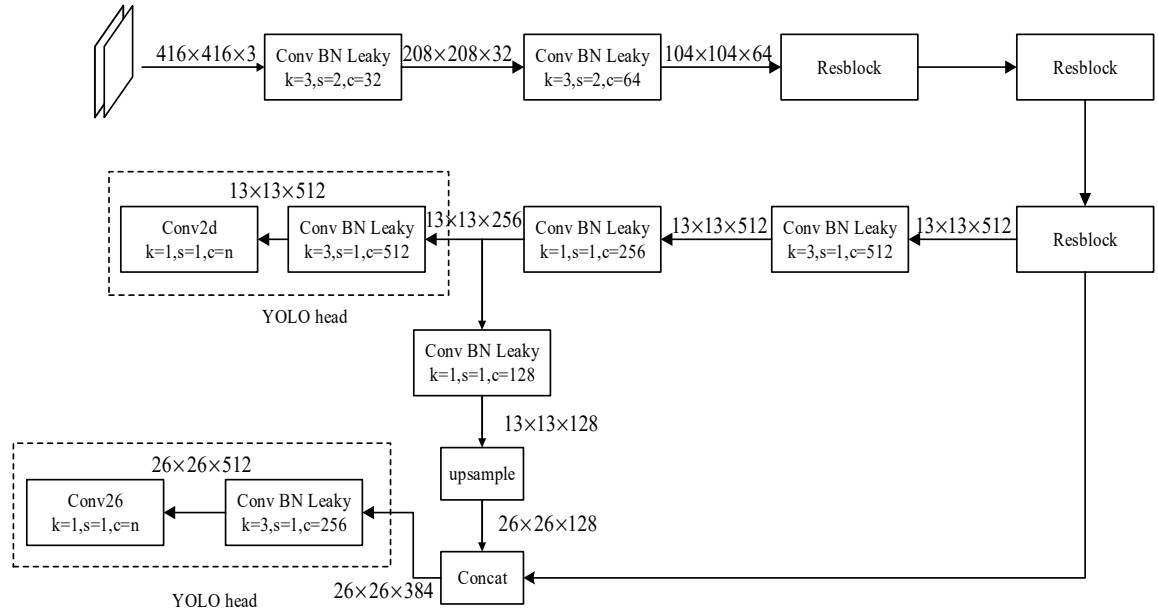

Figure 1. Structure of YOLOv4-tiny.

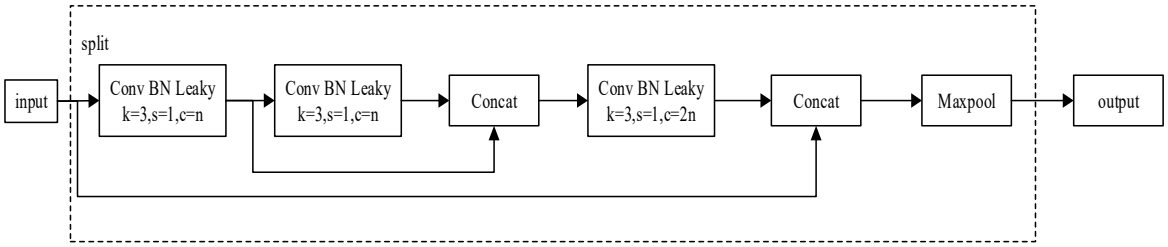

Figure 2. Structure of the Resblock.

In ConvBNLeaky, k represents the size of the convolutional kernel, s represents the stride, and c represents the number of channels. ConvBNLeaky consists of a convolutional layer, a batch normalization layer (BN), and a Leaky activation function. The Leaky function is represented as Equation (2). Compared to the ReLU activation function, during the backpropagation process in deep learning training, the Leaky activation function can still compute gradients for the parts of the input that are less than zero.

$$Leaky(x_i) = \begin{cases} x_i & x_i \geq 0 \\ ax_i & x_i < 0 \end{cases} \tag{2}$$

2.2. YOLOv4-Tiny Improvement

2.2.1. Single-Scale Feature Output

The original YOLOv4-tiny model has two-scale feature outputs. By detecting objects at two scales, YOLOv4-tiny can obtain a more comprehensive understanding of target information and can handle complex scenes more effectively. Compared to other detection tasks such as face detection and vehicle detection, track fastener defect detection tasks typically demonstrate relatively fixed sizes of objects in the image. An image displaying anomalous track fasteners is shown in Figure 3.

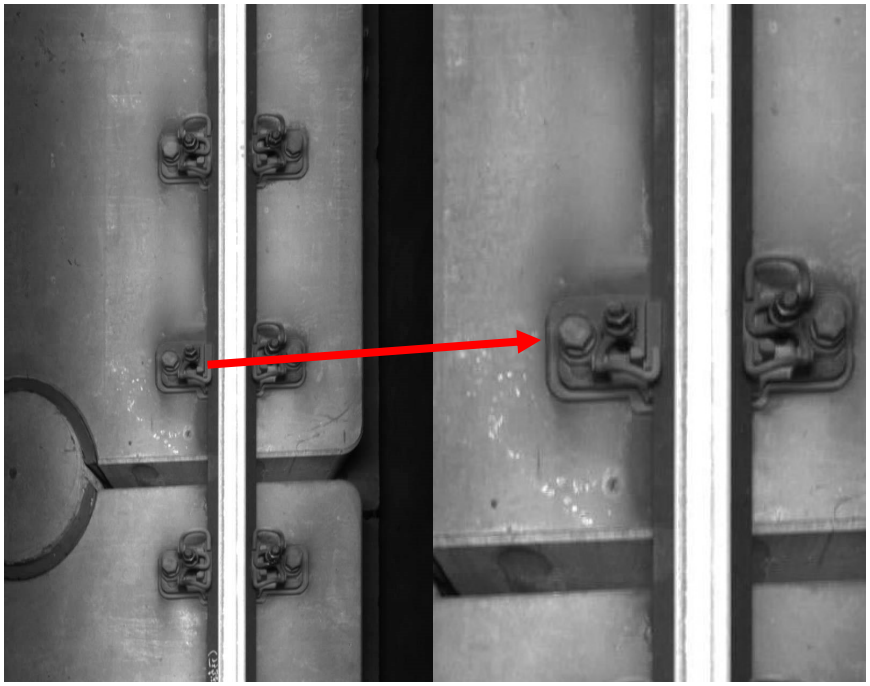

Figure 3. Captured images and abnormal fasteners.

Due to the fixed perspective of the camera and the consistent size of the track fasteners, the size of the detection targets remains fixed relative to the image. To improve the detection speed and reduce computational complexity, we modified YOLOv4-tiny to output features at a single scale. From Figure 3, it can be observed that the track fasteners belong to medium-sized objects. Therefore, we retained features at the scale of (26, 26) while removing features at the scale of (13, 13). The improved model structure is shown in Figure 4.

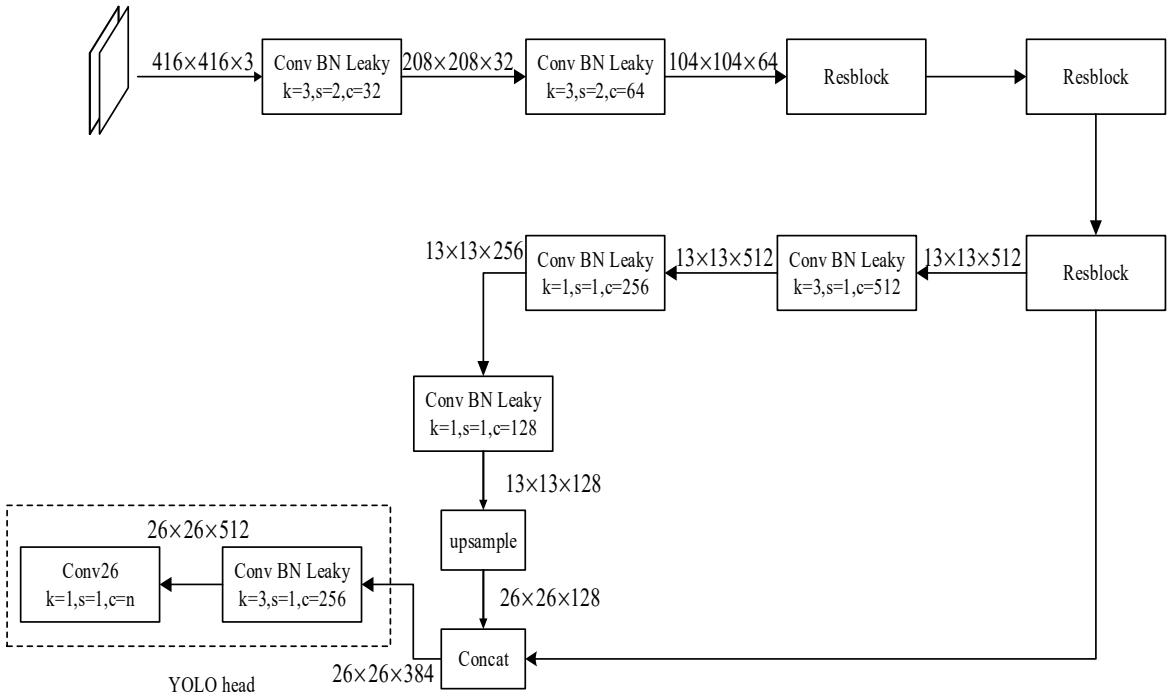

Figure 4. Improved YOLOv4-tiny network structure.

Experiments will be used to validate the hypothesis that the size of the detection targets remains fixed relative to the image.

2.2.2. Anchor Box Optimization

The original YOLOv4-tiny model has problems such as inaccurate localization in track fastener defect detection tasks due to its utilization of default anchor boxes. The default anchor boxes in YOLOv4-tiny are generated through clustering analysis conducted on the COCO dataset, which predominantly consists of object categories commonly encountered in everyday scenarios. Consequently, the anchor boxes obtained from this process may not be optimally tailored for the track fasteners, leading to issues such as inaccurate localization in detection tasks.

To improve the accuracy of the model in detecting track fasteners, this study obtained new anchor box parameters from proprietary datasets. We employed the K-means++ algorithm to conduct clustering analysis on the fastener dataset to obtain new anchor box parameters. Compared to the K-means algorithm, the K-means++ algorithm optimizes the selection of initial cluster centers by maximizing the distance between K initial cluster centers as much as possible, effectively improving clustering efficiency. The steps of the K-means++ algorithm are as follows:

Step 1: Randomly select a sample from dataset N as the first cluster center.

Step 2: Compute the distance $D(x)$ from each sample x to the nearest existing cluster center and calculate the probability $P(x)$ of each sample being identified as the next cluster center using the following formula:

$$IoU = \frac{A \cap B}{A \cup B} \tag{3}$$

$$D(x) = 1 - IoU \tag{4}$$

$$P(x) = \frac{D(x)^2}{\sum_{x \in N} D(x)^2} \tag{5}$$

where IoU denotes the degree of matching between the anchor box and the labeled box. Select the sample with the maximum value of $P(x)$ as the next cluster center.

Step 3: Repeat step 2 until k cluster centers have been selected.

Step 4: Utilize the K-means algorithm to obtain new anchor box parameters.

The original and the new parameters of the anchor boxes are shown in Table 1.

Table 1. The original and the new parameters of the anchor boxes.

Algorithm	Anchor Box
YOLOv4-tiny	[(10, 14) (23, 27) (37, 58)] [(81, 82) (135, 169) (344, 319)]
K-means++	[(25, 27) (34, 48) (55, 73)]

2.3. Hardware Platforms

2.3.1. Comparison of Hardware Platforms

Currently, almost all deep learning algorithms run on GPUs. This is because GPUs offer powerful computational capabilities, and frameworks like PyTorch provide convenience for researchers in their studies. However, the powerful computational capabilities of GPUs also come with drawbacks such as large size and high power consumption. As mentioned earlier, we plan to utilize UAVs for track fastener defect detection. Therefore, we require a lightweight, low-power, high-performance real-time computing platform.

We have noticed that an increasing number of researchers are choosing field-programmable gate arrays (FPGAs) as deployment platforms for deep learning models. Compared to GPUs, FPGAs offer advantages such as programmability, flexibility, and low power con-

sumption. With Xilinx's introduction of the deep learning processing unit (DPU), FPGAs can also provide high-performance inference for deep learning models.

In conclusion, FPGAs were chosen as the hardware platform for our algorithm.

2.3.2. The ZCU104 Development Platform

The hardware development platform used in this paper is the Zynq Ultrascale+ MPSoC ZCU104 development platform from Xilinx. The ZCU104 development board is shown in Figure 5. The ZCU104 is a high-performance development platform suitable for various embedded systems and application scenarios, such as artificial intelligence, video processing, network communication, and industrial control. The ZCU104 platform employs the ZU7EV chip, which includes a quad-core ARM Cortex™-A53 processor and a dual-core Cortex-R5 processor, with a CPU frequency of 1200 MHz.

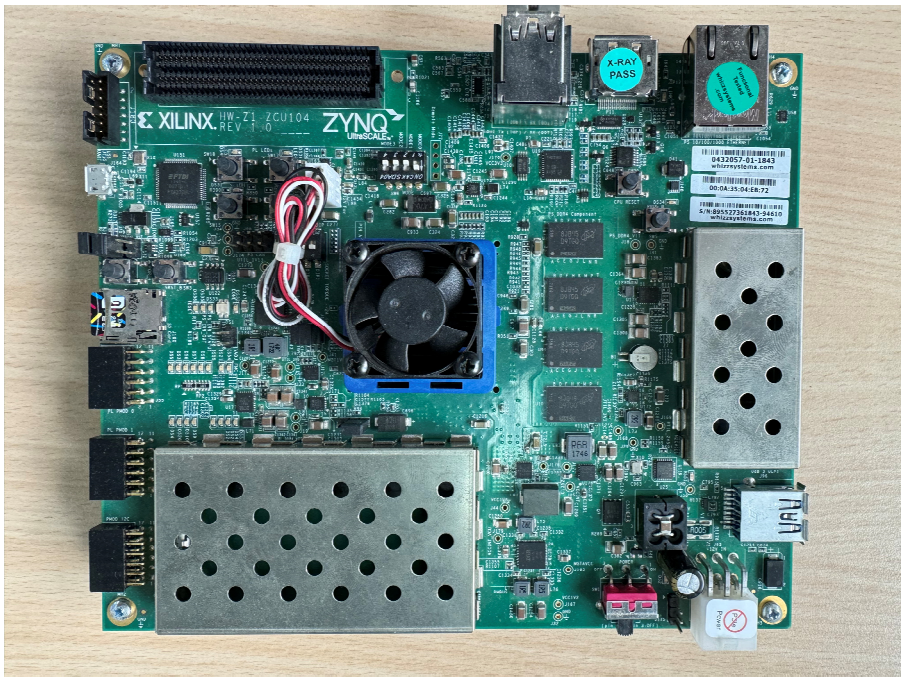

Figure 5. The ZCU104 development board.

The hardware development platform resources also include 312 Block RAMs (BRAMs) for data storage, 1728 digital signal processors (DSPs) for digital signal processing and algorithm acceleration, 230,400 look-up tables (LUTs) for executing logic operations, and 460,800 flip-flops (FFs) for storing state information.

2.3.3. Hardware Platform Development

The development process on the ZCU104 platform is shown in Figure 6.

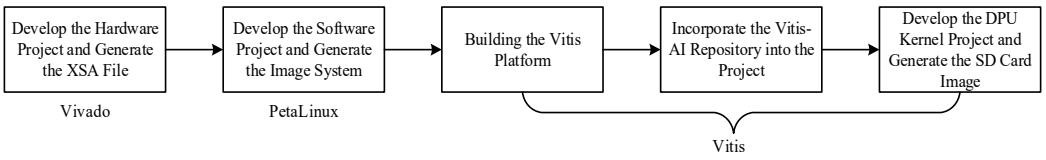

Figure 6. The development process on the ZCU104 platform.

The hardware project was developed on the Vivado 2021.1 platform. We utilized the Zynq UltraScale + MPSoC IP module from the Zynq series, alongside the board preset for ZCU104. This IP module is shown in Figure 7. After configuring all of the ports, an XSA file was generated. The XSA file contained all of the hardware information.

Figure 7. ZCU104 IP module.

PetaLinux is a specialized development platform designed for embedded Linux system development, introduced by Xilinx. This platform enables the configuration of the Linux kernel, device tree, and root file system (rootfs). We utilized PetaLinux 2021.2 and the XSA file to generate a Linux image system that incorporates the required dependency library files. Subsequent development will be based on this customized Linux system.

Vitis is a software development platform introduced by Xilinx, designed to simplify the software development process on FPGAs. It offers a unified software development environment. After building the Vitis platform and incorporating the Vitis-AI repository, we developed the DPU kernel project. The DPU is a hardware accelerator dedicated to deep learning inference tasks, introduced by Xilinx. Its primary objective is to expedite the inference computations of deep learning models, including convolutional neural networks (CNNs). The top-level architecture of the DPU is shown in Figure 8.

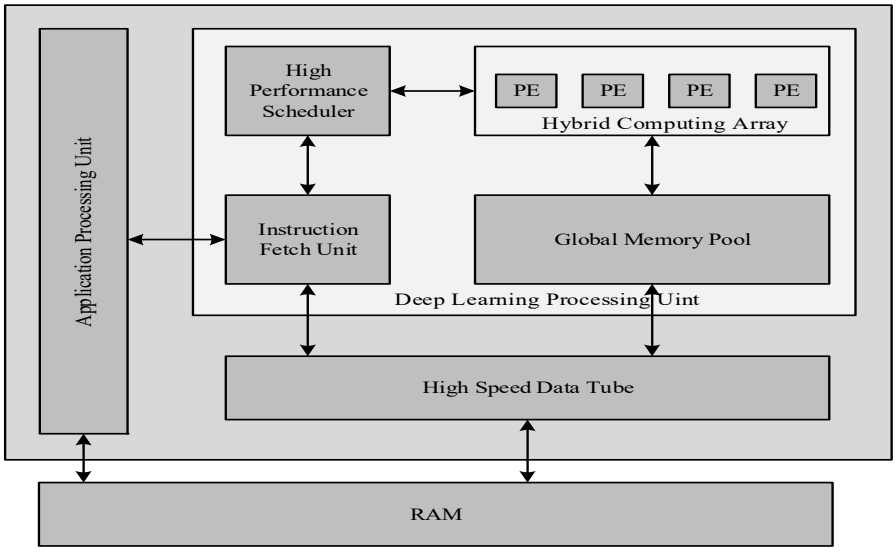

Figure 8. The top-level architecture of the DPU.

A DPU module with model number B4096 was utilized in this paper. With UltraRAM enabled, the ZCU104 platform supports a maximum of two B4096 modules.

Finally, we packaged all of the files into an SD card image.

2.4. Model Transformation

After completing hardware platform development, it is necessary to transform the trained network model to enable forward inference on the hardware platform. The process of model transformation is shown in Figure 9.

Vitis-AI is a development platform aimed at AI acceleration, introduced by Xilinx. It offers a comprehensive set of tools and libraries to assist developers in converting various deep learning models into formats suitable for FPGA deployment, while also accelerating deep learning inference tasks. Following the process shown in Figure 9, we utilized the Vitis-AI 1.4 platform to quantize and compile our model, transforming it into an Xmodel file executable on the hardware platform.

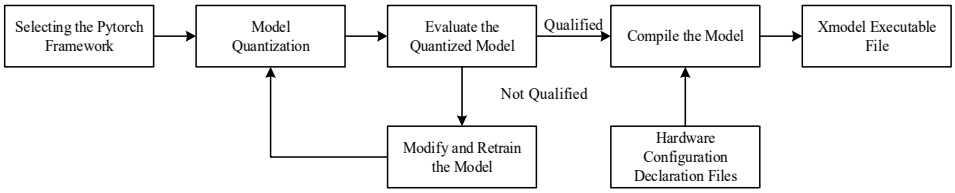

Figure 9. The process of model transformation.

3. Experimental Results

3.1. Dataset

The dataset used in this paper was collected using a line laser camera, comprising a total of 2000 images with a resolution of 800 × 1261 pixels. After removing distorted and blurry images, 1100 images were obtained. Each image contained approximately 6 to 12 fasteners, with a total of approximately 8000 fasteners. Approximately 200 images contained abnormal fasteners. The dataset comprises two distinct categories of fasteners, denoted Class A and Class B, alongside their respective abnormal counterparts, labeled Class A–F and Class B–F.

3.2. Experimental Setting

The experimental environment was configured with Windows 10 as the operating system, NVIDIA GeForce RTX 3070 as the GPU model with 8 G of video memory, Python 3.8 as the compilation language, Pytorch 1.8.0 as the deep learning framework, CUDA 10.2 as the CUDA version, and ZCU104 development platform as the hardware platform. The training parameters were set as follows: the initial learning rate was 0.001, the momentum parameter was 0.9, the weight decay factor was 0.0005, the input image size was 416 × 416, and the Batch Size was 16. A total of 200 epochs were trained using stochastic gradient descent (SGD) for the whole training process.

Figure 10 shows the comparison of loss curves, where the red curve represents the loss curve of the YOLOv4-tiny algorithm, and the blue curve represents the loss curve of the Improved YOLOv4-tiny algorithm. Lower loss values during training indicate better training results. From Figure 10, it can be observed that the loss value of the Improved YOLOv4-tiny algorithm is lower than that of the YOLOv4-tiny algorithm after 75 epochs.

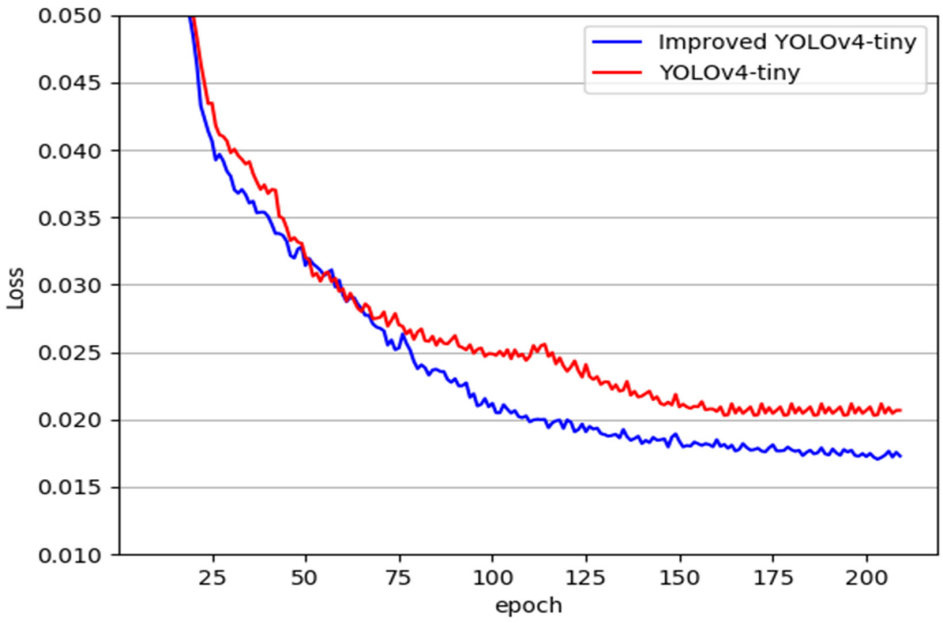

Figure 10. Comparison of loss curves.

3.3. Evaluation Indicators

In this paper, we evaluated the performance of the algorithms according to two aspects, detection accuracy and detection speed, using evaluation methods commonly employed for target detection algorithms.

Detection accuracy evaluation metrics comprise recall, false detection, and mean average precision (mAP).

In the field of object detection, TP denotes the number of positive samples detected correctly, FP denotes the number of positive samples detected incorrectly, TN denotes the number of negative samples detected correctly, and FN denotes the number of negative samples detected incorrectly. All of the metrics are calculated when $IoU = 0.5$.

Recall, denoted by R, is the probability that the model correctly identifies a positive sample in a single category. It is defined as follows:

$$R = \frac{TP}{TP + FN} \tag{6}$$

False detection, denoted by FPR, is the probability that the model incorrectly identifies a negative sample as a positive sample in a single category. It is defined as follows:

$$FPR = \frac{FP}{FP + TN} \tag{7}$$

High R and low FPR are required for the task of track fastener defect detection.

The mAP is the area enclosed by the precision and recall curves. It is an overall network performance evaluation metric considering precision and recall [39]. Therefore, mAP is a more authoritative metric in model performance evaluation, and a larger mAP value represents higher detection precision. It is defined as follows:

$$AP = \int_0^1 P \cdot R dR \tag{8}$$

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{9}$$

where N is the number of categories in the dataset and P is precision, which denotes the probability that the model detects correctly in a single category. It is defined as follows:

$$P = \frac{TP}{TP + FP} \tag{10}$$

The detection speed is evaluated in terms of frames per second (FPS), the number of frames per second that the model processes for the image.

3.4. Experiments on the GPU

3.4.1. Ablation Experiments

We conducted ablation experiments for the two improvements proposed in this paper. We modified YOLOv4-tiny to output single-scale feature maps and then replaced the original anchor boxes with anchor boxes optimized using the K-means++ algorithm. The final results are shown in Table 2.

The experimental results indicate that single-scale feature output can significantly improve detection speed, increasing from 316.3 FPS to 521.9 FPS. However, there was a decrease in mAP by 1.8%, and the FPR for each class of fasteners also increased.

Table 2. Comparison of the results of the ablation experiments.

Network Model	Class	R/%	FPR/%	mAP/%	FPS
YOLOv4-tiny	A	75.0	12.9	90.5	336.3
	B	91.7	0		
YOLOv4-tiny + Single-Scale Feature Output	A	72.6	13.3	88.7	554.9
	B	89.8	0.9		
YOLOv4-tiny + Single-Scale Feature Output + Anchor Box Optimization (Improved YOLOv4-tiny)	A	86.1	3.1	95.8	554.9
	B	100	0		

The experimental results also indicate that optimizing the anchor boxes significantly improves detection performance. The Improved YOLOv4-tiny model achieved a 5.3% increase in mAP compared to YOLOv4-tiny. This validates our previous hypothesis that the relative size of detected objects remains consistent with respect to the image.

3.4.2. Comparison Experiments

To further validate the detection performance of the improved model, the model in this paper was compared with the existing mainstream target detection algorithms Faster R-CNN and SSD. The experiments were conducted in the same hardware and software environment, as well as with identical training and testing parameters. The experimental results are shown in Table 3.

Table 3. Performance comparison of various target detection algorithms.

Network Model	Class	R/%	FPR/%	mAP/%	FPS
Faster R-CNN	A	77.5	13.6	90.6	10.7
	B	88.3	0		
SSD	A	82.0	5.8	93.4	30.3
	B	99.5	0		
YOLOv4-tiny	A	75.0	12.9	90.5	336.3
	B	91.7	0		
Improved YOLOv4-tiny	A	86.1	3.1	95.8	554.9
	B	100	0		

This can be seen based on the data in Table 3. In terms of the mAP metric, Improved YOLOv4-tiny achieved the best performance, reaching 95.8%. Compared to Faster R-CNN, SSD, and YOLOv4-tiny, the improved model's mAP increased by 5.2%, 2.4%, and 5.3%, respectively. In terms of the false positive rate (FPR) metric, Improved YOLOv4-tiny also achieved the best performance, with a rate of 3.1% for Class A fasteners. This is lower than the rates achieved by Faster R-CNN, SSD, and YOLOv4-tiny, which were 13.6%, 5.8%, and 12.9%, respectively. In terms of detection speed, Improved YOLOv4-tiny also achieved the best performance, reaching 554.9 FPS. Compared to Faster R-CNN, SSD, and YOLOv4-tiny, the improved model's speed increased by 544.2 FPS, 524.6 FPS, and 218.6 FPS, respectively. In summary, compared to other target detection models, the Improved YOLOv4-tiny model in this paper outperformed in terms of detection accuracy, detection error rate, and detection speed.

3.5. Experiments on the FPGA

To validate the detection performance of our algorithm on the ZCU104 development platform, we compared both the improved model and the original model after transformation. In the experiment, we utilized two DPU modules with the model number B4096 and set their frequency to 300 MHz. Furthermore, we attempted to improve the detection speed by employing parallel processing techniques. The experimental results are shown in Table 4.

Table 4. Performance comparison of algorithms on the FPGA.

Network Model	Class	R/%	FPR/%	mAP/%	Thread	FPS
YOLOv4-tiny	A	75.0	6.9	89.5	1	70.9
	B	87.5	0		8	179.6
Improved YOLOv4-tiny	A	83.3	3.23	95.1	1	84.2
	B	100	0		8	295.9

This can be seen based on the data in Table 4. In terms of the mAP metric, the Improved YOLOv4-tiny achieved 95.1% on the FPGA, which was 5.6% higher than the original YOLOv4-tiny. In terms of detection speed, the Improved YOLOv4-tiny achieved 295.9 FPS on the FPGA, which was 116.3 FPS higher than the original YOLOv4-tiny. Additionally, parallel processing significantly improved detection efficiency compared to single-thread processing.

3.6. Experimental Comparison of Different Platforms

To validate the FPGA platform as more suitable for practical engineering, we compared the results across the two hardware platforms. The experimental results are shown in Table 5.

Table 5. Performance comparison of the algorithm on the GPU and FPGA.

Network Model	Hardware Platform	mAP/%	FPS	Power Consumption/W
Improved YOLOv4-tiny	GeForce RTX 3070	95.8	554.9	235
Improved YOLOv4-tiny	ZCU104	95.1	295.9	20

This can be seen based on the data in Table 5. In terms of the mAP metric, the performance of the improved model on the FPGA was 0.7% lower than on the GPU. This is attributed to the quantization of model parameters during the model transformation process, resulting in slight precision loss. However, an mAP of 95.1% still meets engineering requirements. In terms of detection speed, the improved model achieved 295.9 FPS on the FPGA, which was lower than the 554.9 FPS achieved on the GPU. However, it still met the real-time requirements in practical engineering applications. In terms of power consumption, the improved model's power consumption on the FPGA was 25 W, significantly lower than the 235 W on the GPU.

In summary, the improved model achieved sufficiently good detection accuracy and speed on the FPGA with very low power consumption. Compared to the GPU platform, the FPGA platform is more suitable for meeting the requirements of track fastener defect detection tasks.

3.7. Visualization of Detection Results

The visualization of detection results is important in practical applications. Figure 11 shows the detection results of the improved model. It can be observed from Figure 11 that the algorithm exhibits excellent recognition performance for both types of fasteners, with the bounding boxes' positions and sizes matching the actual objects. Additionally,

the confidence scores for both positive and negative samples are very high, meeting the requirements of practical detection.

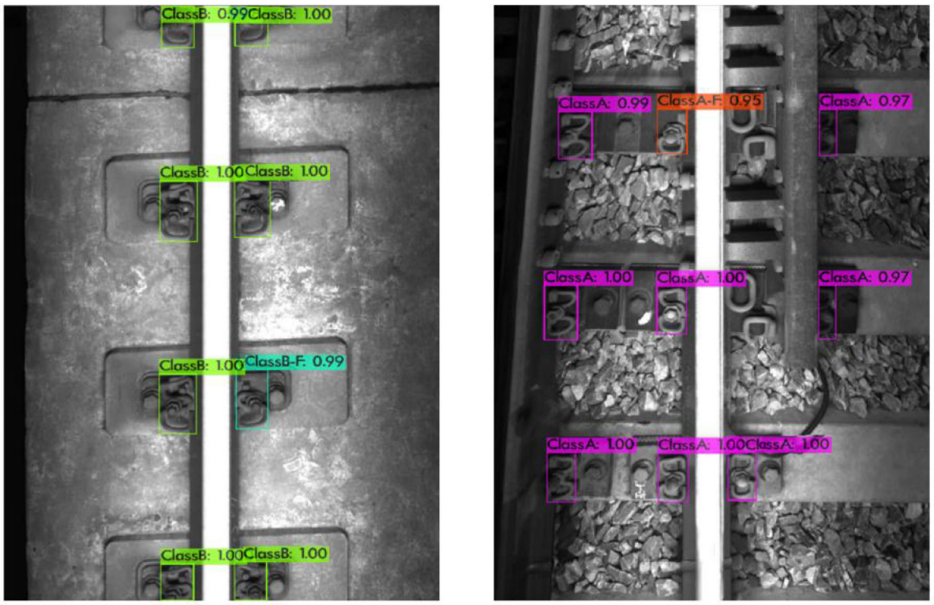

Figure 11. Visualization of detection results.

4. Conclusions

This paper addresses the issues of low efficiency in current track fastener detection algorithms and the lack of lightweight and low-power hardware platforms suitable for practical engineering applications. We constructed our own dataset of track fasteners, proposed an improved model based on YOLOv4-tiny, and deployed the transformed model on an FPGA hardware platform. Considering the dataset characteristics, the model was improved to achieve single-scale feature output, significantly enhancing detection speed. Additionally, to improve detection accuracy, we employed the K-means++ algorithm to cluster the dataset and obtain more suitable anchor boxes. Finally, we developed and deployed the model on the FPGA platform. The experimental results demonstrate that the improved model achieves an mAP of 95.1% and a speed of 295.9 FPS on the FPGA, surpassing the performance of the original YOLOv4-tiny. Moreover, the power consumption of the FPGA platform is 20 W, much lower than that of the GPU platform, meeting the requirements for UAVs carrying detection equipment.

Our improved model has been specifically designed for the fastener types in this dataset and may not be suitable for recognizing other types of fasteners. Additionally, considering the support provided by the Vitis development tools for YOLOv4 series algorithms on the ZCU104 platform, we chose to improve YOLOv4-tiny instead of applying the latest YOLO versions. In the future, we plan to expand the dataset scope and explore the application potential of the latest algorithms on FPGA platforms to optimize and extend our model, making it adaptable to a wider range of fastener types and detection environments.

Author Contributions: Conceptualization, Q.Y. and A.L.; methodology, Q.Y.; investigation, Q.Y. and A.L.; writing—original draft, Q.Y.; writing—review and editing, X.Y. and W.D.; supervision, X.Y. and W.D.; project administration, Q.Y., A.L., X.Y. and W.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: You can use this link to download the dataset: https://github.com/Yuqi1998/FastenerDataset (accessed on 26 April 2024).

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

References

1. Xiang, J.; Yuan, C.; Yu, C.; Lin, S.; Yang, H. Analysis of Elastic Bar Fracture Causes of Fasteners in Ballastless Track of High-Speed Railway. *J. Rail Way Sci. Eng.* **2019**, *16*, 1605–1613.
2. Wei, X.; Yang, Z.; Liu, Y.; Wei, D.; Jia, L.; Li, Y. Railway Track Fastener Defect Detection Based on Image Processing and Deep Learning Techniques: A Comparative Study. *Eng. Appl. Artif. Intell.* **2019**, *80*, 66–81. [CrossRef]
3. Bono, F.M.; Radicioni, L.; Cinquemani, S.; Benedetti, L.; Cazzulani, G.; Somaschini, C.; Belloli, M. A Deep Learning Approach to Detect Failures in Bridges Based on the Coherence of Signals. *Future Internet* **2023**, *15*, 119. [CrossRef]
4. Chellaswamy, C.; Krishnasamy, M.; Balaji, L.; Dhanalakshmi, A.; Ramesh, R. Optimized Railway Track Health Monitoring System Based on Dynamic Differential Evolution Algorithm. *Measurement* **2020**, *152*, 107332. [CrossRef]
5. Zhan, Z.; Sun, H.; Yu, X.; Yu, J.; Zhao, Y.; Sha, X.; Chen, Y.; Huang, Q.; Li, W.J. Wireless Rail Fastener Looseness Detection Based on MEMS Accelerometer and Vibration Entropy. *IEEE Sens. J.* **2020**, *20*, 3226–3234. [CrossRef]
6. Mao, Q.; Cui, H.; Hu, Q.; Ren, X. A Rigorous Fastener Inspection Approach for High-Speed Railway from Structured Light Sensors. *ISPRS J. Photogramm. Remote Sens.* **2018**, *143*, 249–267. [CrossRef]
7. Damljanović, V.; Weaver, R.L. Laser Vibrometry Technique for Measurement of Contained Stress in Railroad Rail. *J. Sound Vib.* **2005**, *282*, 341–366. [CrossRef]
8. Guerrieri, M.; Parla, G.; Celauro, C. Digital Image Analysis Technique for Measuring Railway Track Defects and Ballast Gradation. *Measurement* **2018**, *113*, 137–147. [CrossRef]
9. Khan, R.A.; Islam, S.; Biswas, R. Automatic Detection of Defective Rail Anchors. In Proceedings of the 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), Qingdao, China, 8–11 October 2014; pp. 1583–1588.
10. Feng, H.; Jiang, Z.; Xie, F.; Yang, P.; Shi, J.; Chen, L. Automatic Fastener Classification and Defect Detection in Vision-Based Railway Inspection Systems. *IEEE Trans. Instrum. Meas.* **2014**, *63*, 877–888. [CrossRef]
11. Gibert, X.; Patel, V.M.; Chellappa, R. Sequential Score Adaptation with Extreme Value Theory for Robust Railway Track Inspection. *arXiv* **2015**, arXiv:1510.05822.
12. Gibert, X.; Patel, V.M.; Chellappa, R. Deep Multitask Learning for Railway Track Inspection. *IEEE Trans. Intell. Transp. Syst.* **2017**, *18*, 153–164. [CrossRef]
13. Wang, Z.; Wang, S. *Research of Method for Detection of Rail Fastener Defects Based on Machine Vision*; Atlantis Press: Amstelkade, The Netherland, 2015.
14. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *arXiv* **2014**, arXiv:1311.2524.
15. Girshick, R. Fast R-CNN. *arXiv* **2015**, arXiv:1504.08083.
16. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [CrossRef] [PubMed]
17. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; Berg, A.C. SSD: Single Shot MultiBox Detector. In *Proceedings of the Computer Vision—ECCV 2016, Amsterdam, The Netherlands, 11–14 October 2016*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 21–37.
18. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. *arXiv* **2016**, arXiv:1506.02640.
19. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 7263–7271.
20. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:1804.02767.
21. Bochkovskiy, A.; Wang, C.-Y.; Liao, H.-Y.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv* **2020**, arXiv:2004.10934.
22. Wang, C.-Y.; Bochkovskiy, A.; Liao, H.-Y.M. YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors. *arXiv* **2023**, arXiv:2207.02696.
23. Qi, H.; Xu, T.; Wang, G.; Cheng, Y.; Chen, C. MYOLOv3-Tiny: A New Convolutional Neural Network Architecture for Real-Time Detection of Track Fasteners. *Comput. Ind.* **2020**, *123*, 103303. [CrossRef]
24. Fu, J.; Chen, X.; Lv, Z. Rail Fastener Status Detection Based on MobileNet-YOLOv4. *Electronics* **2022**, *11*, 3677. [CrossRef]
25. Li, X.; Wang, Q.; Yang, X.; Wang, K.; Zhang, H. Track Fastener Defect Detection Model Based on Improved YOLOv5s. *Sensors* **2023**, *23*, 6457. [CrossRef] [PubMed]
26. Wang, L.; Zang, Q.; Zhang, K.; Wu, L. A Rail Fastener Defect Detection Algorithm Based on Improved YOLOv5. *Proc. Inst. Mech. Eng. Part F J. Rail Rapid Transit* **2024**, 09544097241234380. [CrossRef]
27. Qin, Z.; Wang, W.; Dammer, K.-H.; Guo, L.; Cao, Z. Ag-YOLO: A Real-Time Low-Cost Detector for Precise Spraying with Case Study of Palms. *Front. Plant Sci.* **2021**, *12*, 753603. [CrossRef] [PubMed]
28. Han, P.; Ma, C.; Chen, J.; Chen, L.; Bu, S.; Xu, S.; Zhao, Y.; Zhang, C.; Hagino, T. Fast Tree Detection and Counting on UAVs for Sequential Aerial Images with Generating Orthophoto Mosaicing. *Remote Sens.* **2022**, *14*, 4113. [CrossRef]
29. Tilon, S.; Nex, F.; Vosselman, G.; Sevilla de la Llave, I.; Kerle, N. Towards Improved Unmanned Aerial Vehicle Edge Intelligence: A Road Infrastructure Monitoring Case Study. *Remote Sens.* **2022**, *14*, 4008. [CrossRef]
30. Balamuralidhar, N.; Tilon, S.; Nex, F. MultEYE: Monitoring System for Real-Time Vehicle Detection, Tracking and Speed Estimation from UAV Imagery on Edge-Computing Platforms. *Remote Sens.* **2021**, *13*, 573. [CrossRef]

31. Luo, W.; Han, W.; Fu, P.; Wang, H.; Zhao, Y.; Liu, K.; Liu, Y.; Zhao, Z.; Zhu, M.; Xu, R.; et al. A Water Surface Contaminants Monitoring Method Based on Airborne Depth Reasoning. *Processes* **2022**, *10*, 131. [CrossRef]
32. Liu, C.; Liu, Y.; Wu, H.; Dong, R. A Safe Flight Approach of the UAV in the Electrical Line Inspection. *Int. J. Emerg. Electr. Power Syst.* **2015**, *16*, 503–515. [CrossRef]
33. Rau, J.Y.; Hsiao, K.W.; Jhan, J.P.; Wang, S.H.; Fang, W.C.; Wang, J.L. Bridge Crack Detection Using Multi-Rotary UAV and Object-Base Image Analysis. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2017**, *XLII-2-W6*, 311–318. [CrossRef]
34. Wu, Y.; Qin, Y.; Wang, Z.; Jia, L. A UAV-Based Visual Inspection Method for Rail Surface Defects. *Appl. Sci.* **2018**, *8*, 1028. [CrossRef]
35. Banić, M.; Miltenović, A.; Pavlović, M.; Ćirić, I. Intelligent Machine Vision Based Railway Infrastructure Inspection and Monitoring Using UAV. *Facta Univ. Ser. Mech. Eng.* **2019**, *17*, 357–364. [CrossRef]
36. Nex, F.; Armenakis, C.; Cramer, M.; Cucci, D.A.; Gerke, M.; Honkavaara, E.; Kukko, A.; Persello, C.; Skaloud, J. UAV in the Advent of the Twenties: Where We Stand and What Is Next. *ISPRS J. Photogramm. Remote Sens.* **2022**, *184*, 215–242. [CrossRef]
37. Zhu, J.; Wang, L.; Liu, H.; Tian, S.; Deng, Q.; Li, J. An Efficient Task Assignment Framework to Accelerate DPU-Based Convolutional Neural Network Inference on FPGAs. *IEEE Access* **2020**, *8*, 83224–83237. [CrossRef]
38. Dobai, R.; Sekanina, L. Towards Evolvable Systems Based on the Xilinx Zynq Platform. In Proceedings of the 2013 IEEE International Conference on Evolvable Systems (ICES), Singapore, 16–19 April 2013; pp. 89–95.
39. Padilla, R.; Netto, S.L.; da Silva, E.A.B. A Survey on Performance Metrics for Object-Detection Algorithms. In Proceedings of the 2020 International Conference on Systems, Signals and Image Processing (IWSSIP), Niteroi, Brazil, 1–3 July 2020; pp. 237–242.