symmetry

MDPI

*Article*

# YOLO-RDP: Lightweight Steel Defect Detection through Improved YOLOv7-Tiny and Model Pruning

**Guiheng Zhang, Shuxian Liu *, Shuaiqi Nie and Libo Yun**

School of Computer Science and Technology, Xinjiang University, Urumqi 830046, China; 107552204118@stu.xju.edu.cn (G.Z.); 107552204054@stu.xju.edu.cn (S.N.); ylb@stu.xju.edu.cn (L.Y.)
* Correspondence: liushuxian@xju.edu.cn

**Abstract:** During steel manufacturing, surface defects such as scratches, scale, and oxidation can compromise product quality and safety. Detecting these defects accurately is critical for production efficiency and product integrity. However, current target detection algorithms are often too resource-intensive for deployment on edge devices with limited computing resources. To address this challenge, we propose YOLO-RDP, an enhanced YOLOv7-tiny model. YOLO-RDP integrates RexNet, a lightweight network, for feature extraction, and employs GSConv and VOV-GSCSP modules to enhance the network's neck layer, reducing parameter count and computational complexity. Additionally, we designed a dual-headed object detection head called DdyHead with a symmetric structure, composed of two complementary object detection heads, greatly enhancing the model's ability to recognize minor defects. Further model optimization through pruning achieves additional lightweighting. Experimental results demonstrate the superiority of our model, with improvements in mAP values of 3.7% and 3.5% on the NEU-DET and GC10-DET datasets, respectively, alongside reductions in parameter count and computation by 40% and 30%, and 25% and 24%, respectively.

**Keywords:** YOLOv7-tiny; RexNet; DdyHead; channel pruning algorithm



**Citation:** Zhang, G.; Liu, S.; Nie, S.; Yun, L. YOLO-RDP: Lightweight Steel Defect Detection through Improved YOLOv7-Tiny and Model Pruning. *Symmetry* **2024**, *16*, 458. https://doi.org/10.3390/sym16040458

Academic Editor: Dalibor Štys

Received: 29 February 2024
Revised: 3 April 2024
Accepted: 8 April 2024
Published: 10 April 2024



## 1. Introduction

Steel is extensively used across various sectors, from construction and bridges to automotive and aerospace, and even in the manufacturing of household goods. However, during the production process, it is influenced by various factors. During the production process, due to various reasons such as the continuous casting of steel billets, processing technology, and production environment, the surface of steel often sustains different types of defects such as cracks, scratches, folds, and holes. These defects not only affect the appearance but also lead to stress concentration, reducing the fatigue strength and impact resistance of the steel, thus affecting its service life. The production of defective steel results in a large amount of waste of raw materials, greatly affecting the profitability of enterprises. How to control the production rate of defective steel, improve the level of product quality, and meet the demands of modern industry is a problem that must be solved at present.

Traditional methods for detecting surface defects in steel mainly include manual inspection, photoelectric detection, and detection methods based on traditional machine vision. Manual inspection is inefficient and labor-intensive, with problems such as inconsistent inspection standards and missed detections. Although photoelectric detection has improved efficiency to some extent, it has strict requirements regarding the environment and high equipment maintenance costs [1–4]. Detection methods based on traditional machine vision have significantly improved in detection speed and accuracy, but they require manual segmentation and feature extraction of steel images, which demand high technical expertise from operators and computational capabilities from computers.

In recent years, artificial intelligence, especially deep learning technology, has rapidly emerged, providing a more efficient solution for surface defect detection in steel. Target

detection algorithms based on deep learning methods are continuously being tried and applied to steel defect detection. Li [5] studied deep learning models in target detection and proposed a YOLO network-based single-stage target detection method for detecting surface defects on steel plates. Cheng [6] proposed to enhance the feature extraction and localization capabilities of steel defects by adding new layers and the DIoU bounding box regression enhancement algorithm based on YOLOv3. Wang [7] improved the detection accuracy of steel defects by utilizing the de-weighted BiFPN structure based on YOLOv7, fully utilizing feature information, strengthening feature fusion, reducing the loss of feature information during the convolution process, and combining the attention mechanism of ECA in the Backbone part to strengthen important feature channels. However, the large number of parameters and computational complexity of this model make it difficult to deploy on edge terminal devices.

Although target detection algorithms based on deep learning have achieved good results in many fields, there are several issues when directly applied to steel defect detection:

- Large model size and computational complexity: For deployment on edge terminal devices with limited computing power in steel plants, excessively large models and computational complexity can lead to device overload, making it impossible to detect targets;
- Steel surface defects are small targets and are easily overlooked during feature learning, leading to missed detections. Although the YOLO algorithm is known for its excellent performance and balance between accuracy and speed, detecting small targets has always been a challenge for the YOLO series of target detection algorithms.

YOLOv7-tiny sacrifices a certain degree of accuracy compared to YOLOv7, but it has advantages in speed and lightweighting, making it more suitable for deployment on edge terminal devices. Therefore, this study chooses YOLOv7-tiny as the baseline model for steel defect detection. However, it has the following shortcomings:

- The extensive use of ELAN networks in the Backbone, where each ELAN network consists of multiple densely connected standard convolutions, results in a complex network structure, excessive computational complexity, and a large number of parameters. Moreover, the number of network layers is too few, which is not conducive to feature extraction;
- ELAN networks are still used in the Neck section, making it easier to generate redundant features during feature aggregation;
- In the Head section, processing target position and category information together leads to excessive parameter size and computational complexity. Additionally, the lack of multi-level perception of feature information makes it difficult to improve detection performance.

To address these issues, this paper considers a more lightweight solution that reduces parameter size and computational complexity while improving detection accuracy. The main contributions of this study are as follows:

(1) Utilization of the lightweight network RexNet [8] for improved feature extraction in the model, reducing both parameter count and computational load;
(2) Enhancement of the Neck section with lightweight modules GSConv [9] and VoVGSCSP [9], replacing standard convolution with GSConv to mitigate the negative impacts of DSC operations in lightweight models while leveraging DSC's advantages. This reduces model complexity and maintains accuracy, and using VoVGSCSP instead of ELAN lowers computational complexity, fitting the limited resources of edge devices;
(3) Improvement of the original model's detection head with an attention-enhanced detector, DdyHead, enhancing the model's ability to recognize minor defects;
(4) Further model compression through channel-level pruning algorithms without compromising accuracy. The improved YOLO-RDP model significantly enhances parameter efficiency, computational complexity, and model size, improving accuracy over

the original YOLOv7-tiny model, and achieving a balance between precision and lightweighting suitable for deployment on edge devices.

## 2. Related Work

### *2.1. YOLOv7-Tiny Network Structure*

In 2020, Bochkovskiy [10] introduced the YOLOv7 algorithm, an advancement over YOLOv5, marking the latest development in the YOLO series with significant improvements in detection accuracy and speed. To bolster feature extraction capabilities, YOLOv7 employs an extended efficient layer aggregation network (E-ELAN) with a redesigned architecture, introduces the MPConv module for downsampling that combines pooling and convolution to minimize feature loss, and improves the SPP module in the Backbone to prevent image distortion by integrating a series of convolution operations into multiple parallel pooling actions. It continues to use the PANet structure in the Neck for effective feature layer fusion and employs REPConv in the Head for channel adjustment, which simplifies its structure during inference without losing accuracy. Despite its enhanced accuracy, YOLOv7's complexity and large parameter count make it less suited for edge devices due to high computational requirements.

Simplifying YOLOv7 for edge GPUs involves streamlining its architecture, which is composed of three main parts: the backbone network (Backbone) for initial feature extraction, the neck network (Neck) for further feature processing and fusion, and the prediction head (Head) for detecting and classifying objects. This modification aims to maintain high detection performance while ensuring the model runs efficiently on edge devices with limited computational power, as shown in Figure 1.

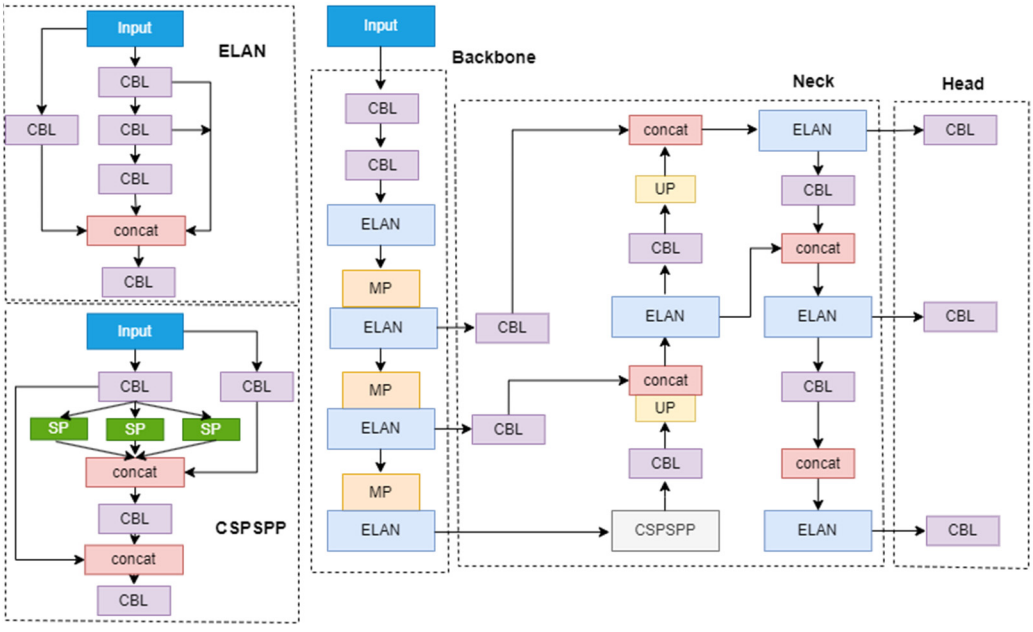


**Figure 1.** YOLOv7-tiny network structure.

In simplifying YOLOv7 for edge GPUs, the structure is modified to improve efficiency. The Backbone uses a simpler ELAN instead of E-ELAN and eliminates convolution in MPConv, relying solely on pooling for downsampling while retaining an optimized SPP structure to enrich Neck layer inputs. The Neck continues to use the PANet structure for feature aggregation, and the Head uses standard convolution instead of REPConv for channel adjustment.

### *2.2. Model Pruning*

To achieve faster detection speeds with the improved YOLOv7-tiny, enabling real-time defect detection in steel and a lightweight model, this study applies six different pruning criteria. The L1 [11] pruning algorithm accelerates CNNs by removing filters considered to have minimal impact on output accuracy, significantly reducing computational costs by eliminating entire filters and their connecting feature maps. Layer-adaptive magnitude-based pruning (LAMP) [12] optimizes the balance between sparsity and performance by calculating and prioritizing the removal of less important connections within a layer, achieving global pruning and automatically determined inter-layer sparsity. GroupNormPruner, specifically designed for networks using Group Normalization, is another method for pruning in deep learning models. Network pruning techniques aim to reduce model size and computational complexity by removing certain weights or neurons, enhancing efficiency while maintaining performance. GroupNormPruner [13] analyzes weights in layers using group normalization to identify and remove weights or channels with minimal impact on model output. GroupSlimPruner focuses on finely pruning network weights in a grouped manner while keeping the model structure stable. It allows for more granular adjustments, reducing unimportant parameters and features to enhance efficiency in resource-limited environments with minimal performance impact. GroupSlPruner [13] adds sparse training to GroupNormPruner. SlimPrune [14] combines Group Lasso and Sparse Group Lasso concepts for sparsity at both group and individual feature levels, using coordinate descent to solve a convex optimization problem.

## 3. Method

### *3.1. YOLO-RDP Model*

To enhance the detection accuracy for small objects and further lighten YOLOv7-tiny, this paper makes several improvements. Firstly, it draws on the RexNet concept for lightweight image classification networks to modify the Backbone, reducing dense connections and increasing network depth for richer feature extraction with lower computational cost. Secondly, in the Neck part, it employs the lightweight GSConv module for feature aggregation and replaces ELAN with VoVGSCSP to reduce parameters and computational demands while preserving feature richness. Thirdly, it innovatively combines the strengths of Dynamic (DyHead) [15] and Decoupled Head [16], integrating scale, spatial, and task awareness of DyHead with feature decoupling and pixel-level prediction of Decoupled Head to recognize minor defects better. The improved YOLOv7-tiny network thus incorporates RexNet, GSConv, and VoVGSCSP modules, and a novel dual detection head for enhanced performance, as shown in Figure 2.

For a more lightweight model with efficient feature extraction, ReXNet is selected as the Backbone over other lightweight networks. ReXNet improves upon MobileNetV2 by adjusting to alleviate feature representation bottlenecks and significantly reduces parameter size, offering an optimized balance between model complexity and lightweight architecture. In the Neck layer, to preserve semantic information and reduce parameter and computational costs, GSConv modules replace standard convolutions for upsampling and downsampling. Due to YOLOv7-tiny's extensive use of ELAN networks, which are densely connected by standard convolutions, the model's complexity and parameter count are high, which is not conducive for feature extraction. By substituting ELAN with the lightweight VOV-GSCSP module, the model's parameter count significantly decreases with minimal accuracy loss, enhancing efficiency without substantially sacrificing performance.

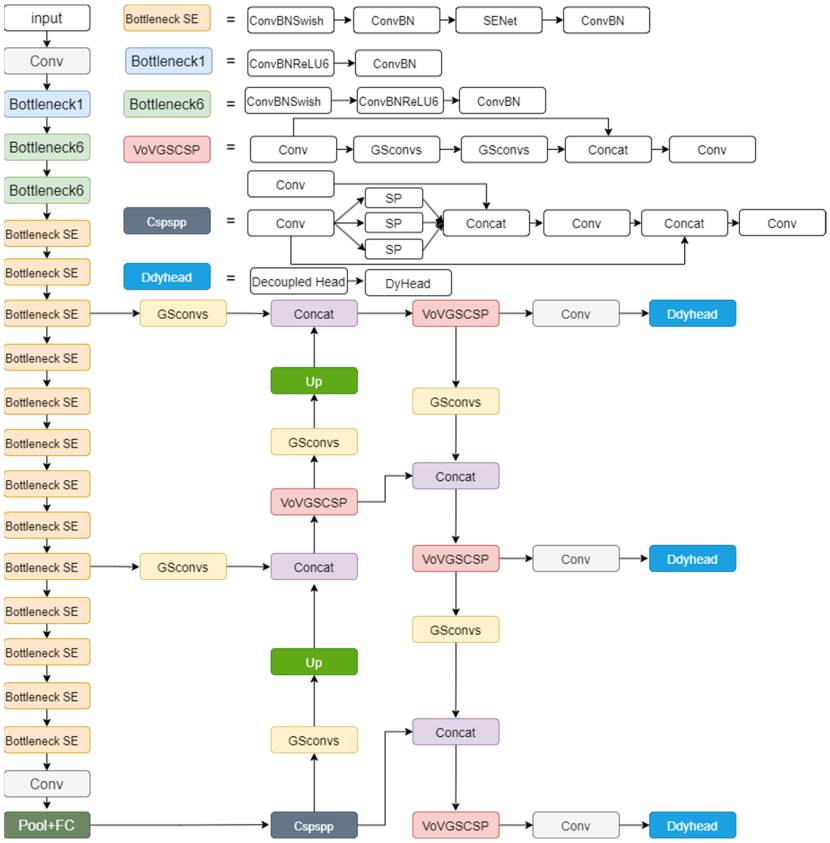


**Figure 2.** YOLO-RDP model.

#### 3.1.1. ReXNet Lightweight Network

ReXNet improves upon MobileNetV2 by addressing feature representation bottlenecks through strategic modifications. It retains MobileNetV2's core elements such as Inverted Residuals, Linear Bottleneck, and SENet's Squeeze-and-Excitation (SE) modules while enhancing channel numbers, adopting the Swish-1 activation function, and designing additional expansion layers. These adjustments alleviate bottlenecks and form the foundation of ReXNet's lightweight network structure, optimizing performance and efficiency.

The ReXNet model processes data similarly to most CNN models, enhancing its data handling capabilities, accelerating computational efficiency, and effectively saving computational resources through optimization algorithms. It addresses the challenge of fully representing image features without compression by elevating the rank of network module data, a concept that is integral to the design of lightweight neural networks. A key architectural element of the ReXNet lightweight network is the inverted residual structure made up of depthwise separable convolutions. Its basic principle involves replacing complete convolution operators with decomposed convolution operations, achieving the same computational effect with fewer operators and calculations.

The inverted residual structure effectively prevents the loss of feature information that can occur when conventional convolution kernels have too many zeros, meaning the kernels fail to perform their feature extraction role. By utilizing the inverted residual structure, more feature data information can be captured, thereby improving the model's training performance. The inverted residual structure primarily employs an initial dimension-expansion operation in the network architecture, meaning that it starts with the expansion of the expansion layer, controlled by an expansion factor; at this stage, the activation function of the dimension-expansion convolution layer is ReLU6. The main goal is to capture more feature extraction information. This is followed by feature extraction through depthwise convolution (DW), where the feature extraction convolution layer's activation function is also ReLU6. Finally, a dimension-reduction convolution process is performed, using a

linear activation function. The overall network structure exhibits a shape that is small at both ends and large in the middle. This aspect is a stark contrast to the traditional residual structure, presenting opposite structures, hence termed as the inverted residual structure.

The concept of separable convolution can be divided into spatially separable convolution and depthwise separable convolution. The core convolutional layers of the ReXNet network are based on depthwise separable convolution. Depthwise separable convolution decomposes a standard convolution into a depthwise convolution and a pointwise convolution. The depthwise convolution performs lightweight filtering by applying separate convolutional filters to each input channel, while the pointwise convolution constructs new features through linear combinations of input channels, achieving dimensionality reduction and expansion of the feature map.

#### 3.1.2. GSConv and VOV-GSCSP Lightweight Modules

The GSConv module combines standard convolution, depthwise separable convolution, and shuffle operations to blend features generated by standard convolutions with those from depthwise separable convolutions through a shuffle strategy. This approach achieves output comparable to standard convolution while reducing computational cost. By incorporating depthwise separable convolution and shuffle layers, GSConv enhances the non-linear representation of features, making it more suitable for lightweight model detectors by improving efficiency without significantly sacrificing performance. The calculation formula for depthwise separable convolution is Equation (1), and the calculation formula for GSConv convolution is Equation (2).

$$GFLOP_{S1} = W \times H \times K \times K \times 1 \times C_{out} \tag{1}$$

$$GFLOP_{S2} = W \times H \times K \times K \times 1 \times \frac{C_{\text{out}}}{2}(C_{in} + 1) \tag{2}$$

$W$ and $H$ represent the width and height of the feature map; $K$ is the kernel size; and $C_{in}$ and $C_{out}$ are the numbers of input and output channels, respectively. According to the formula, as the number of input feature channels increases, the computational cost of GSConv convolution approximates half that of standard convolution, yet its feature extraction capability remains comparable. Introducing GSConv reduces model complexity. To accelerate model inference time while maintaining accuracy, the VOV-GSCSP module is added, showcasing Unit2 of Figure 3 as the bottleneck unit structure of VOV-GSCSP and Unit3 designed with a single aggregation method for cross-stage VOV_GSCSP module implementation.

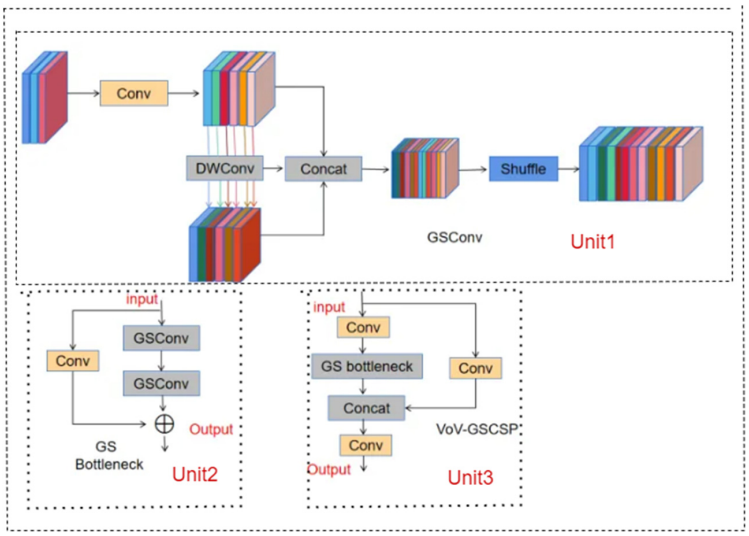


**Figure 3.** Structure diagram of GSConv and VOV-GSCSP lightweight modules.

3.1.3. Dual Detection Head DdyHead with a Symmetric Structure

To enhance the recognition of minor defects in target detection models(see Figure 4), this paper introduces an innovative approach by ingeniously integrating the advantages of dynamic detection heads (DyHead) and decoupled heads (Decoupled Head). This novel dual detection head strategy not only combines the DyHead's capabilities for scale awareness, spatial awareness, and task awareness but also leverages the strengths of the Decoupled Head in feature decoupling and pixel-level prediction.

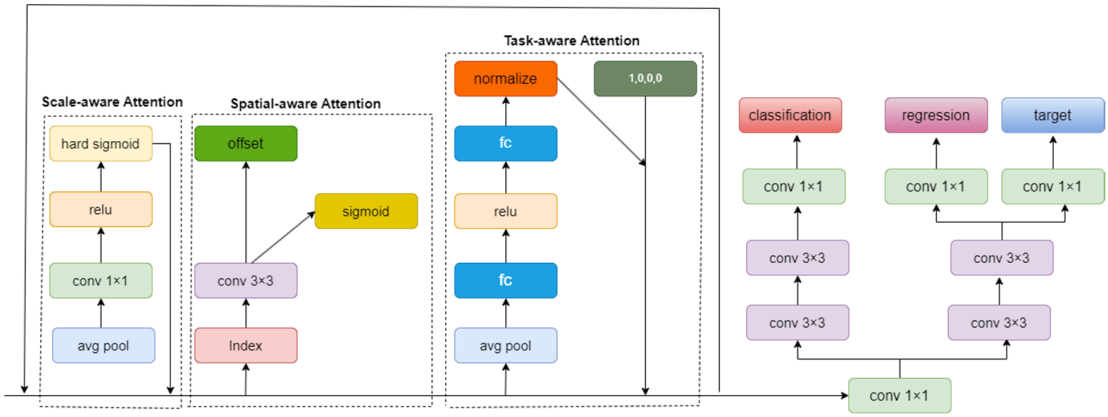


**Figure 4.** Structural diagram of DdyHead.

In this method, the DyHead provides rich scale-aware, space-aware, and task-aware information, enhancing the model's adaptability to various target sizes, understanding of object placement, and contextual comprehension. The Decoupled Head separately extracts and learns object location and category information through distinct network branches, reducing model complexity and computational load. Integrating the DyHead and the Decoupled Head addresses small target detection challenges by efficiently extracting key details from complex images, thus improving accuracy in identifying minor defects. This innovative dual detection head approach offers a new perspective and method in the field, showing remarkable performance in practical applications and injecting vitality into the development of detection technologies.

*3.2. YOLO-RDP Model Pruning*

The Slim pruning algorithm for the YOLO-RDP model operates through precise control at the channel level within CNNs. This control is achieved by imposing sparsity-inducing regularization on scaling factors during training, allowing the model to maintain high accuracy while achieving efficient compression. By intelligently identifying and removing unimportant channels, this reduces the model's parameter count, computational demand, and memory footprint, making it more suitable for deployment in resource-constrained environments.

The principle of channel pruning based on weight $\gamma$ is as follows: By pruning the channels matched with scaling factors close to 0 in the model, a network structure with fewer parameters and lower computational complexity is obtained. By setting a global threshold for all layers of the network to determine the size of the scaling factor $\gamma$, model pruning is achieved. For example, setting the global threshold to 0.3, sorting all $\gamma$ values of the model, and pruning 30% of the channels with smaller values achieve model compression.

First, the batch normalization (BN) layers of the model are processed, and the scaling factors within the normalization are batch-normalized. Then, the normalized scaling factors are subjected to L1 regularization to train the network, enabling the model to acquire sparsity. Subsequently, utilizing a pre-defined skip-layer rule, network slimming is employed to prune the image classifier of the model. This significantly reduces the redundancy of the model. For the detection of steel defects, which often involve numerous

small targets and where detection accuracy is highly sensitive to model pruning, it is necessary to retrain the pruned model with the same model parameters to obtain the adjusted pruned model.

## 4. Experiment

### *4.1. Experimental Design*

(1) Dataset

In our experiment, we employed two popular public datasets, GC10-DET and NEU-DET (see Figure 5).

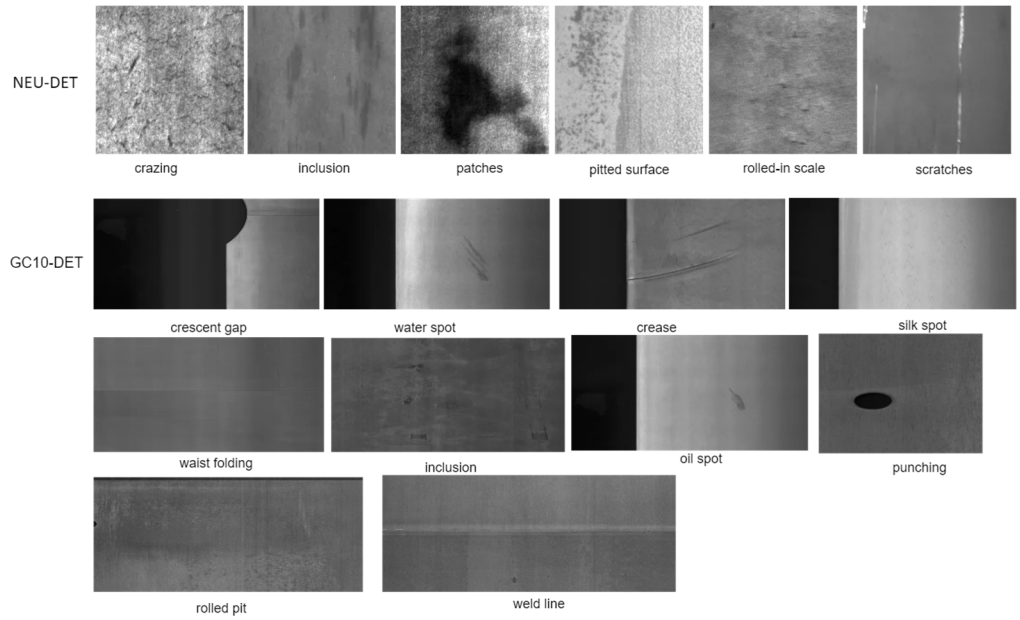


**Figure 5.** Two datasets with different resolutions.

- NEU-DET is a publicly available dataset created by Northeastern University. The dataset consists of 1800 grayscale images and is divided into six different types of typical surface defects. Each type of defect contains 300 samples. These six types of defects are rolled-in scale, patches, crazing, pitted surface, inclusion, and scratches. The above six defects are all common and representative. We will describe in detail the style and reasons for each type of defect below.

Crazing (see Figure 5) typically appears as straight lines, sometimes also in a "Y" shape, often aligned with the direction of forging or rolling, with sharp angles at the openings. Surface cracks in steel materials are mostly caused by rapid heating or improper cooling during the casting process. Uneven heating or improper rolling can result in excessive internal stress in the steel, leading to crack formation. Additionally, cracks can be caused by the extension of sub-surface bubbles, internal cracks, or impurities during the rolling process.

Scratching, also known as scoring, refers to the fine and elongated grooves that appear on the surface of steel under external force along the rolling direction. Improper installation and wear of guide devices, as well as the accumulation of foreign substances such as oxidized iron scale coming into contact with the steel during rolling, are the main causes of scratching.

Rolled-in scale refers to the occurrence of oxide colors such as deep blue, light blue, brown, light yellow, and red on the surface of steel plates after oxidation. This is caused by excessive oxygen content in the gas inside the equipment or inadequate sealing during the annealing process. It can also occur when the temperature is too high during heating before exiting the furnace, leading to brief contact of the steel with water and air. If scale

removal by high-pressure water is not thorough, surface defects will form on the steel after rolling is finished.

Pitted surface refers to the uneven, rough surface and pits on the surface of steel, often distributed in patches. Its cause is often due to foreign substances adhering to the rolls during rolling. If granular foreign substances are pressed into the steel during rolling, they can also form mottling and pits when they detach after cooling.

Inclusion refers to impurities that may be entrapped in steel during the production process, forming inclusions, which appear as black spot-like or linear specks on the surface or inside of the steel.

Patches refer to iron beans generated when the pouring temperature of molten iron is too low, which cannot be remelted by the molten iron. As a result, they are entrapped within the casting along with external gases. Alternatively, in the phenomenon where iron beans appear on the surface of T-shaped groove platform castings, it indicates the presence of small iron beads within the blowholes.

- GC10-DET is a benchmark dataset collected from real industrial scenarios provided by Lv [17]. The dataset including punching (Pu), weld line (Wl), crescent gap (Cg), water spot (Ws), oil spot (Os), silk spot (Ss), inclusion (In), rolled pit (Rp), crease (Cr), and waist folding (Wf) [18]. Compared to NEU-DET, it features 10 different types of defects, with varying numbers of images for each defect. In Figure 6, we can observe significant differences in the quantity of each type of defect.

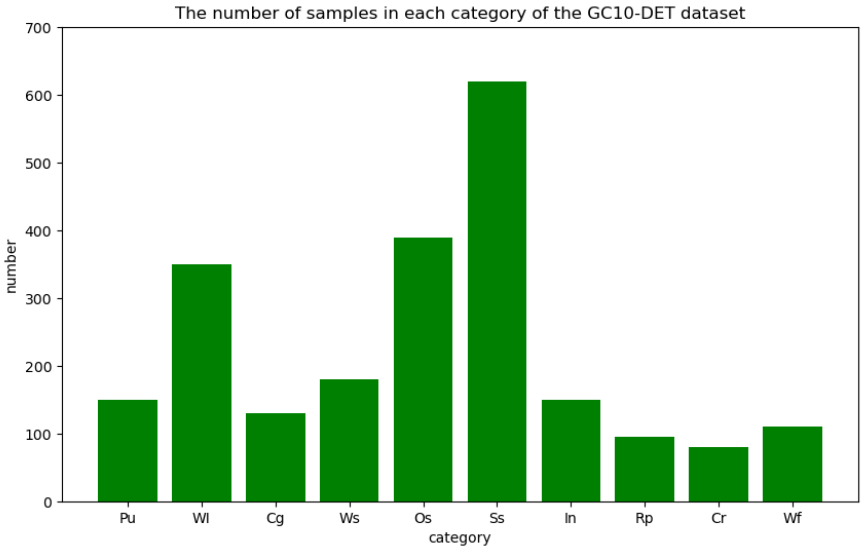


**Figure 6.** The number of samples in each category of the GC10-DET dataset.

In summary, the differences between the two datasets are as follows:

- NEU-DET contains six types of defects, which is four fewer than GC10-DET. Additionally, NEU-DET includes 1800 grayscale images, whereas GC10-DET contains 2257 grayscale images.
- The GC10-DET dataset exhibits class imbalance, with significant differences in the quantity of each type.

(2) Experimental parameters and environment

The model training in this paper was conducted on an Ubuntu 20.04 operating system, equipped with an NVIDIA RTX 3090 graphics card with 32 GB of VRAM. The model was built using the PyTorch 2.0.0 deep learning framework, with training acceleration provided by Cuda 11.8. During the experiment, we set the batch size to 32, considering it as the optimal choice to effectively utilize computational resources without exhausting memory. This batch size accelerates convergence, providing enough samples for gradient descent

calculations, and thereby stabilizing the training process. The choice of an initial learning rate of 0.01 is generally reasonable and is often considered a suitable starting point, subject to appropriate adjustments during training. Additionally, setting the momentum to 0.937 and weight decay to 0.0005 is a common practice, aiding faster convergence and enhancing the model's generalization ability during training. Setting the number of training epochs to 300 is also reasonable, as this duration typically allows the model to learn the dataset's features adequately and converge to a satisfactory state. Finally, adopting the stochastic gradient descent (SGD) optimizer is a common choice in deep learning, as it effectively updates model parameters, reduces the loss function, and entails lower computational costs. Specific parameters are listed in Table 1.

**Table 1.** Experimental Parameter Configuration.

| bs | Epoch | Ir | Momentum | Weight_Decay | Input_Size |
|---|---|---|---|---|---|
| 32 | 300 | 0.01 | 0.937 | 0.0005 | 640 |

We will partition the GC10-DET and NEU-DET datasets into training, validation, and testing sets according to an 8:1:1 ratio.

(3) Evaluation indicators

The experiment utilizes five evaluation metrics: precision (P), recall (R), parameters (Params), mean average precision (mAP), FPS, and FLOPs. The formulas for these metrics are as follows:

$$\mathrm{P} = \frac{TP}{TP + FP} \times 100\% \tag{3}$$

$$R = \frac{TP}{TP + FN} \times 100\% \tag{4}$$

$$mAP = AP = \int_0^1 P(R)dR \times 100\% \tag{5}$$

In the formulas, P represents the proportion of positive samples among all samples; R signifies the ratio of samples correctly predicted as positive among all positive samples; TP is the number of correctly matched predicted frames to annotated frames; FP is the number of incorrectly predicted frames; FN is the count of unpredicted annotated frames; AP denotes the average precision for a category, and mAP is the mean of AP across all categories, serving as a comprehensive indicator of accuracy. This paper focuses on detecting single-category, pointer-type meters, so mAP equals AP. Besides these metrics, the size of the network model and the number of computational parameters are also used as evaluation criteria. Params reflect the number of parameters in the model, indicating the model's memory usage. FLOPs measure the computational complexity of the model, reflecting the amount of computation involved.

### *4.2. Comparative Experiment*

To further validate the effectiveness of the algorithm presented in this paper, comparative experiments were conducted against mainstream two-stage and one-stage target detection algorithms, including SSD, Faster-RCNN, YOLOv5s, and YOLOv7-tiny. The results, shown in Tables 2 and 3, indicate that the proposed model outperforms the comparative models. Specifically, on the NEU-DET and GC10-DET datasets, the method improves mAP by 3.7% and 3.5% over YOLOv7-tiny, respectively, while also reducing parameter count by 40% and 30%, and computational load by 25% and 24%, showing significant advancements over other mainstream models.

**Table 2.** Experimental comparisons on the NEU-DET dataset.

| Method | P% | R% | mAP% | Params/$10^6$ | FLOPs/$10^9$ |
|---|---|---|---|---|---|
| SSD | 66.4 | 71.2 | 71.4 | 22.4 | 77.5 |
| Faster-RCNN | 73.1 | 69.2 | 77.3 | 107 | 90.9 |
| YOLOv5s | 65.6 | 70.6 | 70.7 | 7.0 | 15.8 |
| YOLOv7-tiny | 72.8 | 67.1 | 76.1 | 6.0 | 13.2 |
| ours | 67.0 | 77.9 | **79.8** | **3.5** | **9.95** |

**Table 3.** Experimental comparisons on the GC10-DET dataset.

| Method | P% | R% | mAP% | Params/$10^6$ | FLOPs/$10^9$ |
|---|---|---|---|---|---|
| SSD | 62.1 | 64.5 | 65.1 | 22.4 | 77.5 |
| Faster-RCNN | 73.2 | 69.8 | 74.1 | 107 | 90.9 |
| YOLOv5s | 74.6 | 67.1 | 73.1 | 7.0 | 15.8 |
| YOLOv7-tiny | 81.1 | 66.5 | 72.9 | 6.0 | 13.1 |
| ours | 80.1 | 72.7 | **76.4** | **4.21** | **9.9** |

To select the most suitable pruning algorithm, six different pruning algorithms were tested under the condition of accelerating the process by 1.5 times on both the GC10-DET and NEU-DET datasets. The results of these experiments are presented in Table 4 as well as in Table 5 and Figure 7.

**Table 4.** Comparison experiment of pruning for NEU-DET.

| YOLO-RDP | L1 | Lamp | Slim | Group_Slim | Group_Norm | Group_Sl | mAP% |
|---|---|---|---|---|---|---|---|
| √ | | | | | | | 79.2 |
| √ | √ | | | | | | 77.1 |
| √ | | √ | | | | | 77.5 |
| √ | | | √ | | | | **79.8** |
| √ | | | | √ | | | 77.1 |
| √ | | | | | √ | | 77.8 |
| √ | | | | | | √ | 74.3 |

**Table 5.** Comparison experiment of pruning for GC10-DET.

| YOLO-RDP | L1 | Lamp | Slim | Group_Slim | Group_Norm | Group_Sl | mAP% |
|---|---|---|---|---|---|---|---|
| √ | | | | | | | 74.0 |
| √ | √ | | | | | | 72.9 |
| √ | | √ | | | | | 73.8 |
| √ | | | √ | | | | **76.4** |
| √ | | | | √ | | | 73.6 |
| √ | | | | | √ | | 70.0 |
| √ | | | | | | √ | 67.9 |

The comparative experiments show that under the condition of 1.5× pruning acceleration, the Slim pruning algorithm is more suitable for the improved YOLOv7-tiny model compared to other pruning algorithms. On the NEU-DET dataset, this algorithm increased accuracy by 0.6 points while compressing the model, and on the GC10-DET dataset, it improved accuracy by 2.4 points. Hence, the Slim pruning algorithm was chosen for model compression. From Figures 8 and 9, we can observe the pruning ratio for each layer. The vertical axis represents the number of channels, while the horizontal axis represents the name of each layer. The orange part represents 'before pruning', and the red part represents 'after pruning'.

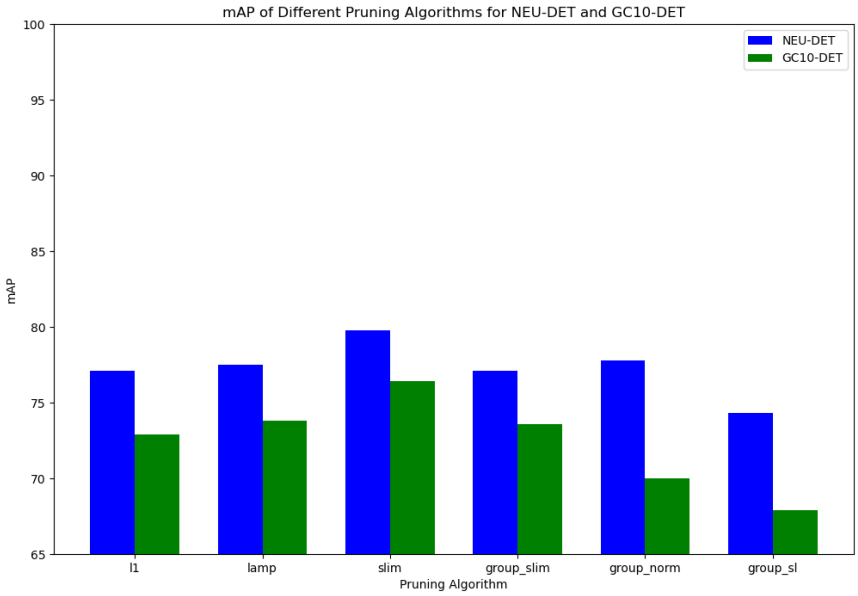


**Figure 7.** mAP of different pruning algorithms for NEU-DET and GC10-DET.

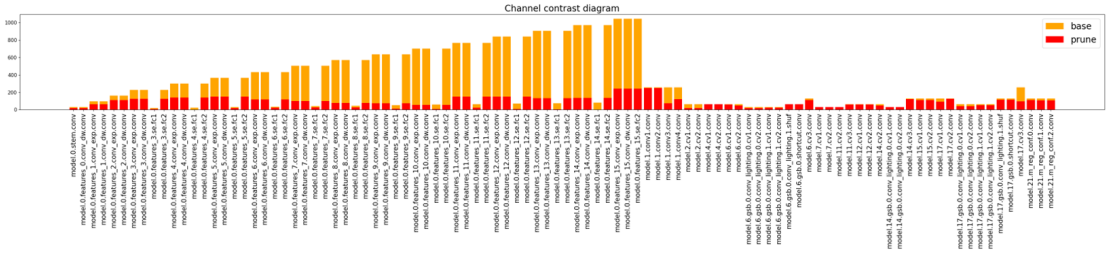


**Figure 8.** Comparison of channel before and after pruning for NEU-DET.

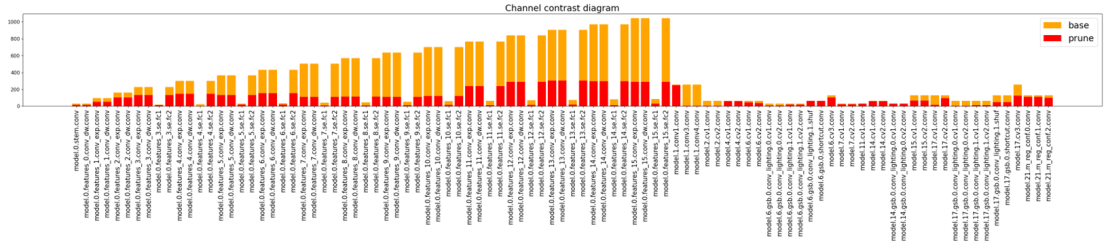


**Figure 9.** Comparison of channel before and after pruning for GC10-DET.

### *4.3. Ablation Experiments*

Conducted on the GC10-DET and NEU-DET datasets, our experiments showcased the efficacy of each component via ablation studies. This validation underscores the enhanced precision of our proposed surface defect detection method. Furthermore, to ascertain the impact of each refined component on the YOLOv7-tiny network model, combination experiments were executed, controlling for variables, thereby reinforcing the effectiveness of the improvement strategies. The results of the ablation experiments on the GC10-DET and NEU-DET datasets are shown in Tables 6 and 7.

From Tables 6 and 7, we can see that our improved model has a 3.7% higher mAP on the NEU-DET dataset, while the number of parameters has been reduced by $2.52 \times 10^6$, and the computational cost has been reduced by $3.25 \times 10^9$. On the GC10-DET dataset, the mAP increased by 3.5%, while the number of parameters decreased by $1.82 \times 10^6$, and the computational cost was reduced by $3.2 \times 10^9$. Therefore, the model proposed in this study

is a significant improvement over the original model. The experiments show that while the improved detection head increases the model's accuracy, it also increases its computational and parameter requirements. At this point, we have greatly reduced the model's number of parameters and computational cost by employing a more lightweight Backbone, ReXNet, along with lighter modules GSConv, VOV-GSCSP, and pruning methods, constructing a model suitable for deployment on edge terminal devices.

**Table 6.** Ablation experiments on NEU-DET.

| YOLOv7-Tiny (Base) | ReXNet | GSConv + VOV-GSCSP | DdyHead | Slim Pruning | P% | R% | mAP% | Params /$10^6$ | FLOPs /$10^9$ |
|---|---|---|---|---|---|---|---|---|---|
| √ | | | | | 72.8 | 67.1 | 76.1 | 6.03 | 13.2 |
| √ | √ | | | | 66.9 | 70.5 | 70.1 | 6.65 | 12.1 |
| √ | √ | √ | | | 64.7 | 72.5 | 71.2 | 4.94 | 8.6 |
| √ | √ | √ | √ | | 86.2 | 70.0 | 79.2 | 6.87 | 14.9 |
| √ | √ | √ | √ | √ | 67.0 | 77.9 | **79.8** | **3.51** | **9.95** |

**Table 7.** Ablation experiments on GC10-DET.

| YOLOv7-Tiny (Base) | ReXNet | GSConv + VOV-GSCSP | DdyHead | Slim Pruning | P% | R% | mAP% | Params/$10^6$ | FLOPs/$10^9$ |
|---|---|---|---|---|---|---|---|---|---|
| √ | | | | | 81.1 | 66.5 | 72.9 | 6.03 | 13.1 |
| √ | √ | | | | 77.5 | 67.2 | 72.4 | 6.66 | 12.1 |
| √ | √ | √ | | | 76.9 | 69.2 | 71.0 | 4.95 | 8.6 |
| √ | √ | √ | √ | | 61.2 | 78.9 | 74.0 | 6.87 | 14.9 |
| √ | √ | √ | √ | √ | 80.1 | 72.7 | **76.4** | **4.21** | **9.9** |

## 5. Conclusions

This paper introduces a novel single-stage lightweight detection model tailored specifically for steel defect detection. The primary objective is to address the limitations of traditional object detection models, particularly in detecting small target defects, while concurrently optimizing the model's weight efficiency.

In order to achieve this, the paper employs several innovative techniques. First and foremost, it tackles the issue of model weight quantization by leveraging advancements such as the ReXNet lightweight network architecture, GSConv, and VOV-GSCSP lightweight modules. These elements play a crucial role in reducing the parameter count and computational cost of the network model without compromising its detection accuracy.

Furthermore, to enhance the detection performance for small targets, the paper integrates the dynamic object detection head (DyHead) and the decoupled head (Decoupled Head) methodologies. By combining these approaches, the model becomes more adept at identifying minute defects, thus improving overall detection sensitivity.

To evaluate the efficacy of the proposed model, extensive experimental studies were conducted using two widely recognized public datasets. The results clearly demonstrate the superiority of the proposed model over existing approaches. The results of partial steel defect detection are shown in Figure 10. Not only does it exhibit enhanced recognition performance for minor steel defects, but it also achieves significant reductions in model size and computational cost. These findings suggest that the proposed model is well-suited for deployment on edge terminal devices with limited computing resources, making it practical for real-world applications.

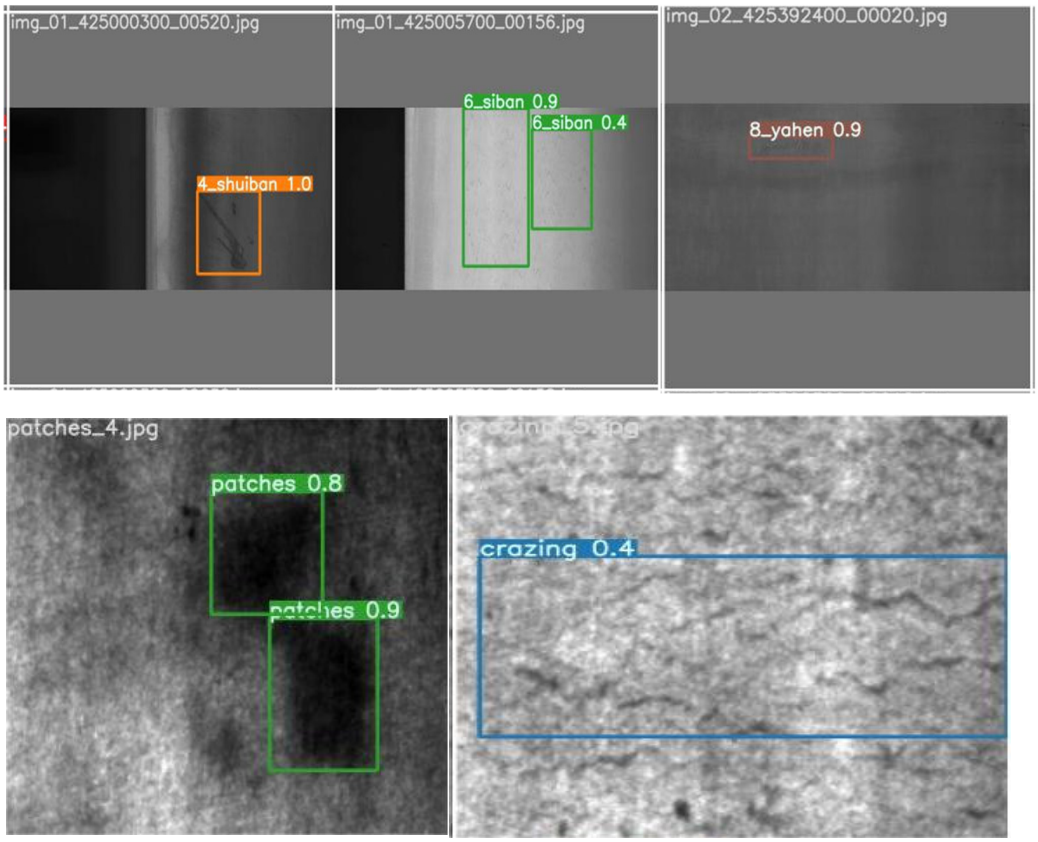


**Figure 10.** Part of steel strip defect detection results.

However, it is worth noting that the model's effectiveness in detecting lighter-colored defects, such as Wf in the GC10-DET dataset, falls short of the desired standard. As a result, future research endeavors will focus on further refining the model to improve its accuracy in identifying these types of defects.

In summary, this paper presents a robust and efficient solution for steel defect detection, offering advancements in both detection performance and model optimization. By addressing the challenges associated with small target detection and model weight efficiency, the proposed model demonstrates promising prospects for practical deployment in industrial settings.

**Author Contributions:** Conceptualization was conducted by G.Z.; methodology was also developed by G.Z.; G.Z. handled software development; validation was conducted by G.Z. and S.N.; S.L. performed formal analysis; investigation was carried out by S.L.; G.Z. provided resources; G.Z. curated the data; original draft preparation was undertaken by G.Z.; review and editing were performed by S.L., S.N. and L.Y. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by the National Natural Science Foundation of China (Grant No. 61762085) and the Natural Science Foundation of Xinjiang Uygur Autonomous Region Project (Grant No. 2019D01C081).

**Data Availability Statement:** The data are available upon request.

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

## References

1. Arya, C.; Tripathi, A.; Singh, P.; Diwakar, M.; Sharma, K.; Pandey, H. Object detection using deep learning: A review. *J. Phys. Conf. Series* **2021**, *1854*, 012012.
2. Patwal, A.; Diwakar, M.; Tripathi, V.; Singh, P. An investigation of videos for abnormal behavior detection. *Procedia Comput. Sci.* **2023**, *218*, 2264–2272. [CrossRef]

3. Roka, S.; Diwakar, M.; Singh, P.; Singh, P. Anomaly behavior detection analysis in video surveillance: A critical review. *J. Electron. Imaging* **2023**, *32*, 042106. [CrossRef]
4. Gangadharan, S.M.P.; Arya, C.; Aluvala, S.; Singh, J.; Singh, P.; Murugesan, A. Advancing Bug Detection in Solidity Smart Contracts with the Proficiency of Deep Learning. In Proceedings of the 2023 3rd International Conference on Innovative Sustainable Computational Technologies (CISCT), Dehradun, India, 8–9 September 2023; IEEE: New York, NY, USA; pp. 1–5.
5. Li, J.; Su, Z.; Geng, J.; Yin, Y. Real-time detection of steel strip surface defects based on improved YOLO detection network. *IFAC Pap. OnLine* **2018**, *51*, 76–81. [CrossRef]
6. Cheng, J.Y.; Duan, X.H.; Zhu, W. Research on metal surface defect detection by improved YOLOv3. *Comput. Eng. Appl.* **2021**, *57*, 252–258.
7. Wang, Y.; Wang, H.; Xin, Z. Efficient detection model of steel strip surface defects based on YOLO-V7. *IEEE Access* **2022**, *10*, 133936–133944. [CrossRef]
8. Han, D.; Yun, S.; Heo, B.; Yoo, Y. Rethinking Channel Dimensions for Efficient Model Design. *arXiv* **2020**, arXiv:2007.00992v3. [CrossRef]
9. Li, H.; Li, J.; Wei, H.; Liu, Z.; Zhan, Z.; Ren, Q. Slim-neck by GSConv: A better design paradigm of detector architectures for autonomous vehicles. *arXiv* **2022**, arXiv:2206.02424.
10. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y.M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for realtime object detectors. *arXiv* **2022**, arXiv:2207.02696.
11. Li, H.; Kadav, A.; Durdanovic, I.; Samet, H.; Graf, H.P. Pruning filters for efficient convnets. In Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 24–26 April 2017.
12. Lee, J.; Park, S.; Mo, S.; Ahn, S.; Shin, J. Layer-adaptive Sparsity for the Magnitude-based Pruning. In Proceedings of the International Conference on Learning Representations, Vienna, Austria, 4 May 2021.
13. Fang, G.; Ma, X.; Song, M.; Mi, M.B.; Wang, X. Depgraph: Towards any structural pruning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 16091–16101.
14. Liu, Z.; Li, J.; Shen, Z.; Huang, G.; Yan, S.; Zhang, C. Learning efficient convolutional networks through network slimming. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2736–2744.
15. Dai, X.; Chen, Y.; Xiao, B.; Chen, D.; Liu, M.; Yuan, L.; Zhang, L. Dynamic Head: Unifying Object Detection Heads with Attentions. *arXiv* **2021**, arXiv:2106.08322. [CrossRef]
16. Ge, Z.; Liu, S.; Wang, F.; Li, Z.; Sun, J. YOLOX: Exceeding YOLO Series in 2021. *arXiv* **2021**, arXiv:2107.08430. [CrossRef]
17. Lv, X.; Duan, F.; Jiang, J.-J.; Fu, X.; Gan, L. Deep Metallic Surface Defect Detection: The New Benchmark and Detection Network. *Sensors* **2020**, *20*, 1562. [CrossRef] [PubMed]
18. Xiang, X.; Wang, Z.; Zhang, J.; Xia, Y.; Chen, P.; Wang, B. AGCA: An adaptive graph channel attention module for steel surface defect detection. *IEEE Trans. Instrum. Meas.* **2023**, *72*, 1–12. [CrossRef]