

Article

An Improved YOLOv5 Algorithm for Tyre Defect Detection

Mujun Xie, Heyu Bian, Changhong Jiang *, Zhong Zheng and Wei Wang

School of Electrical and Electronic Engineering, Changchun University of Technology, Changchun 130012, China; xiemujun@ccut.edu.cn (M.X.); 2202204021@stu.ccut.edu.cn (H.B.); 2202204048@stu.ccut.edu.cn (Z.Z.); 2202204036@stu.ccut.edu.cn (W.W.)

* Correspondence: jch@ccut.edu.cn; Tel.: +86-135-0448-6608

Abstract: In this study, a tyre defect detection model is improved and optimized under the YOLOv5 framework, aiming at radial tyre defects with characteristics such as an elongated shape and various target sizes and defect types. The DySneakConv module is introduced to replace the first BottleneckCSP in the Backbone network. The deformation offset of the DySneakConv module is used to make the convolutional energy freely adapt to the structure to improve the recognition rate of tyre defects with elongated features; the AIFI module is introduced to replace the fourth BottleneckCSP, and the self-attention mechanism and the processing of large-scale features are used to improve the recognition rate of tyre defects with elongated features using the AIFI module. This latter module has a self-attention mechanism and the ability to handle large-scale features to solve the problems of diverse tyre defects and different sizes. Secondly, the CARAFE up-sampling operator is introduced to replace the up-sampling operator in the Neck network. The up-sampling kernel prediction module in the CARAFE operator is used to increase the receptive field and allow the feature reorganization module to capture more semantic information to overcome the information loss problem of the up-sampling operator. Finally, based on the improved YOLOv5 detection algorithm, the Channel-wise Knowledge Distillation algorithm lightens the model, reducing its computational requirements and size while ensuring detection accuracy. Experimental studies were conducted on a dataset containing four types of tyre defects. Experimental results for the training set show that the improved algorithm improves the mAP0.5 by 4.6 pp, reduces the model size by 25.6 MB, reduces the computational complexity of the model by 31.3 GFLOPs, and reduces the number of parameters by 12.7×10^6 compared to the original YOLOv5m algorithm. Experimental results for the test set show that the improved algorithm improves the mAP0.5 by 2.6 pp compared to the original YOLOv5m algorithm. This suggests that the improved algorithm is more suitable for tyre defect detection than the original YOLOv5.

Keywords: YOLOv5; tyre defect detection; elongated features; multiple types of defects; knowledge distillation



Citation: Xie, M.; Bian, H.; Jiang, C.; Zheng, Z.; Wang, W. An Improved YOLOv5 Algorithm for Tyre Defect Detection. *Electronics* **2024**, *13*, 2207. <https://doi.org/10.3390/electronics13112207>

Academic Editors: Séjir Khojet El Khil, Monia Ben Khader Bouzid and Chiara Boccaletti

Received: 17 April 2024

Revised: 31 May 2024

Accepted: 3 June 2024

Published: 5 June 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

As the only essential component of the car in contact with the ground, tyres bear essential responsibilities such as load, cushioning and shock absorption. They directly affect the driving quality and safety of the car [1]. Currently, most tyre manufacturers rely on manual supervision to inspect the quality of tyres, which encounters problems such as low leakage rates and high cost, making it challenging to meet the demand of increasing automobile production.

The rapid development of computers and deep learning techniques [2,3], in recent years has provided an efficient, accurate and cost-effective solution for tyre defect detection. Object detection using deep learning methods is characterised by feature self-extraction, high adaptability and robustness compared to traditional detection methods, such as R-CNN [4], Fast R-CNN [5], Faster R-CNN [6], SSD [7], YOLO [8], RetinaNet [9], CenterNet [10], etc.

Existing research on deep learning detection methods for tyre defects has achieved various results. Pengfei Wang et al. [11] proposed a tyre tread defect detection model based on YOLOv5, which detects defects such as cracks and perforations on the tyre surface. Compared with YOLOv4, it shows excellent performance in overall detection accuracy. However, it shows low confidence in detecting individual tyre defect features. Zejun Wu et al. [12] proposed an automatic tyre defect detection method based on an improved Faster R-CNN, which detects defects such as sidewall foreign objects and crown root openings produced during tyre production. They combine the convolutional features of the third layer and the fifth layer of the convolutional neural network, outputting them as inputs to the ROI pooling layer and introducing the online hard example mining (OHEM) algorithm, after which the pooling layer achieved improved accuracy for tyre defect detection. Yunting Liu et al. [13] proposed a fused attention mechanism adversarial network (FAMGAN) with Skip-GANomaly as the basic framework to detect bubble defects whose differences compared to background pixels are too small. Firstly, the attention feature fusion and mechanism modules form a hopping layer, improving attention to the target features. Then, the joint up-sampling module is added to the discriminator, which enhances the model's speed in detecting image defects. Finally, the goal of improving the detection accuracy is achieved. Mingda Li et al. [14] proposed a tyre defect detection algorithm based on Faster R-CNN that can accurately locate and classify tyre defects by introducing ROI align pooling, thus improving detection efficiency and accuracy. Kuo-Chien Liao et al. [15] inspected and analysed aircraft composite structures using a UAV carrying an infrared camera and analysed the IR images by MATLAB (2020b) image analysis software. Not only can the internal health of the external composite structure of the aircraft be clearly identified, but the proposed procedure also reduces the inspection time significantly.

In this paper, an improved YOLOv5 model is designed for the detection of elongated tyre defects of diverse types and with different target sizes. The following studies are carried out: based on the YOLOv5m network framework, the convolutional nuclear energy is adapted more freely to the structure by introducing Dynamic Snake Convolution (DySnakeConv) into the Backbone network; attention-based intrascale feature interaction (AIFI) is introduced to better understand the relationship between different features at the same scale in the image; and the underlying content information kernel is predictively restructured by introducing the Content-Aware ReAssembly of FEatures (CARAFE) up-sampling operator in the Neck network. The relationship between different features at the same scale is better understood by introducing AIFI, and the underlying content information kernel is predictively restructured by introducing the CARAFE up-sampling operator. Finally, the improved YOLOv5m model is distilled to the YOLOv5s model using Channel-wise Knowledge Distillation to achieve a lightweight model. It was experimentally verified that the improved algorithm can detect tyre defects well.

2. YOLOv5 Network Improvement

2.1. Introduction to YOLOv5

The network architecture of YOLOv5 is shown in Figure 1:

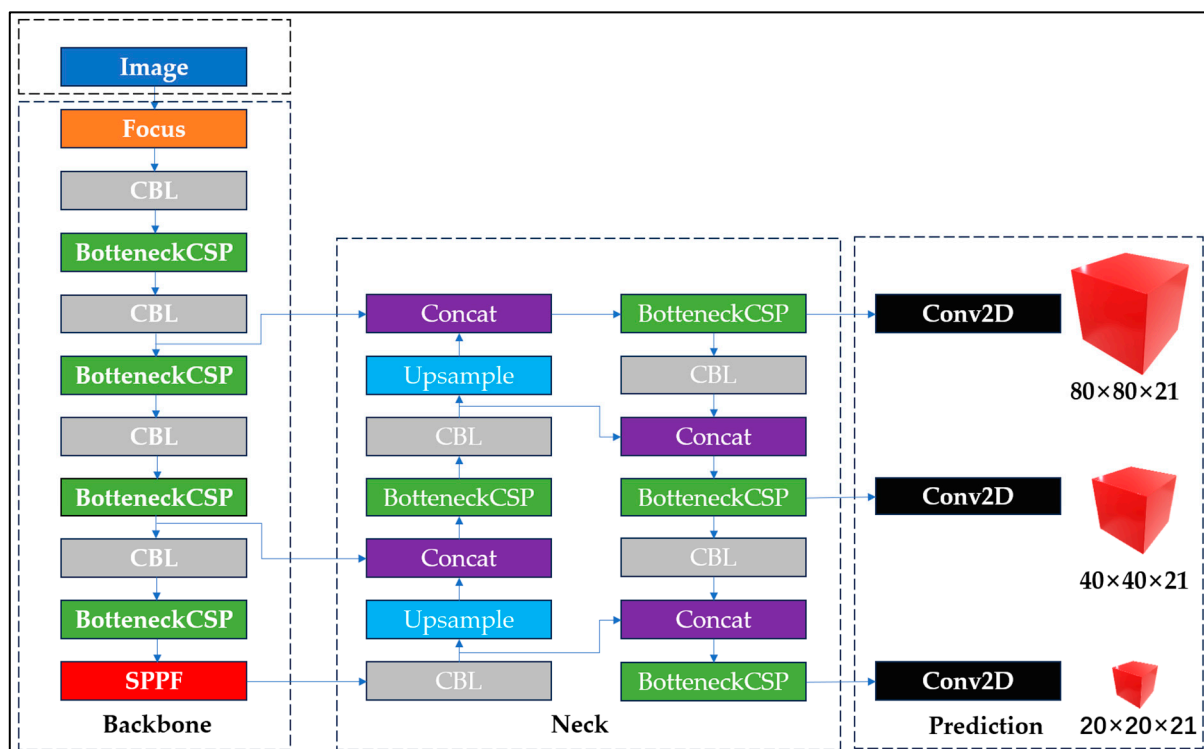


Figure 1. YOLOv5 network framework.

YOLOv5 adopts the classic YOLO architecture, which can be divided into four components: the input (Input), the backbone feature extraction network (Backbone), the Neck and the output layer (Prediction). The network framework includes key components such as Bottleneck Cross Stage Partial (BottleneckCSP), Convolutional Block Layer (CBL), Spatial Pyramid Pooling–Fast (SPPF), Concat and up-sample [16].

BottleneckCSP first processes the input through a bottleneck structure consisting of a convolutional layer, batch normalization and an activation function. Cross-stage partial connections are then introduced to classify the processed features. The role of the bottleneck structure is to facilitate the transfer of information and the flow of gradients in the network. The role of the CSP connections is to enable better sharing of features between different stages, thus improving the performance and training of the network.

CBL consists of a convolutional layer, batch normalization and an activation function. The convolutional layer is responsible for extracting the spatial information of the input features, batch normalization accelerates the training and improves the stability of the model and the activation function introduces nonlinear properties. It performs basic feature extraction and nonlinear transformations to help the network capture the abstract features in an input image.

As a pooling layer, SPPF can handle different sizes of input images and capture multi-scale information through pooling on spatial grids of various sizes to ensure that the model can detect targets of different sizes. Its main effect is improving the model's ability to perceive targets of different scales. Through pooling operations, the model can better adapt to targets of various shapes and reduce the amount of computation while preserving spatial information.

The Concat layer is used to connect multiple feature maps. The role is to combine features from different layers or modules to obtain more information and context, thus improving target detection performance.

Up-sample extends the spatial dimension of the input feature map using bilinear or nearest-neighbour interpolation [17]. It allows low-resolution feature maps to be matched with high-resolution ones, thus helping the network to better capture detailed information.

2.2. Improving the Backbone

2.2.1. DySneakConv

This study used a total of four categories of common tyre defect features: 2_open, cords_defects, impurity and belt_cuobian. Most tyre defects show elongated feature structures if the aspect ratio is greater than or equal to 2:1, which is the evaluation standard of elongated defects. In this paper, we use the open-source dataset of tyre X-ray photos produced by the Networked Control Center of Shanghai University, in which 76% of the 2_open samples, all cords_defects samples and 59.7% of the belt_cuobian samples are classified as elongated feature defects. Moreover, such elongated feature defects account for a relatively small proportion of the whole image, with a limited pixel composition. The traditional convolutional kernel used in the first BottleneckCSP of YOLOv5m is of a fixed shape, which is less effective in dealing with this type of elongated feature defect, resulting in poor performance when facing small changes in the target. Meanwhile, YOLOv5m tends to overfit morphological features presented by the data that it has not seen before, making it unable to effectively identify the unseen feature morphology and reducing its detection accuracy. To address this problem, this paper replaces the first BottleneckCSP in the Backbone layer using DySneakConv [18].

The introduced DySneakConv is a deformed convolution. The deformed convolution gives the convolution kernel spatial adaptability to changes in the target space by introducing learnable offsets. This adaptability allows the network to be more flexible in capturing changes in tyre defect features, improving the model's ability to model complex features. The deformed convolution also allows the receptive field to perceive the target over a broader range by reducing the receptive field blindness.

DySneakConv increases the range of deformation convolution through layer-by-layer positional changes, which in turn enables the Backbone network to better capture slender feature defects. To solve the problems of the convolution kernel not being flexible enough to focus on the target and the geometric feature perception domain being prone to deviate from the target, DySneakConv also employs an iterative strategy of deformation offsets Δ [19]. Each target to be processed will sequentially select the following position to be observed, thus ensuring the continuity and stability of attention [20]. Figure 2 shows the calculation method of DySneakConv coordinates.

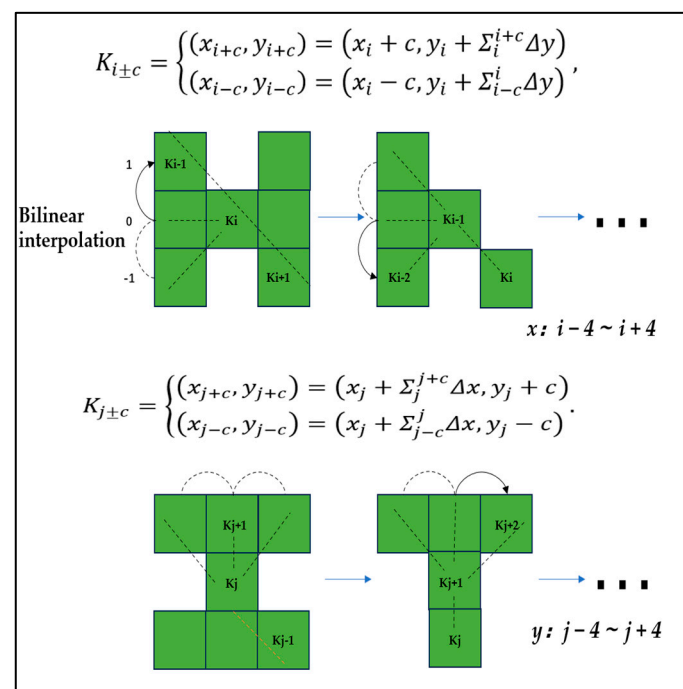


Figure 2. Calculation of DySneakConv coordinates.

The x -axis and y -axis change formulas are as follows:

$$K_{i\pm c} = \begin{cases} (x_{i+c}, y_{i+c}) = (x_i + c, y_i + \sum_i^{i+c} \Delta y) \\ (x_{i-c}, y_{i-c}) = (x_i - c, y_i + \sum_{i-c}^i \Delta y) \end{cases}, \quad (1)$$

$$K_{j\pm c} = \begin{cases} (x_{j+c}, y_{j+c}) = (x_j + \sum_j^{j+c} \Delta x, y_j + c) \\ (x_{j-c}, y_{j-c}) = (x_j + \sum_{j-c}^j \Delta x, y_j - c) \end{cases}. \quad (2)$$

Ultimately, the values at the position of the fractional coordinates are estimated using bilinear interpolation on a discrete grid of integer coordinates since the offsets Δ are usually fractional and the position coordinates of the convolution kernel are integers. The bilinear interpolation varies as follows:

$$K = \sum_{K'} B(K', K) \cdot K', \quad (3)$$

$$B(K, K') = b(K_x, K'_x) \cdot b(K_y, K'_y). \quad (4)$$

where K denotes the fractional positions of Equations (3) and (4), K' denotes all integral space positions, B denotes the bilinear interpolation kernel and b denotes the one-dimensional kernel of the decomposition.

As shown in Figure 3, DySneakConv covers a 9×9 range during deformation due to the variation in the x - and y -axes. The purpose of this design structure is to allow the convolutional energy to freely adapt to the structure for effective feature extraction while ensuring that it does not deviate too far from the target structure within the defined constraints. Ultimately, more accurate feature learning is achieved.

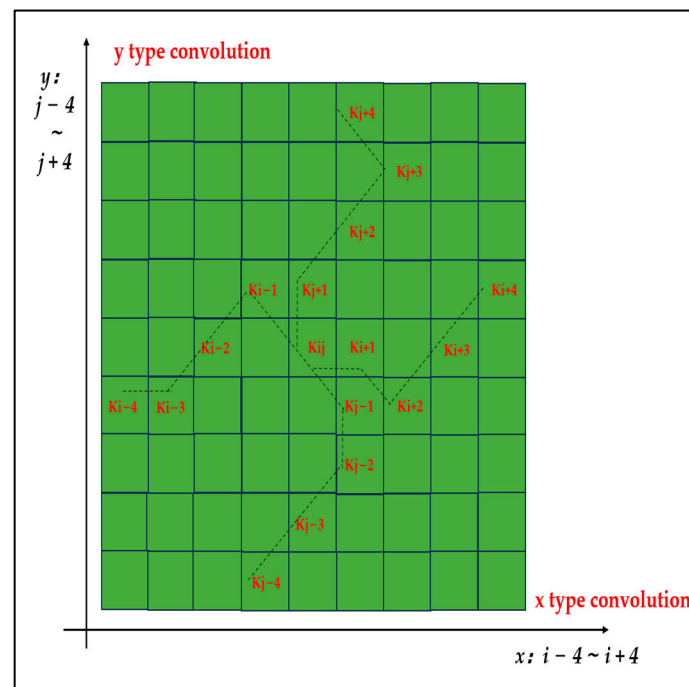


Figure 3. True feeling range of DySneakConv.

2.2.2. AIFI

A module with strong feature fusion capability is needed to deal with and extract features from complex tyre defects in X-ray image datasets that are of various types and different shapes and sizes, and are not easy to distinguish. In the Backbone network, the fourth layer of BottleneckCSP is responsible for processing higher-level semantic features,

but it also considers the interaction between the shallow features and the high-level deep features. This is because shallow features are easily affected by the target traits and other types of external interference, which negatively effects the semantic interaction of the BottleneckCSP. To solve this problem, this paper introduces the AIFI [21] module. Its primary purpose is to improve the efficiency and effectiveness of feature extraction by introducing the internal-scale feature interaction based on the attention mechanism. The core idea is to use the attention mechanism between features of the same scale to promote more comprehensive feature fusion. The structure of AIFI is shown in Figure 4.

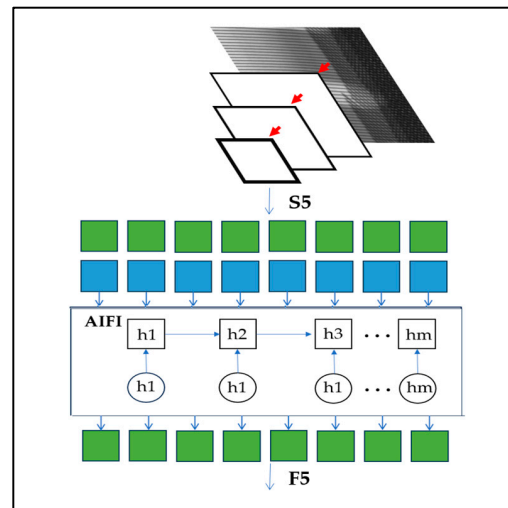


Figure 4. General structure of AIFI.

The essence of AIFI is the Encoder layer of the Transformer, and AIFI only deals with the deeper semantic features of the Backbone network [22]. Its working process is to input the query, key and value vectors into the fully connected layer so that the query and key vectors are multiplied together to produce a scoring matrix. Then, the query and key vectors are scaled by their respective dimensionalities to improve the stability of the gradient. The scaled score is then subjected to softmax computation to obtain the attention weights, and these attention weights are multiplied by the value vectors to obtain the output vectors. Finally, the output vectors of each self-attention vector are synthesized into a single vector. The attention output vector is synthesized into one vector. The working principle of AIFI is shown in Figure 5.

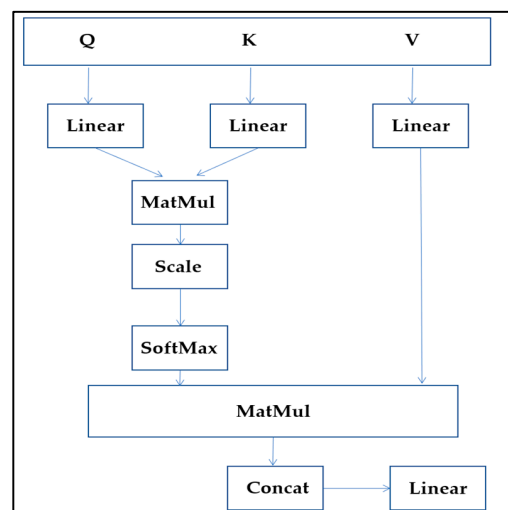


Figure 5. Working principle of AIFI.

AIFI's self-attention mechanism allows the model to consider global contextual information when processing features at each location [23]. This helps to better understand the relationships between different regions in an image. The process of its operation is formulated as follows:

$$\begin{aligned} Q &= K = V = \text{Flatten}(S_5), \\ F_5 &= \text{Reshape}(\text{Attn}(Q, K, V)). \end{aligned} \quad (5)$$

where *Flatten* denotes the conversion of multidimensional input data into one-dimensional vectors, and *Reshape* denotes the restoration of the shape of the features to be the same as S_5 , i.e., the inverse of Flatten. *Attn* denotes the multi-head self-attention mechanism.

In summary, AIFI is based on processing deeper semantic features. It first captures the global contextual information of the image through the structure of the Encoder layer and then makes use of the feature interaction mechanism of the multi-head self-attention mechanism to make the model more effective in processing and fusing important feature information, thus improving the overall detection performance. It is especially suitable for the case of tyre defect detection in X-ray images, where there are various types of targets of different shapes and sizes.

2.3. Improving the Backbone

In YOLOv5m's Neck network, although the up-sample operator is able to convert low-resolution feature maps into high-resolution feature maps, it fails to adequately consider the distances between pixels in the up-sampling process, leading to the problems of fixed up-sampling multiplicity and significant interpolation loss. These problems result in YOLOv5m having a poor up-sampling effect on different sizes of input data. At the same time, bilinear or nearest-neighbour interpolation methods are also prone to cause information loss in up-sampling. To solve this problem, two CARAFE up-sampling operators are introduced in this paper [24]. The CARAFE up-sampling operation possesses a large sensory field and can guide the reorganization process based on the input features. This approach ensures information retention and effectively improves the performance and efficiency of deep learning networks.

The CARAFE interpolation is calculated as follows:

$$I(p) = \sum_q W(p, q) \cdot X(q). \quad (6)$$

where p is the interpolated position, q is the offset of the interpolated position with respect to the original position, and $W(p, q)$ is the weight at the interpolated position p , computed by means of learnable parameters.

The working process is as follows: firstly, the original feature map is up-sampled through the CARAFE interpolation operation to generate the interpolated feature map; secondly, the neighbourhood information of the interpolated feature map is recombined using a content-aware recombination kernel to obtain a more semantically informative output feature map. The model of CARAFE is shown in Figure 6.

CARAFE consists of two parts: up-sampling kernel prediction and feature reorganization. The formulae for these are as follows:

$$W_{l'} = \psi(N(\mathcal{X}_l, k_{\text{encoder}})), \quad (7)$$

$$\mathcal{X}'_{l'} = \phi(N(\mathcal{X}_l, k_{\text{up}}), W_{l'}). \quad (8)$$

$N(X_l, k_{\text{encoder}})$ denotes the $k_{\text{encoder}} \times k_{\text{encoder}}$ neighbourhood centred at l , and ψ is the function used to generate the recombination kernel. $N(X_l, k_{\text{up}})$ denotes the $k_{\text{up}} \times k_{\text{up}}$ neighbourhood centred at l , and ϕ is the content-aware feature recombination module.

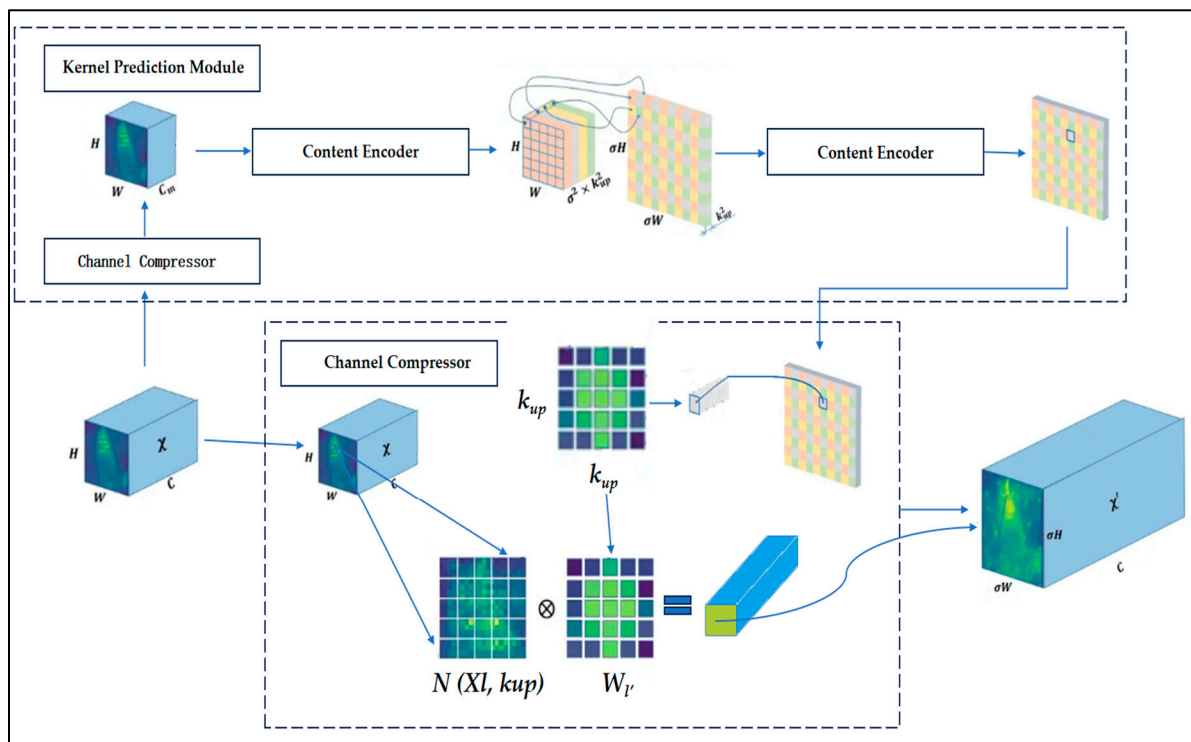


Figure 6. General structure of CARAFE.

In up-sampling kernel prediction, the channels are first compressed by a 1×1 convolution operation to reduce the subsequent computational burden. The prediction of the up-sampling kernel is carried out using a convolutional layer that expands the channel dimension in the spatial dimension to obtain up-sampling kernel prediction that meets the shape requirements. Finally, the weights of the convolution kernel are ensured to be 1 [25] by applying the softmax function for normalization. Therefore, up-sampling kernel prediction improves adaptability to changes in target scale by introducing deformable convolution, which allows the multiplicity of up-sampling to be dynamically adjusted according to the specific location. The interpolation process is handled more carefully using deformable convolution to calculate the positional offset and then using interpolation calculation and weight adjustment, ultimately reducing the interpolation loss.

In feature reorganization, the convolution operation is first performed on the low-resolution feature map to redistribute the information to the high-resolution feature map. The recombined and original high-resolution feature maps are summed element-by-element to achieve the fusion of low-resolution feature map information. Since CARAFE uses the generated content-aware recombination kernel to recombine each pixel position of the input feature map, the dynamic kernel is able to weight the combination of multiple pixels in the neighbourhood, which makes the recombination of each pixel position more accurate, and thus the feature map is able to retain more detail information and edge features during the up-sampling process, which avoids the problem of information loss in the traditional up-sampling operator. This step significantly enhances the expressive power of the feature map and makes the model more effective in capturing the semantic information in the image. Therefore, the interpolation calculation and weight fusion are performed in feature recombination through the introduction of new learnable parameters to ensure that the interpolation process can better preserve the feature information and mitigate the interpolation loss.

The network architecture of the final improved YOLOv5 is shown in Figure 7, where the black boxes show the locations of the improvements.

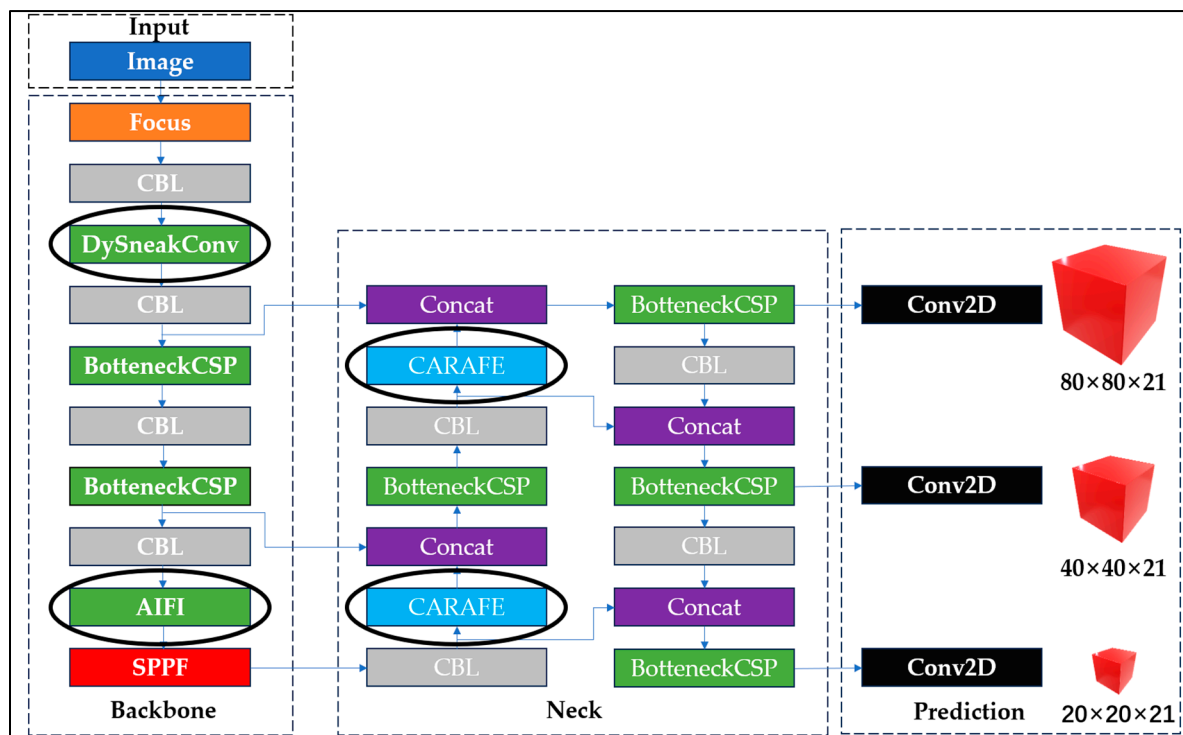


Figure 7. Improved YOLOv5 network framework.

3. Research on a Lightweight Detection Algorithm

Models are often lightweight to cope with deployment requirements in resource-constrained environments. Standard lightweight methods in deep learning include knowledge distillation; channel, layer and neural network pruning; and model quantization. Channel pruning reduces the number of parameters in and the computational complexity of the model by removing redundant channels in the neural network; layer pruning focuses on reducing the depth of the network and streamlines the model by removing unnecessary layers; and neural network pruning methods reduce the size of the model by removing redundant connections or parameters from the network.

Compared with other standard lightweight methods, knowledge distillation possesses a strong generalization ability and a low risk of overfitting. Therefore, knowledge distillation is chosen as the lightweight method in this paper.

3.1. Introduction to Knowledge Distillation

Knowledge distillation achieves model compression by transferring knowledge from one model to another [26]. That is, knowledge from a larger, parameter-heavy model (teacher model) is transferred to a smaller, lightweight model (student model), maintaining the model size while improving detection accuracy. The steps of the distillation process are the preparation of the teacher model and the design of the student model, followed by soft target generation, i.e., using the teacher model to predict the training results of the student model. This soft target helps guide the student model in learning complex feature representations. Finally, the generated soft goals are used for distillation training to guide the student model in understanding the knowledge of the teacher model. The working principle of knowledge distillation is shown in Figure 8.

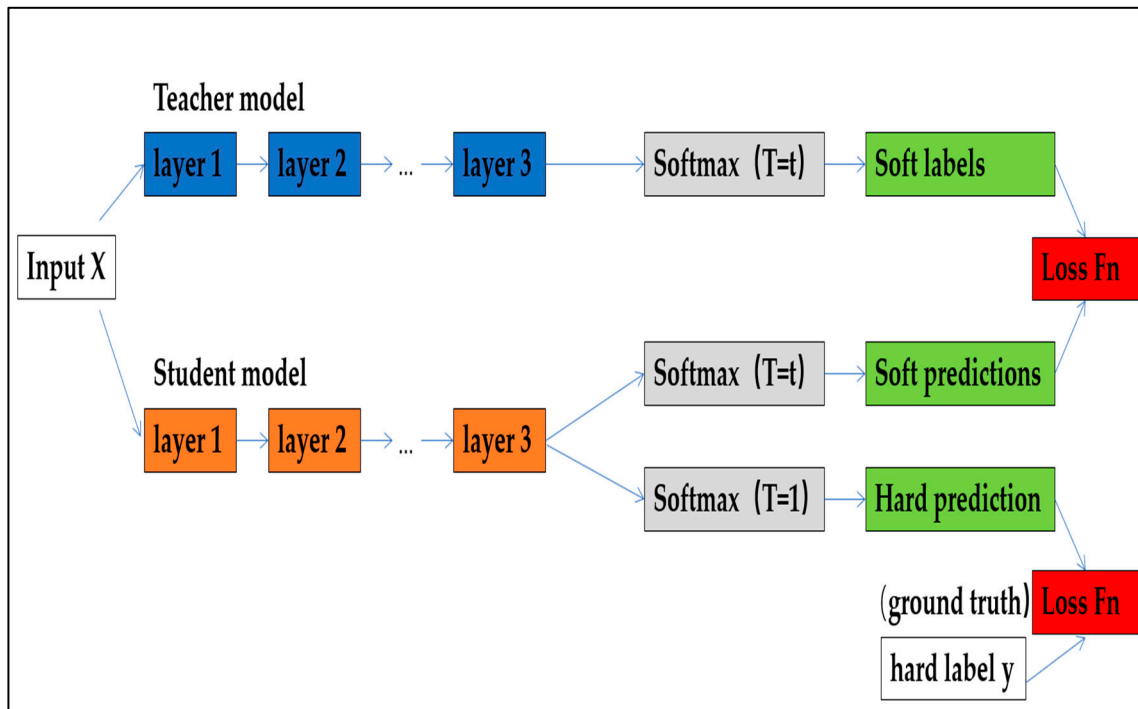


Figure 8. Working principle of knowledge distillation.

The introduction of AIFI, DySneakConv and CARAFE in the previous section improves the accuracy of the YOLOv5m model in the tyre defect detection task. As the complexity of the model increases, this paper distills the knowledge from YOLOv5m to YOLOv5s through the Channel-wise Knowledge Distillation algorithm. This results are a balance between detection accuracy and model size.

3.2. Channel-Wise Knowledge Distillation Algorithm

Channel-wise Knowledge Distillation [27] is a channel-based knowledge distillation algorithm. In contrast to feature-based knowledge distillation algorithms that are based on the overall structure of the feature map, Channel-wise Knowledge Distillation passes the output information of the channels in the teacher model to the corresponding channels in the student model through mapping. The best correspondence between the teacher and student models is obtained by minimizing the distillation loss function after the channels are aligned. The network structure of Channel-wise Knowledge Distillation is shown in Figure 9.

The soft probability map is first obtained by normalizing the feature map of each channel. That is, the activation values of each channel are converted into probability distributions, and the probability distribution metric is used to evaluate the differences between different channels. The activation values are converted as follows:

$$\phi(y_c) = \frac{\exp\left(\frac{y_{c,i}}{T}\right)}{\sum_{i=1}^{W \cdot H} \exp\left(\frac{y_{c,i}}{T}\right)}. \quad (9)$$

where i denotes the pixel position in the channel and T denotes the temperature hyperparameter; the larger the T , the larger the spatial region of interest for each channel.

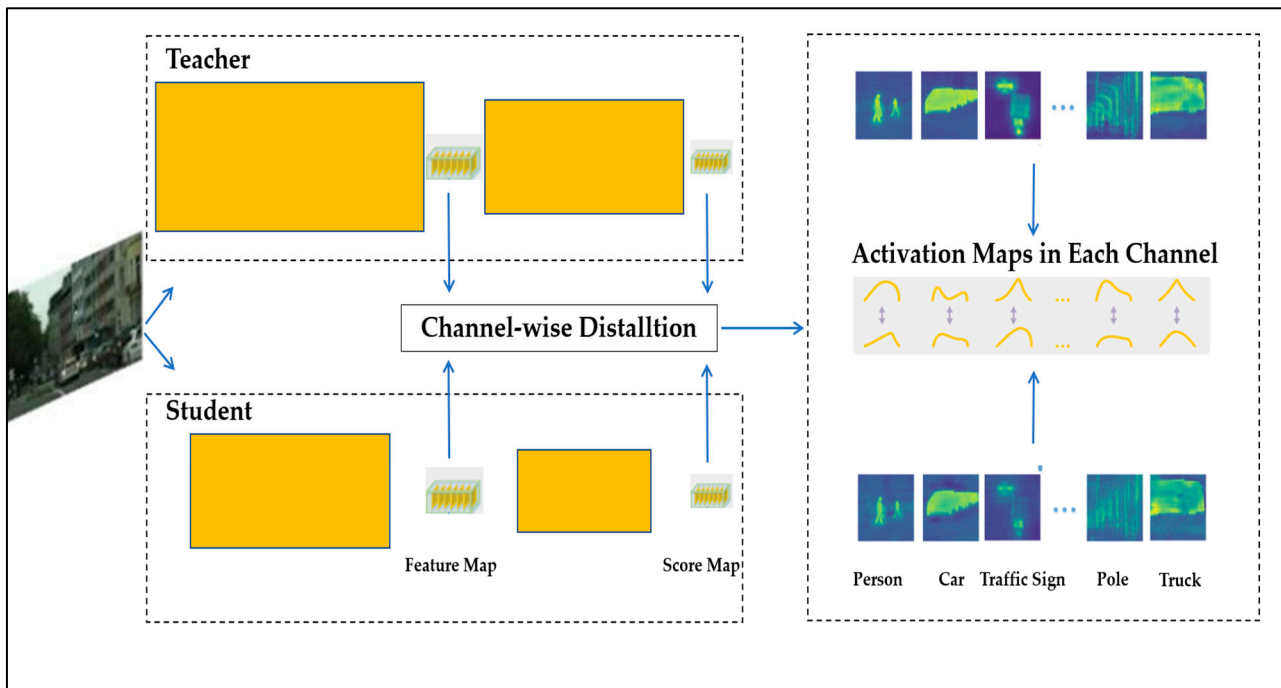


Figure 9. Overall structure of Channel-wise Knowledge Distillation.

The asymmetric Kullback–Leibler (KL) scatter of the probability maps of the two network channels is then minimized [28]. KL scatter was introduced to measure the difference in probability distributions between the two models to guide the student model in learning from the teacher model. Such a transformation makes the activation values of each channel more inclined to encode the salient features of each category. The KL scatter is calculated as follows:

$$\varphi(y^T, y^S) = \frac{T^2}{C} \sum_{c=1}^C \sum_{i=1}^{W \cdot H} \phi(y_{c,i}^T) \cdot \log \left[\frac{\phi(y_{c,i}^T)}{\phi(y_{c,i}^S)} \right]. \quad (10)$$

Finally, the student model is allowed to learn from the teacher model. The predicted transparent category-specific masks are obtained using the trained teacher model. The loss function for this method is as follows:

$$\left(\phi(y^T), \phi(y^S) \right) = \varphi(\phi(y_c^T), \phi(y_c^S)). \quad (11)$$

where y^T and y^S represent the probabilistic soft plots of the teacher and student models, respectively, and ϕ represents how the activation values are converted into probability distributions.

This approach makes the activation maps for each channel focus more on salient regions in the corresponding channel, causing the student network to produce similar activation distributions in foreground salience. In contrast, activations corresponding to background regions of the teacher’s network have less impact on learning.

4. Experimental Evaluation

This section first describes the dataset used in the experiment and the evaluation metrics used to assess performance. Subsequently, the dataset is used to conduct ablation experiments to assess the impact and effectiveness of the improvements.

4.1. Methodology

4.1.1. Experimental Environment and Datasets

In this study, we used Windows 11 22H2 as the operating system, the computer configuration was a 13th Gen Intel (R) Core (TM) i9-13900HX CPU and an NVIDIA GeForce RTX 4070 Laptop GPU, the memory capacity was 16 GB, and the deep learning framework was built using PyTorch2.0.0. The acceleration environment was CUDA11.8 and CUDNN8.9.0. In the training process, the input image size was $\text{img_size} = (384, 384)$; the batch size was $\text{batch_size} = 8$; the number of iterations was $\text{epochs} = 100$; the learning rate was $\text{lr} = 0.01$; and the momentum factor was $\text{momentum} = 0.9385$.

This study used the open-source dataset of tyre X-ray photos produced by the Networked Control Centre of Shanghai University, with a total of 1142 training set images, 382 validation set images and 381 test set images. The training, validation and test data were kept strictly independent to ensure an unbiased evaluation. The dataset included a total of four categories of common tyre defect features: 2_open, cords_defects, impurity and belt_cuobian. Some examples of these defects are shown in Figure 10, where the yellow, blue, red and green boxes show the training results for 2_open, cords_defects, impurity and belt_cuobian, respectively. The image size in the chosen dataset is 480×480 pixels, while the size of the network input image was modified to a resolution of 384×384 pixels.

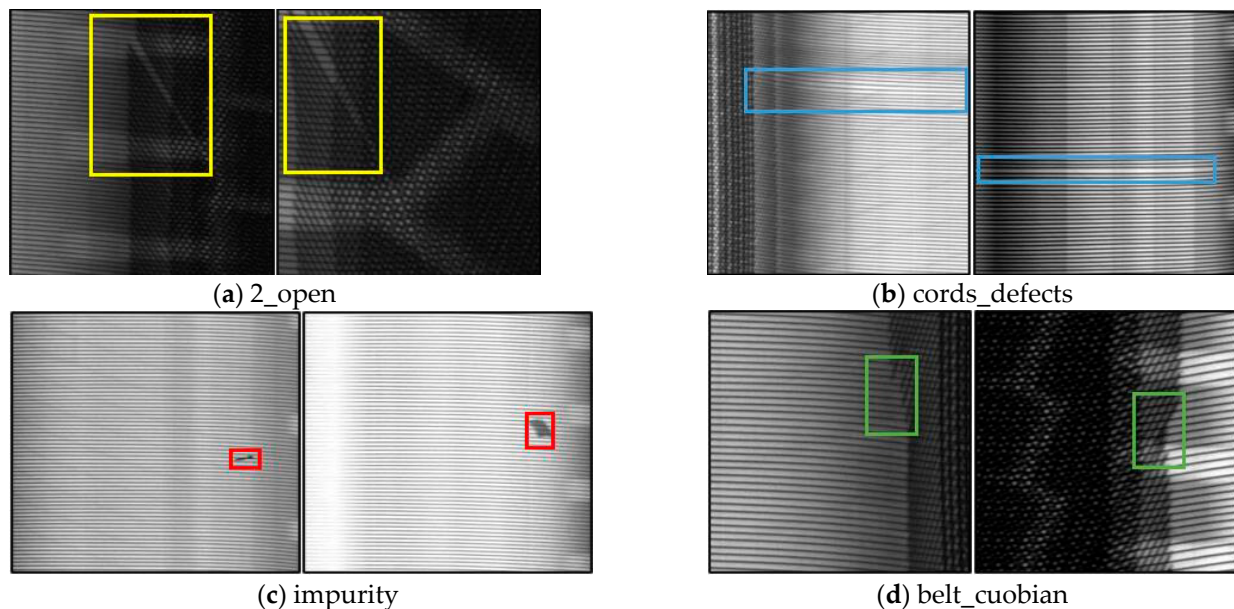


Figure 10. Tyre defect characteristics.

4.1.2. Evaluating Indicator

Four metrics are used in this paper. Mean Average Precision (mAP) is used to assess the accuracy of the model, and the numbers of billion floating-point operations (GFLOPs) and parameters in the model are used to assess its size [29].

mAP is widely used to measure the detection accuracy of network models. It is crucial in algorithm performance metrics such as target location and category prediction. It represents the mean of the average precision (AP) of each category, which provides a more comprehensive picture of the overall performance of the model relative to recall (R) and precision (P). The mean value of average precision is calculated in Equation (12) as:

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^N P_i. \quad (12)$$

where N denotes the number of categories and P_i is the average precision rate for category i .

Model size refers to the amount of space a deep learning model occupies in storage or memory. This metric is influenced by several factors, such as the number of parameters in the model, the model architecture, compression and optimization techniques and data type. Evaluating model size is important for using models in resource-constrained environments such as deployments to embedded devices, mobile devices or edge computing platforms. A smaller model size means lower storage requirements and a lower memory footprint, which can help improve the efficiency and speed of model deployment.

GFLOPs are an essential metric for algorithmic complexity. They indicate how many billions of floating-point operations a model needs to perform. GFLOPs are usually related to the number of layers and parameters included in the model. More extensive models typically have more parameters and complex structures, requiring more computational resources to perform forward and backpropagation operations. This means that larger models usually produce higher values of GFLOPs.

Parameters refer to the number of tuneable parameters in the model that need to be trained. These tuneable parameters are the weights and biases that need to be adjusted for the model to extract features from the data and make predictions through learning. The number of parameters is not only related to the network's structure but also to the depth and width of the network and the specific configuration of each layer. Therefore, the number of parameters is directly related to the complexity of the model and the storage space requirement.

4.2. Results

4.2.1. Comparative Experiment before and after Algorithm Improvement

To compare the detection results of the improved algorithm on the four defect categories, comparative experiments were conducted on YOLOv5s, YOLOv5s + DySneakConv + AIFI + CARAFE, YOLOv5m and YOLOv5m + DySneakConv + AIFI + CARAFE. The results are shown in Table 1.

Table 1. Comparison of the accuracy in identifying tyre defects under different methods.

Method	2_Open	Cords_Defects	Impurity	Belt_Cuobian
YOLOv5s	91.1	51.1	94.8	97.4
YOLOv5s + DySneakConv + CARAFE + AIFI	94.5	72.2	92.6	99.1
YOLOv5m	91.5	57.3	94.1	96.3
YOLOv5m + DySneakConv + CARAFE + AIFI	95.0	72.6	95.9	99.1

Figure 11a–d show the detection results of YOLOv5s, YOLOv5s + DySneakConv + AIFI + CARAFE, YOLOv5m and YOLOv5m + DySneakConv + AIFI + CARAFE, respectively.

The data in the figures show that the detection effect of the improved YOLOv5s is greater than that of YOLOv5s. Similarly, the detection effect of the improved YOLOv5m is greater than that of YOLOv5m, and it is the best among the four methods.

4.2.2. Ablation Experiment

To verify the effectiveness of DySneakConv, AIFI and CARAFE, each model was combined with the original YOLOv5m while using the unmodified YOLOv5m configuration as a basis for comparison. As shown in Table 2, DySneakConv, AIFI and CARAFE improved the mAP0.5 by approximately 1.9 pp, 3.6 pp and 1.4 pp, respectively. Next, the three were integrated in pairs, and the mAP0.5 of DySneakConv + AIFI improved by 4.5 pp, and the mAP0.5 of both the DySneakConv + CARAFE and the AIFI + CARAFE combinations improved by 2 pp. The final combination of YOLOv5m + DySneakConv + AIFI + CARAFE resulted in a 5.9 pp improvement in mAP0.5. The experiments demonstrate that all three improvements enhance the model's detection and recognition ability.

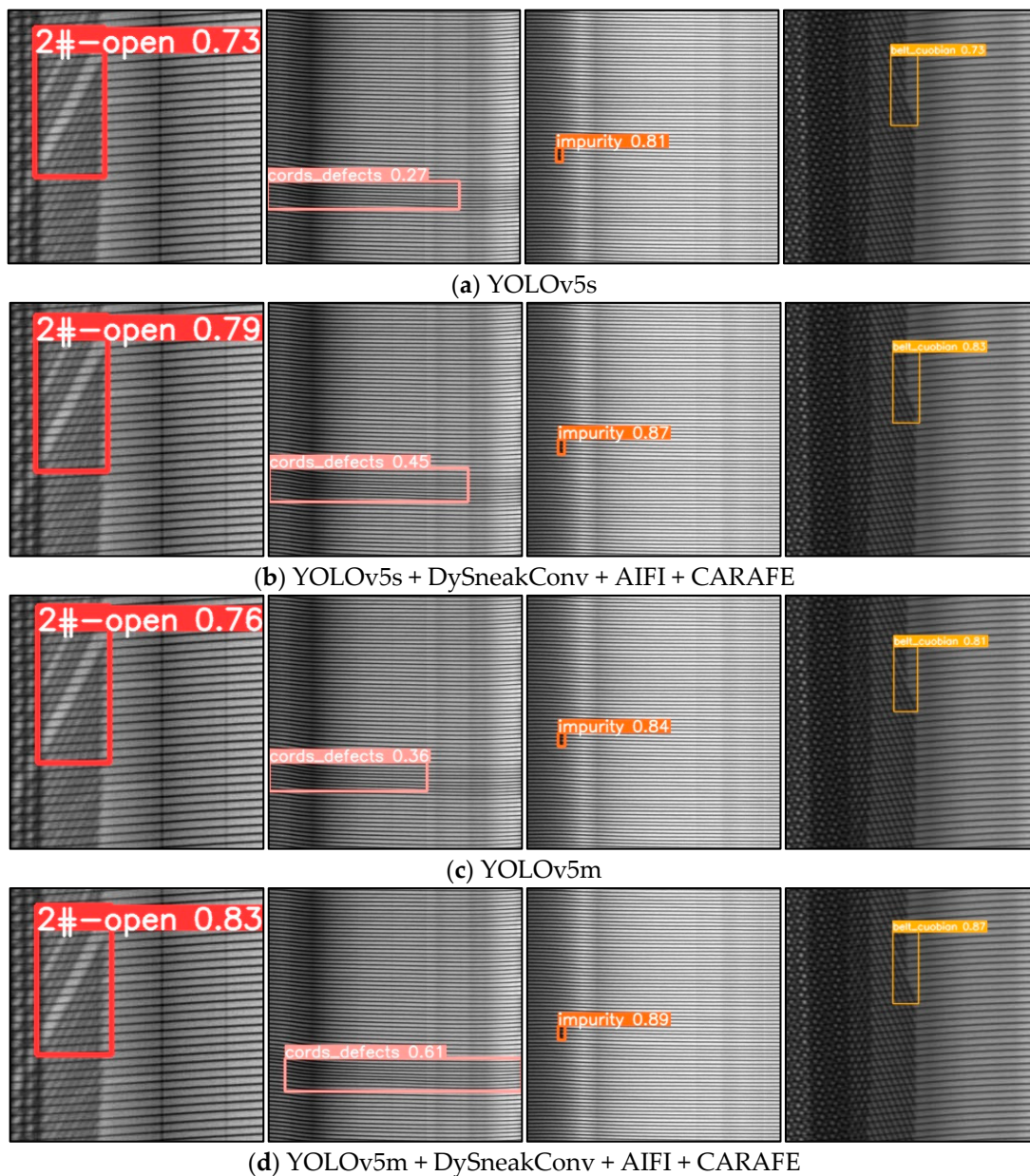


Figure 11. Detection results for the four defect types.

Table 2. Ablation experiments to verify detection accuracy.

Method	mAP0.5/%	Model Size/MB	GFLOPs	Parameters/ 10^6
YOLOv5m	84.8	42.1	47.9	20.87
YOLOv5m + DySneakConv	86.7	47.6	47.4	24.77
YOLOv5m + AIFI	88.4	47.3	47.1	24.61
YOLOv5m + CARAFE	86.2	42.5	48.2	21.02
YOLOv5m + DySneakConv + AIFI	89.3	47.5	49.8	24.69
YOLOv5m + DySneakConv + CARAFE	86.8	46.8	54.7	24.32
YOLOv5m + AIFI + CARAFE	86.8	46.7	47.5	24.28
YOLOv5m + DySneakConv + AIFI + CARAFE	90.7	47.8	50.1	24.85

4.2.3. Lightweight Experiment

Based on the final results in Table 1, YOLOv5m + DySneakConv + AIFI + CARAFE was set as the teacher model for knowledge distillation and YOLOv5s + DySneakConv + AIFI + CARAFE was set as the student model.

Figure 12 shows the training results of YOLOv5m (Figure 12a) and the trained student model (Figure 12b), where the green, white, blue and purple boxes show the training results of mAP0.5, GFLOPs, model size and parameters, respectively.

```

100 epochs completed in 0.373 hours.
Optimizer stripped from runs\train\exp19\weights\last.pt, 42.1MB
Optimizer stripped from runs\train\exp19\weights\best.pt, 42.1MB

Validating runs\train\exp19\weights\best.pt...
Fusing layers...
YOLOv5m summary: 212 layers, 20865057 parameters, 0 gradients, 47.9 GFLOPs

```

Class	Images	Instances	P	R	mAP50	mAP50-95: 100%
all	382	381	0.778	0.825	0.848	0.48
2#-open	382	201	0.762	0.92	0.915	0.6
cords_defects	382	79	0.675	0.468	0.573	0.217
impurity	382	54	0.824	0.952	0.941	0.545
belt_cb	382	47	0.851	0.957	0.963	0.559

Results saved to runs\train\exp19

(a) YOLOv5m

```

100 epochs completed in 0.675 hours.
Optimizer stripped from runs\train\exp17\weights\last.pt, 16.5MB
Optimizer stripped from runs\train\exp17\weights\best.pt, 16.5MB

Validating runs\train\exp17\weights\best.pt...
Fusing layers...
YOLOv5s summary: 186 layers, 8101757 parameters, 0 gradients, 16.6 GFLOPs

```

Class	Images	Instances	P	R	mAP50	mAP50-95: 100%
all	382	381	0.849	0.853	0.894	0.5
2#-open	382	201	0.872	0.946	0.937	0.635
cords_defects	382	79	0.761	0.525	0.722	0.259
impurity	382	54	0.845	0.963	0.926	0.527
belt_cuobian	382	47	0.919	0.979	0.991	0.578

Results saved to runs\train\exp17

(b) Trained student model

Figure 12. Training results of the two algorithms.

YOLOv5m + DySneakConv + AIFI + CARAFE + Channel-wise Knowledge Distillation were set up as trained student models, and the three were compared. By comparing the experimental results of the trained student model with those of the teacher model in Table 3.

It can be seen that although the lightweight treatment using knowledge distillation caused a decrease of about 1.3 pp in mAP0.5, the model size, GFLOPs and parameters were reduced to 31.3 MB, 33.5 and 16.75×10^6 , respectively. Comparing the trained student model with the student model, mAP0.5 is improved by 3.6 pp while all model size parameters are kept constant. Thus, it is demonstrated that the scheme can achieve an ideal balance between accuracy and model size.

Table 3. Lightweight validation of knowledge distillation.

Method	mAP0.5/%	Model Size/MB	GFLOPs	Parameters/ 10^6
YOLOv5m	84.8	42.1	47.9	20.87
Teacher Model	90.7	47.8	50.1	24.85
Student Model	85.8	16.5	16.6	8.10
Trained Student Model	89.4	16.5	16.6	8.10

4.2.4. Using the Improved before and after Algorithms on the Test Set

To confirm the performance of the improved algorithm, we finally compared the trained student model with the original YOLOv5m on a test set. Figure 13 shows the training results for the YOLOv5m (Figure 13a), the trained student model (Figure 13b), where the green, white and purple boxes show the training results of mAP0.5, GFLOPs and parameters, respectively.

```

Fusing layers...
YOLOv5m summary: 212 layers, 20865057 parameters, 0 gradients, 47.9 GFLOPs
val: Scanning D:\YOLOv5\new\last\yolov5-master snake AIFI\Unamusume\datasets\labels\test... 381 images, 0 backgrounds, 0 corrupt: 100% | 381/381 [00:03:00:00, 97.77it/s]
val: New cache created: D:\YOLOv5\new\last\yolov5-master snake AIFI\Unamusume\datasets\labels\test.cache

```

Class	Images	Instances	P	R	mAP50	mAP50-95
all	381	381	0.721	0.837	0.862	0.49
2#-open	381	281	0.633	0.91	0.917	0.621
cords_defects	381	79	0.756	0.51	0.625	0.248
impurity	381	54	0.727	0.926	0.94	0.541
belt_cb	381	47	0.769	1	0.966	0.551

```

Speed: 0.1ms pre-process, 4.7ms inference, 0.9ms NMS per image at shape (1, 3, 384, 384)
Results saved to runs\val\exp13

```

(a) YOLOv5m

```

Fusing layers...
YOLOv5s summary: 186 layers, 8101757 parameters, 0 gradients, 16.6 GFLOPs
val: Scanning D:\YOLOv5\new\last\yolov5-master snake AIFI\Unamusume\datasets\labels\test.cache... 381 images, 0 backgrounds, 0 corrupt: 100% | 381/381 [00:00:00, ?it/s]

```

Class	Images	Instances	P	R	mAP50	mAP50-95
all	381	381	0.802	0.809	0.888	0.502
2#-open	381	281	0.854	0.925	0.958	0.677
cords_defects	381	79	0.877	0.449	0.75	0.315
impurity	381	54	0.699	0.926	0.88	0.479
belt_cuobian	381	47	0.779	0.936	0.965	0.539

```

Speed: 0.1ms pre-process, 8.4ms inference, 0.8ms NMS per image at shape (1, 3, 384, 384)
Results saved to runs\val\exp14

```

(b) Trained student model

Figure 13. Comparison of the two algorithms on a test set.

As can be seen in Table 4, the mAP0.5 of the student model improved by 2.6 pp after training in the test set compared to YOLOv5m. The model size is reduced by 25.6 MB, GFLOPs are reduced by 31.3, and parameters are reduced by 12.7×10^6 , which in turn proves that the trained student model has some generalisation and practicality.

Table 4. Validation results on the test set.

Method	mAP0.5/%	GFLOPs	Parameters/ 10^6
YOLOv5m	86.2	47.9	20.87
Trained Student Model	88.8	16.6	8.10

4.3. Discussion

4.3.1. Comparison with Other Methods

To further confirm the performance of the improved algorithm, the trained student models were compared with YOLOv8s [30], YOLOv8m, YOLOv9 [31] and YOLOv9e, respectively.

Figure 14 shows the training results of YOLOv8s (Figure 14a), YOLOv8m (Figure 14b), YOLOv9 (Figure 14c) and YOLOv9e (Figure 14d), where the green, white, blue and purple boxes show the training results for mAP0.5, GFLOPs, model size and parameters, respectively.

As shown by the experimental results in Table 5, the trained student model not only gained a 2.7 pp increase in mAP0.5 but also obtained a lighter model size of 6 MB when compared with YOLOv8s, and GFLOPs were reduced by 11.8. When compared with YOLOv8m, although the mAP0.5 decreased by 0.7 pp, the model size became 35.5 MB lighter and GFLOPs reduced by 62.1. Compared with YOLOv9, the model size became 105.9 MB lighter while the mAP0.5 stayed the same and GFLOPs reduced by 248.3. Compared with YOLOv9e, mAP0.5 was increased by 8.1 pp, the model size became 343.6 MB lighter and GFLOPs reduced by 226.7. After this study, we plan to continue related research on YOLOv8 or YOLOv9.

```

100 epochs completed in 0.751 hours.
Optimizer stripped from runs\detect\train25\weights\last.pt, 22.5MB
Optimizer stripped from runs\detect\train25\weights\best.pt, 22.5MB

Validating runs\detect\train25\weights\best.pt...
Ultralytics YOLOv8.0.136 Python-3.9.16 torch-2.0.0 CUDA:0 (NVIDIA GeForce RTX 4070 Laptop GPU, 8188MiB)
YOLOv8s summary (fused): 168 layers, 11127132 parameters, 0 gradients, 28.4 GFLOPs

```

Class	Images	Instances	Box(P)	R	mAP50	mAP50-95	100%
all	382	381	0.822	0.8	0.867	0.5	
2#-open	382	201	0.829	0.846	0.912	0.629	
cords_defects	382	79	0.865	0.568	0.74	0.269	
impurity	382	54	0.775	0.891	0.879	0.507	
belt_cuobian	382	47	0.821	0.894	0.937	0.594	

```

Speed: 0.1ms preprocess, 2.0ms inference, 0.0ms loss, 0.8ms postprocess per image
Results saved to runs\detect\train25

```

(a) YOLOv8s

```

100 epochs completed in 0.752 hours.
Optimizer stripped from runs\detect\train30\weights\last.pt, 52.0MB
Optimizer stripped from runs\detect\train30\weights\best.pt, 52.0MB

Validating runs\detect\train30\weights\best.pt...
Ultralytics YOLOv8.0.136 Python-3.9.16 torch-2.0.0 CUDA:0 (NVIDIA GeForce RTX 4070 Laptop GPU, 8188MiB)
YOLOv8m summary (fused): 218 layers, 25842076 parameters, 0 gradients, 78.7 GFLOPs

```

Class	Images	Instances	Box(P)	R	mAP50	mAP50-95	100%
all	382	381	0.889	0.825	0.901	0.537	
2#-open	382	201	0.858	0.905	0.925	0.659	
cords_defects	382	79	0.911	0.517	0.774	0.282	
impurity	382	54	0.846	0.919	0.927	0.562	
belt_cuobian	382	47	0.941	0.957	0.98	0.643	

```

Speed: 0.1ms preprocess, 3.5ms inference, 0.0ms loss, 0.9ms postprocess per image
Results saved to runs\detect\train30

```

(b) YOLOv8m

```

100 epochs completed in 0.799 hours.
Optimizer stripped from runs\train\exp7\weights\last.pt, saved as runs\train\exp7\weights\last_stripped.pt, 122.4MB
Optimizer stripped from runs\train\exp7\weights\best.pt, saved as runs\train\exp7\weights\best_stripped.pt, 122.4MB

Validating runs\train\exp7\weights\best.pt...
Fusing layers...
yolov9 summary: 700 layers, 60763192 parameters, 0 gradients, 264.9 GFLOPs

```

Class	Images	Instances	P	R	mAP50	mAP50-95	100%
all	382	381	0.862	0.84	0.894	0.546	
2#-open	382	201	0.866	0.851	0.907	0.63	
cords_defects	382	79	0.742	0.69	0.737	0.339	
impurity	382	54	0.909	0.923	0.964	0.611	
belt_cuobian	382	47	0.933	0.896	0.969	0.606	

```

Results saved to runs\train\exp7

```

(c) YOLOv9

```

100 epochs completed in 1.023 hours.
Optimizer stripped from runs\train\exp11\weights\last.pt, saved as runs\train\exp11\weights\last_stripped.pt, 140.1MB
Optimizer stripped from runs\train\exp11\weights\best.pt, saved as runs\train\exp11\weights\best_stripped.pt, 140.1MB

Validating runs\train\exp11\weights\best.pt...
Fusing layers...
yolov9-e summary: 1119 layers, 69352952 parameters, 0 gradients, 243.3 GFLOPs

```

Class	Images	Instances	P	R	mAP50	mAP50-95	100%
all	382	381	0.749	0.798	0.813	0.481	
2#-open	382	201	0.785	0.856	0.905	0.617	
cords_defects	382	79	0.478	0.476	0.456	0.161	
impurity	382	54	0.912	0.926	0.945	0.547	
belt_cuobian	382	47	0.819	0.936	0.945	0.599	

```

Results saved to runs\train\exp11

```

(d) YOLOv9e

Figure 14. Training results of other comparison algorithms.

Table 5. Performance validation of student model after training.

Method	mAP0.5/%	Model Size/MB	GFLOPs	Parameters/ 10^6
Trained Student Model	89.4	16.5	16.6	8.10
YOLOv8s	86.7	22.5	28.4	11.12
YOLOv8m	90.1	52.0	78.7	25.84
YOLOv9	89.4	122.4	264.9	60.76
YOLOv9e	81.3	140.1	243.3	69.35

4.3.2. Limitations

In the experiments of this paper, the improved method shows significant advantages in the task of defect detection in tyre X-ray images. However, the study has some limitations.

Firstly, the proposed improved method is mainly optimised for defect detection in tyre X-ray images. This type of image has specific characteristics, such as defects that usually exhibit elongated features. However, it was not verified whether these improvements are equally applicable to other types of images, such as natural scene images or medical images. Therefore, despite the success on tyre X-ray images, its generalisation ability has yet to be further verified.

Secondly, the dataset of this study is mainly derived from the open-source dataset of the Network Control Centre of Shanghai University, and the quality and variety of images in this dataset may be different from those in real applications. For example, tyre X-ray images in industrial production may be subject to different shooting conditions and equipment, which may have an impact on the performance of the detection algorithm. Therefore, our experimental results are somewhat specific, and future research needs to be validated on more diverse and realistic image datasets to fully assess the robustness and practicality of the algorithm.

5. Conclusions

In this paper, we present an improved YOLOv5m algorithm for tyre defect detection, which introduces AIFI to deal with deep semantic features, thus overcoming insignificant interference from defective regions. It also introduces DySneakConv to allow convolutional kernels to freely adapt to the structure for more efficient feature learning, and the introduction of the CARAFE up-sampling operator allows adaptive and optimized recombination kernels to be used at different locations to overcome the problems of fixed up-sampling multiplicity and significant interpolation loss. Finally, the Channel-wise Knowledge Distillation algorithm is introduced to lighten the model. After training and testing the model on the tyre X-ray image dataset, it was shown that the model size was reduced by 25.6 MB, the GFLOPs were reduced by 31.3, the parameters were reduced by 12.7×10^6 , and the mAP0.5 was improved by 4.6 pp compared with YOLOv5m.

The improved model not only improves detection accuracy, it also reduces the parameters and model size, and it can effectively reduce energy consumption. It is easier to deploy on mobile devices, which helps to improve the operational efficiency of tyre defect detection.

In future work, we plan to continue our research using YOLOv8 or YOLOv9 algorithms. Only the four most common tyre defect data types were selected for this study, and our future work will focus on trying to build a more comprehensive tyre defect dataset to improve the usefulness of the model, and trying to optimise the algorithm so that the model is not limited to specific features.

Author Contributions: Conceptualization, H.B. and M.X.; methodology, H.B. and M.X.; software, H.B.; validation, H.B. and M.X.; formal analysis, H.B.; investigation, Z.Z. and C.J.; resources, W.W. and Z.Z.; data curation, H.B. and M.X.; writing—original draft preparation, H.B. and W.W.; writing—review and editing, H.B. and M.X.; visualization, H.B.; supervision, M.X.; project administration, C.J.; funding acquisition, M.X. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Science and Technology Development Plan Project of Jilin Province, grant number 20220201071GX.

Data Availability Statement: This dataset follows the Open Data Licence. The datasets that support the findings of this study are available in Zenodo at 10.5281/zenodo.11381120. These data were derived from the following resources available in the public domain: <https://aistudio.baidu.com/datasetdetail/215731> (accessed on 2 June 2024).

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

References

1. Liu, Y. The Importance of Automotive Tire Safety. *Sci. Consult. (Sci. Technol. Manag.)* **2018**, *69*. [CrossRef]
2. Dharmawan, O.M.D.I.; Lee, J.; Winata, A.P.M.I. Real-time deep-learning-based object detection and unsupervised statistical analysis for quantitative evaluation of defect length direction on magnetooptical faraday effect. *NDT E Int.* **2024**, *145*, 103127. [CrossRef]
3. Saleh, T.; Weng, X.; Holail, S.; Hao, C.; Xia, G.S. DAM-Net: Flood detection from SAR imagery using differential attention metric-based vision transformers. *ISPRS J. Photogramm. Remote Sens.* **2024**, *212*, 440–453. [CrossRef]
4. 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, 24–27 June 2014; pp. 580–587.
5. Girshick, R. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Washington, DC, USA, 7–13 December 2015; pp. 1440–1448.
6. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In Proceedings of the Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; Volume 28.
7. 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: 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016; Part I 14, pp. 21–37.
8. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 24–30 June 2016; pp. 779–788.
9. Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2980–2988.
10. Zhou, X.; Wang, D.; Krähenbühl, P. Objects as points. *arXiv* **2019**, arXiv:1904.07850.
11. Wang, P.; Wang, X.; Liu, Y.; Zhou, P.; Zhao, J. Analysis of Tire Surface Defect Detection Based on YOLOv5 Network. *Automob. Pract. Technol.* **2022**, *47*, 25–30.
12. Wu, Z.; Jiao, C.; Chen, L. Tire Defect Detection Method Based on Improved Faster R-CNN. *Comput. Appl.* **2021**, *41*, 8.
13. Liu, Y.; Liu, X.; Gao, Y. Tire X-ray Image Defect Detection Based on FAMGAN. *J. Electron. Meas. Instrum.* **2023**, *37*, 58–66. [CrossRef]
14. Li, M.; Jiang, J. Tire Defect Detection Algorithm Based on Deep Learning. *Inf. Technol. Informatiz.* **2021**, 52–53. [CrossRef]
15. Liao, K.-C.; Liou, J.-L.; Hidayat, M.; Wen, H.-T.; Wu, H.-Y. Detection and Analysis of Aircraft Composite Material Structures Using UAV. *Inventions* **2024**, *9*, 47. [CrossRef]
16. Shao, Y.; Zhang, D.; Chu, H.; Zhang, X.; Rao, Y. Overview of YOLO Object Detection Based on Deep Learning. *J. Electron. Inf. Technol.* **2022**, *44*, 12.
17. Liu, Y. Research on Deep Learning Based on Up-Sampling Technology. Master's Thesis, Yanshan University, Qinhuangdao, China, 2022.
18. Lv, W.; Zhao, Y.; Xu, S.; Wei, J.; Wang, G.; Dang, Q.; Liu, Y.; Chen, J. DETRs Beat YOLOs on Real-time Object Detection. *arXiv* **2023**, arXiv:2304.08069.
19. Zhu, X.; Su, W.; Lu, L.; Li, B.; Wang, X.; Dai, J. Deformable DETR: Deformable Transformers for End-to-End Object Detection. In Proceedings of the International Conference on Learning Representations, Virtual Event, 3–7 May 2021.
20. Sun, L.; Liu, J.; Wang, J.; Xing, J.; Zhang, Y.; Wang, C. Survey of Vision Transformer in Fine-Grained Image Classification. *Comput. Eng. Appl.* **2024**, *60*, 30–46. [CrossRef]
21. Qi, Y.; He, Y.; Qi, X.; Zhang, Y.; Yang, G. Dynamic Snake Convolution based on Topological Geometric Constraints for Tubular Structure Segmentation. *arXiv* **2023**, arXiv:2307.08388.
22. Dai, J.; Qi, H.; Xiong, Y.; Li, Y.; Zhang, G.; Hu, H.; Wei, Y. Deformable Convolutional Networks. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017.
23. Dong, S.; Zhao, J.; Zhang, M.; Shi, Z.; Deng, J.; Shi, Y.; Tian, M.; Zhuo, C. DeU-Net: Deformable U-Net for 3D Cardiac MRI Video Segmentation. In Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention, Lima, Peru, 4–8 October 2020.
24. Wang, J.; Chen, K.; Xu, R.; Liu, Z.; Loy, C.C.; Lin, D. CARAFE++: Unified Content-Aware ReAssembly of FEatures. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *44*, 4674–4687. [CrossRef] [PubMed]
25. Liu, W.; Wen, Y.; Yu, Z.; Yang, M. Large-Margin Softmax Loss for Convolutional Neural Networks. *arXiv* **2016**, arXiv:1612.02295.

26. Mishra, A.; Marr, D. Apprentice: Using Knowledge Distillation Techniques to Improve Low-Precision Network Accuracy. *arXiv* **2017**, arXiv:1711.05852.
27. Shu, C.; Liu, Y.; Gao, J.; Xu, L.; Shen, C. Channel-wise Distillation for Semantic Segmentation. *arXiv* **2020**, arXiv:2011.13256.
28. Kosheleva, O.; Kreinovich, V. Why Deep Learning Methods Use KL Divergence Instead of Least Squares: A Possible Pedagogical Explanation. *Математические Структуры Моделирование* **2017**, *2*, 102–106.
29. Cao, Y.; Liu, H.; Jia, X.; Li, X. A Review of Image Quality Evaluation Methods Based on Deep Learning. *Comput. Eng. Appl.* **2021**, *57*, 27–36.
30. Ma, S.; Lu, H.; Liu, J.; Zhu, Y.; Sang, P. LAYN: Lightweight Multi-Scale Attention YOLOv8 Network for Small Object Detection. *IEEE Access* **2024**, *12*, 29294–29307. [[CrossRef](#)]
31. Wang, C.Y.; Yeh, I.H.; Liao, H.Y.M. YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information. *arXiv* **2024**, arXiv:2402.13616.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.