algorithms

MDPI

Article

High-Precision Chip Detection Using YOLO-Based Methods

Ruofei Liu [1,2] and Junjiang Zhu [3,*]

[1] Center for Balance Architecture, Zhejiang University, Hangzhou 310028, China; rf_liu2025@163.com
[2] The Architectural Design & Research Institute of Zhejiang University Co., Ltd., Hangzhou 310028, China
[3] College of Mechanical and Electrical Engineering, China Jiliang University, Hangzhou 310018, China
* Correspondence: zhujunjiang@cjlu.edu.cn

Abstract

Machining chips are directly related to both the machining quality and tool condition. However, detecting chips from images in industrial settings poses challenges in terms of model accuracy and computational speed. We firstly present a novel framework called GM-YOLOv11-DNMS to track the chips, followed by a video-level post-processing algorithm for chip counting in videos. GM-YOLOv11-DNMS has two main improvements: (1) it replaces the CNN layers with a ghost module in YOLOv11n, significantly reducing the computational cost while maintaining the detection performance, and (2) it uses a new dynamic non-maximum suppression (DNMS) method, which dynamically adjusts the thresholds to improve the detection accuracy. The post-processing method uses a trigger signal from rising edges to improve chip counting in video streams. Experimental results show that the ghost module reduces the FLOPs from 6.48 G to 5.72 G compared to YOLOv11n, with a negligible accuracy loss, while the DNMS algorithm improves the debris detection precision across different YOLO versions. The proposed framework achieves precision, recall, and mAP@0.5 values of 97.04%, 96.38%, and 95.56%, respectively, in image-based detection tasks. In video-based experiments, the proposed video-level post-processing algorithm combined with GM-YOLOv11-DNMS achieves crack–debris counting accuracy of 90.14%. This lightweight and efficient approach is particularly effective in detecting small-scale objects within images and accurately analyzing dynamic debris in video sequences, providing a robust solution for automated debris monitoring in machine tool processing applications.

Keywords: ghost module; YOLOv11; DNMS; chip detection; video

check for updates

Academic Editors: Marcin Iwanowski and Bogusław Cyganek

Received: 27 May 2025
Revised: 10 July 2025
Accepted: 16 July 2025
Published: 21 July 2025

Citation: Liu, R.; Zhu, J. High-Precision Chip Detection Using YOLO-Based Methods. *Algorithms* **2025**, *18*, 448. https://doi.org/10.3390/a18070448

Copyright: © 2025 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

With the rise of dark factories, automated machining quality monitoring has become an essential requirement in modern manufacturing. To monitor the machining process, online measurements for various physical parameters have been widely used in factories. These parameters include the torque, acceleration, temperature, acoustic emission, displacement, and cutting force, which are critical in ensuring process optimization and product quality [1,2]. Traditionally, experienced workers evaluated the processing quality and machine tool status by examining the shape and number of tuning chips. Recently, many studies have also explored the relationship between chips and processing quality. For example, Tao Chen et al. [3] tested the relationship between the chip morphology and surface quality of products in a high-speed hard-cutting experiment with PCBN tools on hardened steel GCr15. They found that tools with variable chamfered edges produced more regular and stable chips. However, tools with uniform chamfered edges experienced a transition in

chip morphology from wavy to irregular curves over time. Yaonan Cheng et al. [4] found that, in the process of the heavy milling of 508III steel, the debris morphology changed significantly at different stages of tool wear. Firstly, the chips are C-shaped at the beginning of wear; then, the shape changes to strip-like formations as the wear intensifies, ultimately becoming a spiral when the tool experiences severe wear. Research by Vorontsov [5] found that a band-type chip could scratch the workpiece surface or damage cutting edges, while C-shaped chips could affect the surface roughness, and shattered chips tend to wear down the sliding surfaces of the machine tool. However, in practice, although the pixel quality and sampling rates of current cameras meet the requirements for chip monitoring, the automated monitoring of chips in factories to manage the processing conditions is not commonly observed. This is because two key challenges must be addressed: firstly, how to perform the target detection of chips; secondly, how to count the number of chip particles appearing in the video.

The chips produced by machining exhibit diverse shapes and dimensions, including, but not limited to, spiral-shaped fragments and triangular-shaped fragments; this morphological complexity thus renders it challenging for conventional methods like template matching to accurately identify chips in photographic images. With the development of deep learning algorithms, many excellent object detection algorithms have emerged, which may be suitable for the detection of chips and processing of videos [6]. Paper [7] established a correlation between post-detachment chips and the tool wear status using a simple convolutional neural network. In paper [8], various deep learning algorithms, including a CNN, AlexNet, EfficientNetB0, MobileNetV2, CoAtNet-0, and ResNet18, are explored for the monitoring and measurement of wear through images of machining chips. Paper [9] used the cutting temperature and chip characteristics with the neural network BP and LSTM methods to predict the tool life. However, research on accurate video-based chip quantification remains notably scarce in the existing literature. YOLO (You Only Look Once) is one of the most prominent and widely applied models in various fields, such as firefighting [10], monitoring [11], agricultural production [12], defect detection [13], and online wear tool monitoring [14]. The early YOLOv1 significantly improved the detection speed through its end-to-end single-stage detection framework, but its localization accuracy and adaptability to complex scenarios were relatively weak. Subsequent versions introduced optimizations from various perspectives. For instance, YOLOv3 [15] enhanced the detection capabilities for small objects by incorporating multi-scale prediction and residual structures. YOLOv4 [16] integrated self-attention mechanisms and cross-stage partial networks (CSPNet), which improved the feature representation while maintaining real-time performance. YOLOv7 [17] proposed a scalable and efficient network architecture, utilizing reparameterization techniques to achieve parameter sharing. YOLOv8 [18] further optimized the training process by introducing a dynamic label assignment strategy, which enhanced the convergence efficiency. YOLOv9 [19] addressed the limitations of information loss and deep supervision in deep learning by introducing programmable gradient information (PGI) and a generalized efficient layer aggregation network (Gelan). YOLOv10 [20] integrated global representation learning capabilities into the YOLO framework while maintaining low computational costs, significantly enhancing model performance and facilitating further improvements.

In addition to version updates, numerous scholars have made improvements to various iterations of YOLO. For instance, Doherty et al. [21] introduced the bidirectional feature pyramid network (BiFPN) into YOLOv5, enhancing its multi-scale feature fusion capabilities. Jianqi Yan et al. [22] incorporated the convolutional block attention module (CBAM) into YOLOv7 and YOLOv8, while Jinhai Wang et al. [23] integrated the squeeze-and-excitation (SE) module into YOLOv5, thereby improving the model's ability to focus

on critical features. Furthermore, Mengxia Wang et al. proposed DIoU-NMS [24] based on YOLOv5, addressing the issue of missed detection in densely packed objects. These advancements have collectively contributed to the refinement and robustness of the YOLO framework. Moreover, the lightweight design of models has also garnered significant attention. For instance, Chen Xue et al. [25] proposed a sparsely connected asymptotic feature pyramid network, which optimized the architectures of YOLOv5 and YOLOv8. Yong Wang et al. [26] combined the PP-LCNet backbone network with YOLOv5, effectively reducing the model's parameter count and computational load. Among various lightweight strategies, the ghost module [27] has emerged as a notable technique, capable of generating the same number of feature maps as conventional convolutional layers but with significantly reduced computational costs. This module can be seamlessly integrated into YOLO networks to minimize the computational overhead. Previous studies have demonstrated the applicability of the ghost module in versions such as YOLOv5 and YOLOv8, achieving a reduction in model parameters while maintaining detection accuracy [28,29]. However, to our knowledge, no research has yet explored the impact of integrating the ghost module into the more recent YOLOv11. Additionally, it is noteworthy that, starting from version 10, YOLO introduced the dual label assignment and consistent matching metric strategy [23]. Benefiting from this strategy, we can choose to use the original one-to-many head for training and then perform inference using the non-maximum suppression (NMS) algorithm or use a one-to-one detection head to directly obtain the inference results (also referred to as NMS-free training). Although NMS-free training results in lower inference latency, according to the literature [23], it requires more one-to-one matching discriminative features and may reduce the resolution of feature extraction.

Machining chips exhibit significant variations in size and shape. Furthermore, the complex trajectories of flying chips during machining processes make accurately determining chip counts from video footage a significant computational challenge. In this paper, we introduce the ghost module into YOLOv11 and continue to employ the traditional one-to-many training approach. Additionally, we propose a novel dynamic non-maximum suppression (DNMS) algorithm to improve the accuracy of chip detection. Moreover, we present a post-processing method for chip counting in dynamic video sequences. The main novelties and contributions of this paper can be summarized as below:

- We compare the performance of different YOLO versions in detecting debris in images of machining processes;
- A ghost module is introduced into the backbone of the standard YOLOv11 to reduce the computation;
- A dynamic non-maximum suppression algorithm is proposed to enhance the accuracy in identifying small objects—in this case, chips;
- Based on the rising edge signal trigger mechanism, a video-level post-processing algorithm is developed to automatically count the number of chips that fall within the video.

The remainder of the paper is organized as follows: Section 2 summarizes the structure of GM-YOLOv11 and presents the dynamic non-maximum suppression algorithm and the video-level post-processing algorithm in detail. An analysis of the proposed algorithms and the experimental results are provided in Section 3. Other details related to the proposed algorithms are given in Section 4. Finally, Section 5 concludes this paper.

2. Materials and Methods

2.1. Model Architecture

2.1.1. YOLOv11

YOLO (You Only Look Once) is a one-stage object detection algorithm. Its core idea is to model the object detection task as an end-to-end regression problem. It directly predicts the bounding boxes and class probabilities of all objects in an image through a single forward pass. Compared to algorithms like R-CNN, YOLO can leverage global contextual information and reduce computational resource usage. Thus, it is suitable for real-time monitoring tasks. YOLOv11 introduces numerous improvements over its predecessors compared to the previous version. The key advantages of YOLOv11 include better feature extraction, optimized efficiency and speed, fewer parameters with higher accuracy, cross-environment adaptability, and support for a wide range of tasks. The same version of YOLO incorporates suffixes like n, s, m, l, and x (denoting nano, small, medium, large, and extra large, respectively) to represent models of varying depth, width, size, and computational complexity. The architecture of YOLOv11n is shown in Figure 1.

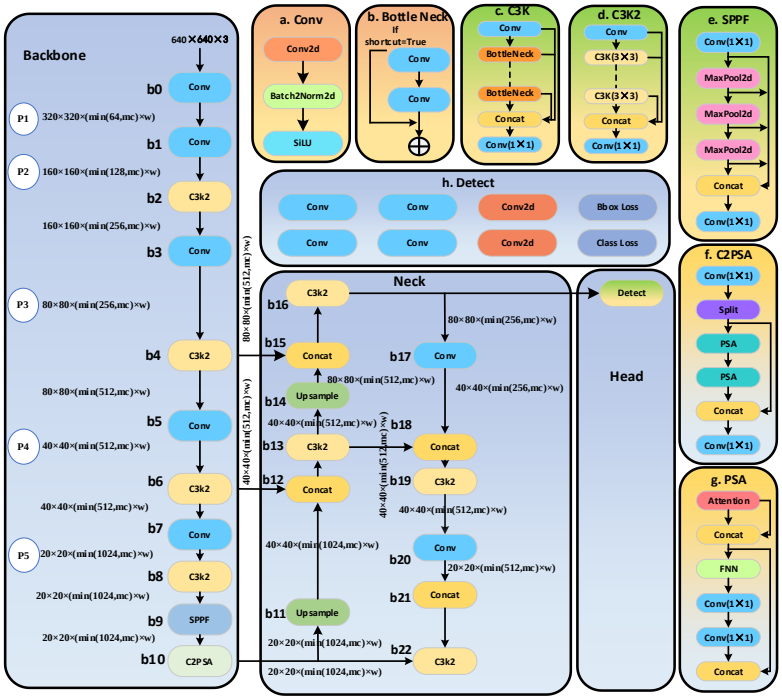


Figure 1. Structure of YOLOv11n.

The detailed network structure and model parameter information of YOLOv11n used in this paper are shown in Figure 1. YOLOv11n consists of three critical components: the backbone, neck, and head. The backbone is responsible for extracting key features at different scales from the input image. This component consists of multiple convolutional blocks (Conv), each of which contains three sub-blocks, as illustrated in the 'a' block in Figure 1: Conv2D, BatchNorm2D, and the SiLU activation function to mitigate the effects of gradient vanishing. In addition to Conv blocks, YOLOv11n also includes multiple C3K2 blocks, which replace the C2f blocks used in YOLOv8, optimizing the cross-stage partial (CSP) design, thereby reducing computational redundancy in YOLOv11n, as shown in the 'd' block in Figure 1. The C3K2 blocks provide a more computationally efficient implementation of CSP. The final two modules in the backbone are spatial pyramid pooling fast (SPPF) and cross-stage partial with spatial attention (C2PSA). The SPPF module utilizes multiple max-pooling layers (as shown in Figure 1e) to efficiently extract multi-scale

features from the input image. On the other hand, as depicted in the 'f' block in Figure 1, the C2PSA module incorporates an attention mechanism to enhance the model's accuracy.

The second major structure of YOLOv11n is the neck. The neck consists of multiple Conv layers, C3K2 blocks, Concat operations, and upsampling blocks, leveraging the advantages of the C2PSA mechanism. The primary function of the neck is to aggregate features at different scales and pass them to the head structure.

The final structure of YOLOv11n is the head, a crucial module responsible for generating prediction results. It determines object categories, calculates objectness scores, and accurately predicts the bounding boxes of identified objects. It is worth noting that YOLOv11n, being the smallest model in the YOLOv11 series, has only one detect layer, while the YOLOv11s, YOLOv11m, YOLOv11l, and YOLOv11x models all feature three detect layers.

From the structure of YOLOv11n, it is evident that YOLOv11n achieves efficient multi-scale feature extraction and fusion by optimizing the design of the backbone, neck, and head. Its structural highlights include the C3K2 blocks, the SPPF module, and the C2PSA attention mechanism. Through a size-specific model pruning strategy, YOLOv11n significantly enhances the resource efficiency while maintaining high accuracy. In this paper, YOLOv11n is used to extract not only chips but also workpieces. Moreover, the predicted location of the workpiece is used to improve the results of chip detection.

2.1.2. Ghost Module

The ghost module is a model compression technique [27]. Compared to traditional convolutional modules, the ghost module reduces the overall computational costs by controlling the number of filters in the first part of the standard convolution and generating additional feature maps using low-cost linear operations. This approach reduces the number of parameters and the computational complexity without altering the size of the output feature maps. The ghost module consists of three steps: primary convolution, ghost generation, and feature map concatenation. The structure of the ghost module is illustrated in Figure 2.

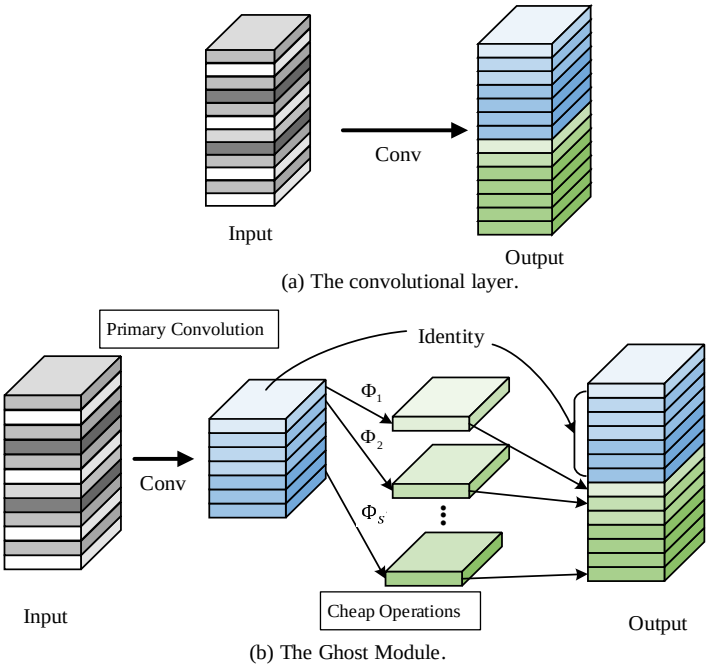


Figure 2. CNN and ghost module: (**a**) operation of CNN, (**b**) operation of ghost module.

Figure 2a illustrates the computational approach of a standard convolutional network. The relationship of input $X \in \mathbb{R}^{c \times h \times w}$ and output $Y \in \mathbb{R}^{h' \times w' \times n}$ can be expressed as

$$Y = X * f + b \tag{1}$$

where * is the convolution operation, b is the bias term, c is the number of input channels, and h and w are the height and width of the input data. $h' \times w'$ is the spatial size of the output feature map. n is the number of filters. $f \in \mathbb{R}^{c \times k \times k \times n}$ is the convolution filter in this layer. $k \times k$ is the kernel size of the convolution filters. Compared to the standard convolutional network, the ghost module reduces the number of required parameters and the computational complexity. Its implementation involves a primary convolution process and a cheap operation process, as illustrated in Figure 2b. In the primary convolution, m (where $m \leq n$) intrinsic feature maps $Y' \in \mathbb{R}^{h' \times w' \times m}$ are generated, as formulated in Equation (2):

$$Y' = X * f' \tag{2}$$

where $f' \in \mathbb{R}^{c \times k \times k \times m}$ is the utilized filter. As formulated in Equation (3), next, a series of cheap linear operations is applied on each intrinsic feature in Y' to generate s ghost features.

$$y_{i,j} = \Phi_{i,j}(y'_i), \ \forall \ i = 1, \ldots m, j = 1, \ldots s \tag{3}$$

where y'_i is the intrinsic feature map in Y', and $\Phi_{i,j}$ is the j-th linear operation. As specified in [27], $d \times d$ linear convolution kernels of $\Phi_{i,j}$ are required to maintain consistent dimensions for distinct indices i and j. s can be obtained by $n = m \cdot s$. By consolidating the $y_{i,j}$ terms in Equation (3), we derive the aggregated output tensor $Y^{\mathrm{g}} = [y_{1,1}, y_{1,2} \ldots y_{m,s}]$. This demonstrates that the standard convolution output Y and the ghost module output Y^{g} exhibit identical dimensionality. Consequently, the ghost module achieves plug-and-play compatibility, serving as a drop-in replacement for conventional convolutional layers. Moreover, the computational cost ratio r_s and parameter ratio r_c between standard convolution and ghost convolution are expressed as in Equations (4) and (5), respectively:

$$r_s = \frac{n \cdot h' \cdot w' \cdot c \cdot k \cdot k}{\frac{n}{s} \cdot h' \cdot w' \cdot c \cdot k \cdot k + (s-1) \cdot \frac{n}{s} \cdot h' \cdot w' \cdot d \cdot d} = \frac{c \cdot k \cdot k}{\frac{1}{s} \cdot c \cdot k \cdot k + \frac{s-1}{s} \cdot d \cdot d} \approx \frac{s \cdot c}{s + c - 1} \approx s \tag{4}$$

$$r_c = \frac{n \cdot c \cdot k \cdot k}{\frac{n}{s} \cdot c \cdot k \cdot k + (s-1) \cdot \frac{n}{s} \cdot d \cdot d} \approx \frac{s \cdot c}{s + c - 1} \approx s \tag{5}$$

As revealed in Equations (4) and (5), the parameter s determines the degree of model compression: the tensor size of intrinsic feature maps is compressed to $\frac{1}{s}$ compared with that of standard convolution modules, since the magnitudes of d and k are similar and $s \ll c$.

From Equations (2) and (3), it can be observed that the ghost convolution incorporates two hyperparameters: the kernel size ($d \times d$) and the number of ghost features generated from a single intrinsic feature map s.

2.1.3. GM-YOLOv11

In both the backbone and head of YOLOv11, multiple convolutional layers are included. To reduce the computational load of YOLOv11, we replace the convolutional layers in the backbone and head with ghost modules. We design three models by replacing only the backbone, only the head, and replacing both entirely, named GM-YOLOv11-backbone, GM-YOLOv11-head, and GM-YOLOv11, respectively. The specific architectures are illustrated in Figures 3–5.

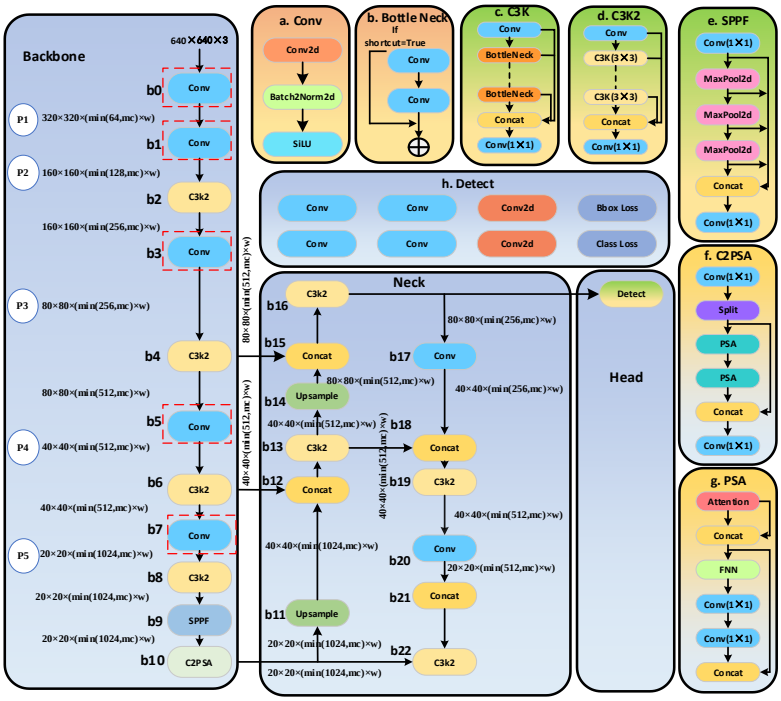


Figure 3. Structure of GM-YOLOv11-backbone.

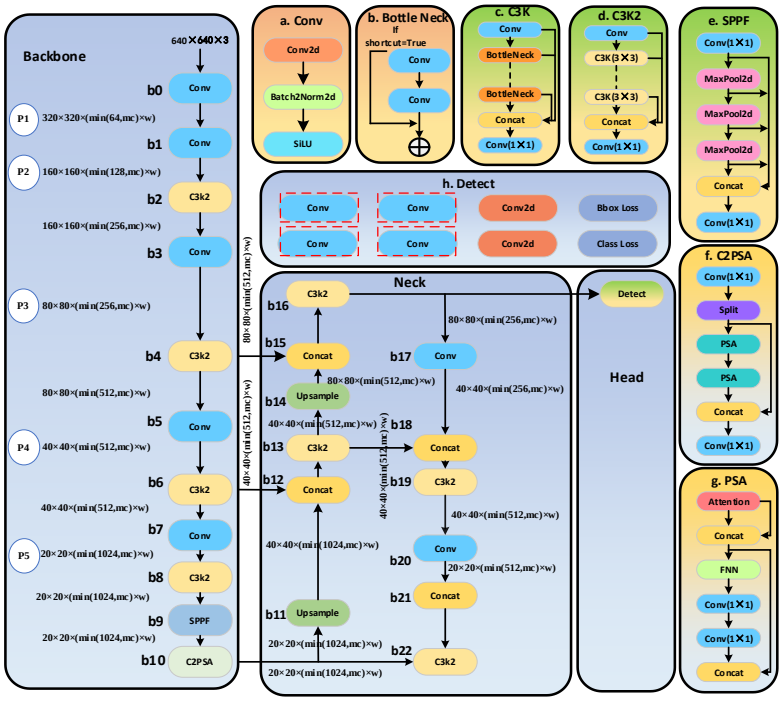


Figure 4. Structure of GM-YOLOv11-head.

The detailed network architectures and parameter configurations of GM-YOLOv11-backbone, GM-YOLOv11-head, and GM-YOLOv11 used in this study are illustrated in Table A1, Table A2, and Table A3, respectively. By comparing Figures 3–5 with Figure 2, it is evident that GM-YOLOv11-backbone integrates five ghost modules into the backbone of the standard YOLOv11n model, while GM-YOLOv11-head introduces four such modules, and GM-YOLOv11 incorporates nine. This modification results in an increase in the number of layers but a reduction in the total number of parameters. Specifically, the standard YOLOv11n comprises 319 layers, whereas GM-YOLOv11-backbone, GM-YOLOv11-head,

and GM-YOLOv11 have 339, 327, and 347 layers, respectively. In terms of parameters, the standard YOLOv11n contains 2,590,230 parameters, while GM-YOLOv11-backbone, GM-YOLOv11-head, and GM-YOLOv11 are optimized to 2,354,294, 2,500,470, and 2,264,534 parameters, respectively. According to reference [27], the kernel size for the cheap operation is set to $d = 3$, and the mapping ratio for the ghost module is set to $s = 2$. Notably, the architectures of the C3K, C3K2, bottleneck, Conv2d, SPPF, C2PSA, and PSA modules remain unchanged, as their effectiveness has been well established in their respective roles. Maintaining these components ensures the preservation of the model's performance and stability, allowing us to focus on optimizing other aspects of the network. Furthermore, their proven efficiency contributes to achieving a desirable balance between accuracy and computational complexity.

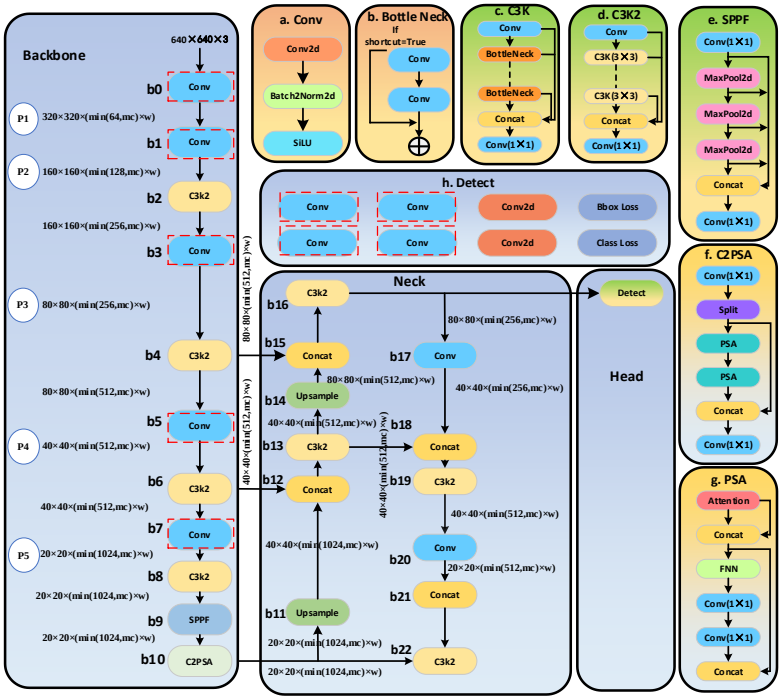


Figure 5. Structure of GM-YOLOv11.

2.2. Dynamic Non-Maximum Suppression (DNMS) Algorithm

It can be seen that, starting from version 10, YOLO provides a non-NMS method. However, since chips do not occupy many pixels, this paper still adopts a one-to-many head for chip detection to maintain the resolution of feature extraction. Moreover, in this paper, YOLO is used to extract not only chips but also workpieces. The detected positions of workpieces are used to develop DNMS. Our aim is to count the chips appearing in the video. Considering that the debris on the workpiece and away from the workpiece may be different in shape and scale, we develop a DNMS method to enhance GM-YOLOv11 and help to distinguish debris in terms of leaving debris and left debris.

2.2.1. Leaving Debris and Left Debris

Observing the machining process of the workpiece, it can be seen that, when a chip is generated on the workpiece, it typically forms a continuous curl. This type of chip is referred to as a "leaving chip". After the chip breaks away from the workpiece, it is influenced by the high-speed rotation of the workpiece and flies off in various directions, which we term "left chips". The left chip may predominantly be curled, but some of it can also appear as small, fragmented, and complex-shaped particles. While a leaving chip overlaps with the workpiece, it is especially important to note that the chip shares the

same material as the workpiece, resulting in consistent colors. Therefore, this study sets different non-maximum suppression thresholds based on whether the chip overlaps with the workpiece pixels, as shown in Figure 6b.

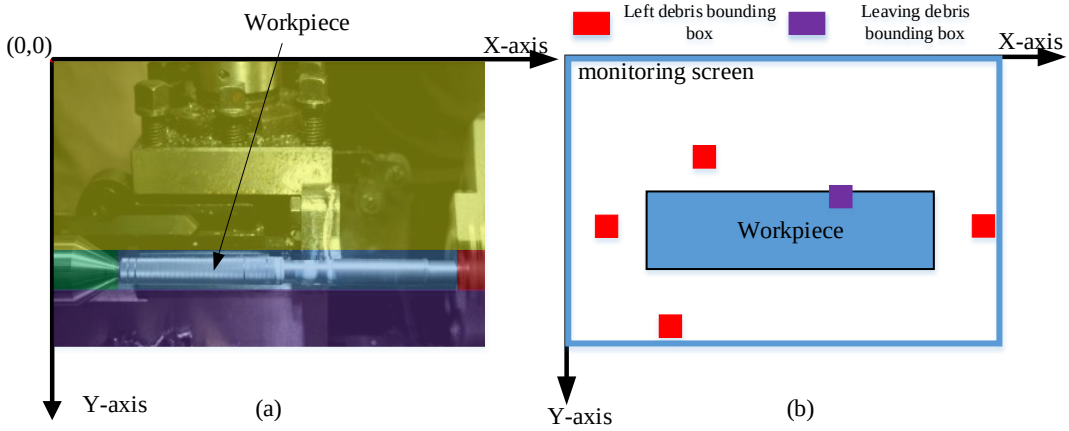


Figure 6. Leaving chips and left chips: (**a**) the workpiece processing site, (**b**) pixel size information.

2.2.2. Box Size Adjustment

During the non-maximum suppression (NMS) process, bounding boxes are suppressed by calculating the IoU values between the ground truth and the predicted bounding boxes. The formula for the standard IoU computation is shown in Equation (6):

$$\text{IoU} = \frac{Area\ of\ Overlap}{Area\ of\ Union} \tag{6}$$

From Equation (6), it can be observed that the IoU value is related to both the area of the ground truth and the area of the predicted bounding box. However, the chips are usually very small; as a result, when the predicted bounding boxes deviate from their true positions, the NMS algorithm can easily lead to false positives or missed detections, especially in scenarios with overlapping objects or localization errors. In actual processing, in most images, there will be a leaving chip. When the chip is ejected, the most likely situation in the image is the presence of one leaving chip and one left chip. Since the breaking of chips is a random behavior, the possibility of multiple left chips appearing in images where chips have broken off is also very high. Based on this possibility, this paper first introduces a scaling mechanism for the size of the predicted bounding box. Through this adjustment, the monitoring results can be better combined with the actual conditions.

In this paper, the size of the predicted bounding box of the chips (denoted as Bd) will be adjusted according to the relative positions of the ground truth box of the workpiece (denoted as BW^{gt}) and the chip box. Suppose that $\left(xw_c^{gt}, yw_c^{gt}\right)$ and (xd_c, yd_c) are the center points of BW^{gt} and Bd, respectively. ww^{gt} and hw^{gt} are the width and height of BW^{gt}, and wd and hd are the width and height of Bd. A ratio is used as the adjustment coefficient, and the new width and height of Bd (wd', hd') are formulated as

$$wd' = \begin{cases} wd + ratio * (1 - wd)\ if\ wd < 0.9 \\ wd,\ if\ wd \geq 0.9 \end{cases} \tag{7}$$

$$hd' = \begin{cases} hd + ratio * (1 - hd)\ if\ hd < 0.9 \\ hd,\ if\ hd \geq 0.9 \end{cases} \tag{8}$$

where the ratio is determined by Equation (9):

$$\text{ratio}_i = \begin{cases} 0 \ \ if \ \ \sum \text{IoU}\left(Bd_i, BW^{gt}\right) = 0 \\ 0.1 \text{ if } \ \sum \text{IoU}\left(Bd_i, BW^{gt}\right) > 0 \text{ and } i = \text{argmin}\left(\min\left(\left|xw_c^{gt} - xd_c{}^i\right|, \ \left|yw_c^{gt} - yd_c{}^i\right|\right)\right) \end{cases} \tag{9}$$

For each predicted bounding box of chip Bd_i, $\text{IoU}\left(Bd_i, BW^{gt}\right)$ is the IoU of Bd_i and BW^{gt}, which can be calculated by Equation (6).

When a leaving chip is not detected in the image, the bounding box that is most likely to be the leaving chip can be adjusted. By enlarging it, the suppression of surrounding leaving chip bounding boxes can be strengthened.

2.2.3. Dynamic Non-Maximum Suppression Algorithm

Since the left chip may be smaller, when the distance between two chips is relatively small, the likelihood of overlapping bounding boxes between different fragments is higher. Therefore, a larger threshold can be used. However, for leaving chips, all their bounding boxes are near the workpiece, and, in most cases, there is only one chip. Thus, a slightly smaller threshold can be used. Additionally, given that leaving chips will overlap with the workpiece, we also use a new indicator, pd, instead of the confidence, to find the first bounding box of the chip, as well as a soft threshold, *ths*, to determine whether to delete the bounding box. They are described as follows:

$$pd = \alpha * IoU\left(Bd_i, BW^{gt}\right) + \beta * cd \tag{10}$$

$$ths = \begin{cases} a1, when \ IoU\left(Bd, BW^{gt}\right) = 0 \\ a2, when \ IoU\left(Bd, BW^{gt}\right) > 0 \end{cases} \tag{11}$$

where α and β are the regular parameters, *cd* is the confidence for the bounding box, $a1$ is the left chip threshold, and $a2$ is the leaving chip threshold. Based on the aforementioned ideas, this paper designs the DNMS process, as shown in Table 1.

Table 1. Algorithm of dynamic non-maximum suppression.

Input: output of YOLO net Output: all chips and their locations
(1) Find BW^{gt} using the standard non-maximum suppression algorithm *(2) Adjust the size of Bd_i according to Equations (7)–(9)* *(3) Find the bounding box Bd_k with the highest score of pd according to Equation (10)* *(4) Calculate $IoU(Bd_k, Bd_j)$ $j \neq k$, as well as ths_j; if $IoU(Bd_k, Bd_j) > ths_j$, delete bounding box Bd_j* *(5) Repeat the sorting and soft threshold operation for the remaining predicted bounding boxes and perform steps (3) and (4) until no more boxes can be deleted*

From these steps, we can see that the input of DNMS is the output of YOLO (including all the predicted boxes and their confidence probabilities for the workpiece and chips), which we can directly obtain from the YOLO network. The prediction boxes of chips are classified as a leaving chip boxes and left chip boxes according to their IoUs with the workpiece box.

2.3. Video-Level Post-Processing Algorithm (VPPA)

The DNMS algorithm has enhanced the capabilities of YOLO to detect chips. However, when processing videos, YOLO processes them as a sequence of continuous frame images, detecting each one individually. A chip typically falls off the workpiece surface and exits the frame in about 0.13 s, appearing in several consecutive frames. Therefore, relying solely

on the original YOLO algorithm to count the number of left chips in the video can lead to large errors. To address this issue, this paper proposes a video post-processing algorithm that utilizes a rising edge signal trigger based on the prediction result as follows.

The post-processing algorithm in this paper uses the leaving chips detected by the YOLO algorithm as a trigger and calculates the chip quantity based on the number of left chip occurrences throughout the triggered frames. During detection, we only verify the presence of leaving chips in each frame (without considering the quantity), whereas, for left chips, we track the number of occurrences in every frame. Therefore, this method is applicable to statistical tasks under both single-target and multi-target scenarios. The specific steps are as follows. The process begins with the preliminary processing of the prediction results of YOLO into temporal information, where the horizontal axis represents the time and the vertical axis indicates the number of predictions. Figure 7 illustrates the complete procedure of YOLO in detecting the video frames. On the time axis, the horizontal coordinate represents the timeframe, while the corresponding vertical coordinate shows the number of prediction results. YOLO detects crack chips in each frame and generates prediction results. For instance, the first frame may detect zero chips, yielding a prediction flag of 0; the 2nd to 5th frames all contain one or more leaving chips, resulting in a prediction flag of 1; finally, the last frame may again show no detected leaving chips, reducing the prediction flag back to 0.

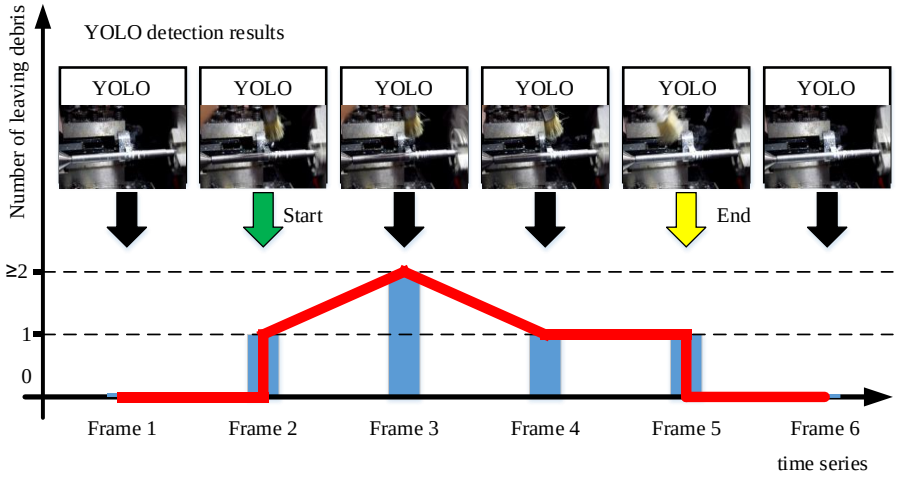


Figure 7. Video-level post-processing algorithm.

Regarding the prediction results of the 6-frame image, 4 prediction boxes are identified, indicating the presence of 4 chips. However, upon closer examination, it becomes clear that the chips predicted in frames 2, 3, 4, and 5 are the same. The sixth frame's left frame is key in determining the number of ejected chips. If the sixth frame contains w (where w $\geq$ 1) left chips, then the number of ejected chips is recorded as w. Otherwise, it is counted as 1. As shown in Figure 7, the prediction results on the time axis reveal that, in the 2nd frame, the prediction increases from 1 to 2, creating a distinct rising edge signal. The prediction results for frames 3, 4, and 5 remain consistent with that for the 2nd frame, with no additional leaving chips detected; hence, no further rising edge signals are generated.

3. Results

3.1. Data Resources

This study conducted two experimental campaigns. Chip images were systematically acquired through turning operations, with video data collected under diverse machining parameters. The first experimental dataset was captured using a Pentax KS2 digital camera

equipped with a Pentax DA 16–45 mm f/4 ED AL lens, configured at a 40 mm focal length. The shooting distance between the camera and the workpiece was approximately 500 mm, and 73 videos capturing machine tool cutting chips were recorded, with a total duration of about 4 h. The pixel size was 720 × 480 for each frame image. In the turning experiments, the turning machine tool was the CA6140A horizontal lathe, and the chip image data were primarily collected from the chips produced during the outer circle turning process. The cutting depth was 1.5 mm, with a rotation speed of 260 r/min and a cutting speed of 0.16 mm/r. Figure 8 illustrates the chip targets within a real processing site environment, in line with the actual production conditions. The second experimental campaign was conducted using a HUAWEI P60 smartphone positioned at an approximate working distance of 1 m, capturing 42 video clips, totaling approximately one hour in duration. The machining setup employed a CK6140S CNC lathe with a 50-mm-diameter workpiece, maintaining the following cutting parameters: rotation speed $V_c = 90$ m/min, cutting speed $f = 0.12$ mm/r, and cutting depth $ap = 1.4$ mm.

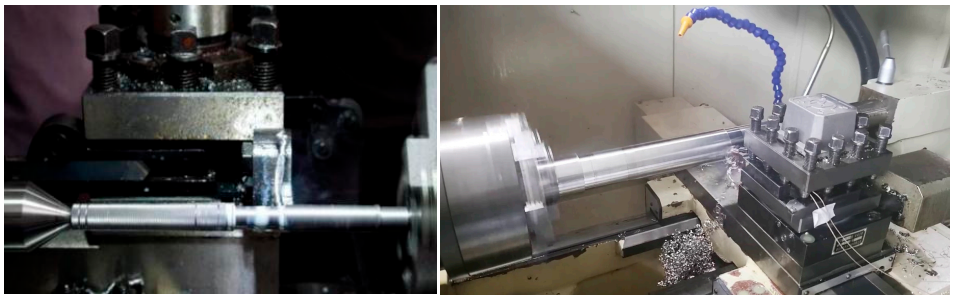


Figure 8. Chip video data.

The first and second experiments yielded 73 and 42 videos, respectively. From these, 105 videos (67 from Experiment 1, 38 from Experiment 2) were processed into 28,234 chip images for training. The remaining 10 videos (six from Experiment 1, four from Experiment 2) served as test data: first, they were decomposed into 5567 images to evaluate the model's accuracy in detecting chips; second, they were tested as complete videos to assess the overall system's accuracy. All images were reshaped to 640 × 640.

3.2. Performance Indicators

In order to evaluate the extraction accuracy and computational cost of the proposed algorithm for chip detection, several commonly used metrics are employed [16,17,19,30,31]: precision, recall, F1-score, mAP@0.5, FLOPs, model size (MS), and frames per second (FPS). To evaluate the accuracy of the chip quantity statistics in videos, the error proportion is used as the indicator.

1. Precision measures the proportion of correctly detected chip images relative to the total number of images identified as chips (both correct and incorrect). It is formally defined as

$$P = \frac{TP}{TP + FP} \tag{12}$$

where TP is the number of samples correctly identified as debris, FP is the number of samples whose background is identified as chips, and FN is the number of chip samples identified as the background.

2. Recall measures the proportion of correctly detected chip images relative to the total number of images that should have been detected (including both correctly detected and undetected chips). It is formally defined as

$$R = \frac{TP}{TP + FN} \tag{13}$$

3. The F1-score is a composite metric for the evaluation of the performance of classification models, defined as the harmonic mean of the precision and recall. It is formally defined as

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \tag{14}$$

4. The average precision (AP) averages the accuracy of the chips in the dataset. The mean average precision (mAP) refers to the average of the AP values for each category. However, the only category in this experiment is pineapple, and the AP is equal to the mAP. The mAP is formally defined as

$$mAP = \frac{\sum_{i=1}^{N} AP_i}{N} \tag{15}$$

where AP $= \int_0^1 \mathrm{P(R)dR}$, and N represents the quantity of the detection category. mAP@0.5 represents the mean average precision when the IoU of the prediction and ground truth boxes is greater than 0.5.

5. The FLOPs value quantifies the computational complexity of the algorithm by measuring the number of floating-point operations required during inference. It is formally defined as

$$FLOPs = \sum_{layer=1...N} \left(K^{layer} \times K^{layer} \times {C_{in}}^{layer} \times {C_{out}}^{layer} \times H^{layer} \times W^{layer} \right) \tag{16}$$

For each layer, H^{layer} and W^{layer} represent the sizes of the output feature maps, respectively. K^{layer} is the convolutional kernel size, and ${C_{in}}^{layer}$ and ${C_{out}}^{layer}$ stand for the numbers of input channels and output channels, respectively.

6. Model Size (MS) refers to the size of model and is used to evaluate the complexity of the model.
7. Frames per second (FPS) is the number of pictures that the model can detect in one second. The calculation formula is

$$FPS = \frac{1}{T_{pre} + T_{in} + T_{post}} \tag{17}$$

where $\mathrm{T_{pre}}$, $\mathrm{T_{in}}$, $\mathrm{T_{post}}$ indicate, respectively, the time (units: second) taken for preprocessing, inference, and post-processing.

8. The error proportion is used to evaluate the accuracy of the chip quantity statistics in videos to assess the overall system's accuracy. It is formally defined as

$$EP = \frac{|S_{true} - S_{det}|}{S_{true}} \times 100\% \tag{18}$$

where $\mathrm{S_{true}}$ represents the true number of falling chips shown in a video, and $\mathrm{S_{det}}$ represents the number of falling chips detected by the algorithm in a video.

3.3. Experimental Results

3.3.1. Visualization of the Results

The hardware environment for this experiment was as follows: CPU Intel i5-13500HX, GPU NVIDIA GTX 4050, memory 16 G. The operating system was Windows 11. The software environment for the experiment consisted of Ultralytics-8.3.7, Python-3.11.0, and PyTorch-2.6.0+cu124. During training, the total number of epochs was set to 100, the batch size was set to 16, the Adam optimizer was chosen, the initial learning rate was set to 0.01, and the weight decay coefficient was set to 0.0005. Moreover, 10% of the data was used for validation. Three sets of anchor boxes were used. The training convergence curves of YOLOv11n, GM-YOLOv11-backbone, GM-YOLOv11-head, and GM-YOLOv11 are shown in Figure 9.

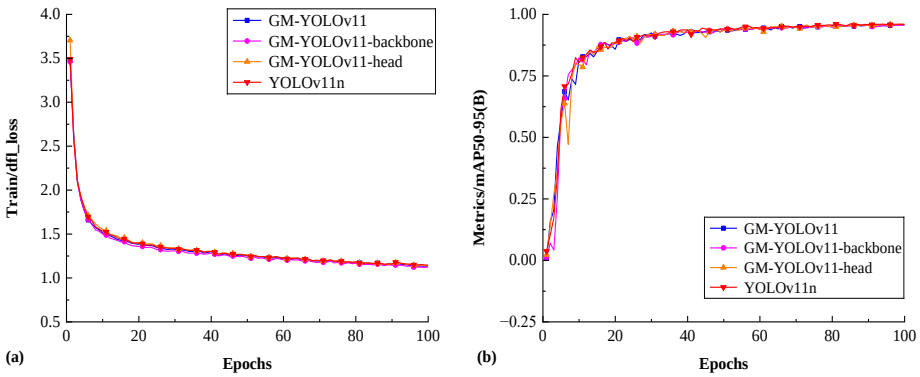


Figure 9. Training loss and mAP@0.5 for YOLOv11n, GM-YOLOv11-backbone, GM-YOLOv11-head, and GM-YOLOv11: (**a**) training loss, (**b**) mAP@0.5.

As can be seen from Figure 9, the convergence curves of the YOLOv11n, GM-YOLOv11-backbone, GM-YOLOv11-head, and GM-YOLOv11 models are very close, indicating that their performance during the learning process is basically consistent after replacing the CNN with the GM. The results of chip detection by GM-YOLOv11-DNMS are shown in Figure 10.

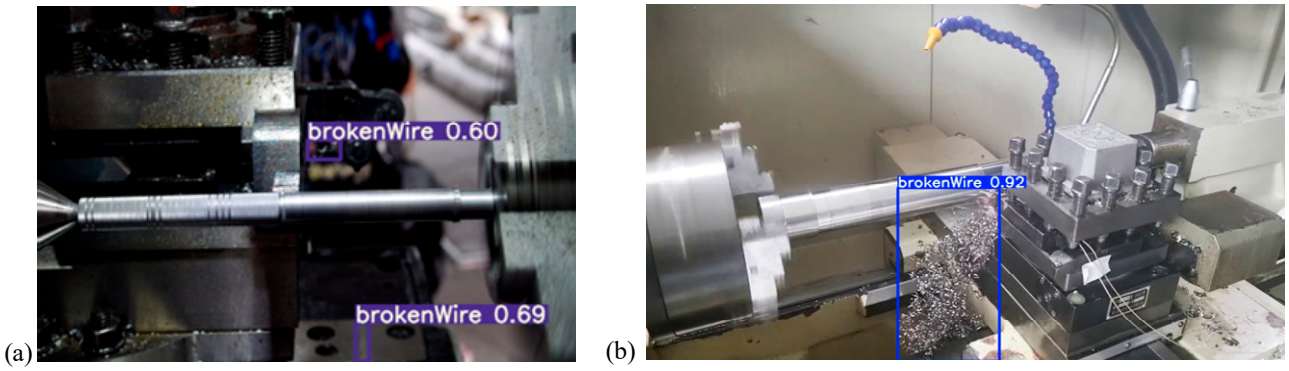


Figure 10. Detection result: (**a**) image captured by Pentax KS2, (**b**) image captured by HUAWEI P60 smartphone.

3.3.2. Ghost Module Improvement Experiment

Following the method provided in Section 2.1.3, modifications were made to YOLOv11n, resulting in GM-YOLOv11-backbone, GM-YOLOv11-head, and GM-YOLOv11. These three models were compared with YOLOv11n, as well as YOLOv10n, YOLOv9t, YOLOv8n, and YOLOv5n, whose model sizes were around or less than 3MB. All models were trained to extract chips and workpieces from the background. The comparisons were

based on the following metrics: precision, recall, F1-score, mAP@0.5, FLOPs, model size (MS), and frames per second (FPS). The results are shown in Table 2.

Table 2. Comparative analysis of chip detection for different models.

Model	Precision (%)	Recall (%)	F1-Score (%)	mAP@0.5 (%)	FLOPs (G)	MS (MB)	FPS
YOLOv5n	92.66	92.44	92.55	91.10	7.73	1.9	116.68
YOLOv8n	90.22	88.72	89.46	89.53	8.75	3.2	149.02
YOLOv9t	89.67	90.12	89.89	92.24	8.23	2	73.78
YOLOv10n	91.14	86.66	88.84	87.05	8.57	2.3	100.2
YOLOv11n	**94.11**	**95.42**	**94.76**	**94.68**	6.48	2.6	154.34
GM-YOLOv11-backbone	93.60	94.08	93.84	93.63	5.81	2.2	167.34
GM-YOLOv11-head	93.36	93.67	93.51	93.66	6.33	2.5	156.23
GM-YOLOv11	93.80	93.96	93.88	93.81	**5.72**	2	**173.41**
Literature [29]	92.27	91.69	91.98	90.12	6.44	**1.7**	146.34
Literature [28]	91.22	90.63	90.92	90.29	7.84	2.8	135.26

It can be observed that the proposed GM-YOLOv11 achieves precision, recall, and mAP@0.5 values of 93.80%, 93.96%, and 93.81%, respectively. These results are very close to the highest values of 94.11%, 95.42%, and 94.68%, all achieved by YOLOv11n. However, GM-YOLOv11 achieves the lowest FLOPs and the highest FPS. Compared to GM-YOLOv11-head, GM-YOLOv11-backbone shows slightly higher precision, recall, and mAP@0.5 values, while also having lower FLOPs, a smaller model size, and higher FPS. Compared to GM-YOLOv11-backbone, GM-YOLOv11 has slightly lower recall but better precision, a smaller model size, lower FLOPs, and higher FPS.

3.3.3. DNMS Improvement Experiment

In DNMS, there are four NMS thresholds to be set: regular parameters α and β in Equation (10), left chip threshold $a1$, and leaving chip threshold $a2$. In this study, we conducted an orthogonal experiment to optimize parameter selection, with some of the better parameter combinations shown in Table 3. Ultimately, we set the left chip threshold at 0.5 and the leaving chip threshold at 0.4, $\alpha = 0.4$, and $\beta = 0.7$. The chip extraction results obtained using DNMS and the model presented in Table 2 are displayed in Table 4. As shown in Table 4, reference [28,29] introduced the ghost module to YOLOv8n-seg and YOLOv5s, respectively. However, to ensure a fair comparison while maintaining a comparable number of model parameters, this work applied their enhancement methods to YOLOv8n and YOLOv5n, with all hyperparameters configured according to Ultralytics' recommendations.

Table 3. Partial results from the orthogonal experiment on the hyperparameters in the DNMS algorithm.

Regular Parameters α in Equation (10)	Regular Parameters β in Equation (10)	Left Chip Threshold $a1$	Leaving Chip Threshold $a2$	Recall (%)
0.3	0.6	0.4	0.3	96.27
0.3	0.7	0.5	0.3	95.89
0.4	0.7	0.5	0.4	**96.38**
0.5	0.4	0.6	0.5	95.75
0.8	0.7	0.7	0.5	95.75
0.9	0.6	0.7	0.6	95.86

Table 4. Comparative analysis of chip detection for different models with DNMS.

Model	Precision (%)	Recall (%)	F1-Score (%)	mAP@0.5 (%)
YOLOv5n+DNMS	93.85	94.11	93.98	92.99
YOLOv8n+DNMS	93.30	90.78	92.02	90.85
YOLOv9t+DNMS	90.83	91.80	91.31	93.34
YOLOv10n+DNMS	92.46	88.33	90.35	88.89
YOLOv11n+DNMS	97.05	96.81	96.93	96.48
GM-YOLOv11-backbone+DNMS	95.13	96.18	95.65	95.08
GM-YOLOv11-neck+DNMS	94.81	94.37	94.59	95.01
GM-YOLOv11+DNMS	97.04	96.38	96.71	95.56
Literature [25]	92.27	91.69	91.98	90.12
Literature [24]	91.22	90.63	90.92	90.29

3.3.4. Results of Video-Level Post-Processing Algorithm

Ten videos that had not been included in training were used to test the results of all algorithms. The details of the six videos are shown in Table 5. The number of chips in the videos was counted using all versions of the YOLO algorithm combined with two post-processing algorithms. The results obtained by GM-YOLOv11+DNMS+VPPA are shown in Table 4. The number of crack chips in these videos was 436, while GM-YOLOv11+DNMS+VPPA counted 393 crack chips, resulting in overall accuracy of 89.9%. The comparison results with other models are shown in Table 6.

Table 5. Details of the ten videos.

Serial Number	Duration (s)	Actual Number of Crack Chips
1	36	13
2	103	47
3	158	61
4	210	82
5	213	79
6	213	80
7	51	25
8	49	18
9	35	24
10	28	7
Total	1096	436

Table 6. Chip quantity counting experiment.

Model	Error Proportion (%)
YOLOv5n+DNMS+VPPA	11.93
YOLOv8n+DNMS+VPPA	15.37
YOLOv9t+DNMS+VPPA	13.07
YOLOv10n+DNMS+VPPA	13.30
YOLOv11n+DNMS+VPPA	9.63
GM-YOLOv11-backbone+DNMS+VPPA	10.55
GM-YOLOv11-neck+DNMS+VPPA	11.47
GM-YOLOv11+DNMS+VPPA	9.86
Literature [29] + VPPA	15.14
Literature [28] + VPPA	15.60

4. Discussion

During machining, experienced craftsmen can assess the quality of the process based on the number and shapes of chips produced. Therefore, it is essential to detect and quantify chips during the machining process. Papers [7–9] utilized deep learning algorithms to investigate the correlation between machining chips and tool wear, incorporating analyses of the chip morphology and color recognition. While some of these algorithms are deployable in online systems, research on the precise video-based quantification of chip volumes remains notably scarce in the existing literature.

The use of an improved backbone and neck architecture enhances the feature extraction capabilities. However, because the chip is relatively small compared to the workpiece and the background, typically ranging from 0.4 mm to 0.7 mm or even smaller, and because industrial environments generally prefer industrial computers or embedded devices with high requirements in terms of software reliability and real-time performance, the use of YOLO for chip detection poses certain challenges. Table 2 describes the performance of various versions of the YOLO model, with parameter sizes of around 3 MB or less, in chip monitoring. Table 5 provides seven metrics, evaluating both the accuracy and real-time performance of the YOLO model and its improved versions. From Table 5, it can be observed that YOLOv11 achieved the highest values in precision, recall, and mAP@0.5, indicating its strong performance in terms of accuracy. YOLOv11 employs an improved backbone and neck architecture, enhancing the feature extraction capabilities, which may be the reason for its higher accuracy in detecting smaller objects.

Additionally, from Table 2, we can observe that, with the introduction of the ghost module into YOLOv11, the model's accuracy slightly decreased. Compared to YOLOv11, GM-YOLOv11 shows lower values in terms of precision, recall, and mAP@0.5 by 0.31%, 1.46%, and 0.87%, respectively; GM-YOLOv11-backbone, compared to YOLOv11, exhibits lower values for precision, recall, and mAP@0.5 by 0.51%, 1.34%, and 1.05%, respectively. From Table 6, it can be seen that this decrease in accuracy does not affect the final statistical results of chip counting in the video. When using the error statistic, GM-YOLOv11 differs by only one count compared to YOLOv11, while GM-YOLOv11-backbone differs by only three counts compared to YOLOv11. However, the introduction of the ghost module significantly reduces the computational complexity, as shown in the Table 2. The FLOPs value for GM-YOLOv11 is 5.72G, while that of YOLOv11 reaches 6.48 G. Many scholars [28,29,32–34] have previously demonstrated in earlier versions such as YOLOv8 and YOLOv5 that introducing the ghost module can result in similar accuracy while significantly reducing the computational loads of YOLO models. This study applied it to YOLOv11 and found that similar conclusions hold even in the detection of smaller chips.

In turning operations, the workpiece is generally much larger in size compared to the chips and has more distinct features, making it relatively easier to identify. The DNMS method proposed in this paper classifies the chips based on the position of the workpiece and then processes the extracted bounding boxes using different NMS thresholds. Comparing Tables 2 and 3, the accuracy of different versions of the YOLO algorithm was improved after adopting DNMS. For the precision metric, the highest improvement was observed in the GM-YOLOv11 model, which saw an increase of 3.24%, while the lowest improvement was in the YOLOv9t model, which increased by 1.16%. For recall, the most significant improvement was in GM-YOLOv11, where it increased by 2.42%, while the smallest improvement was in GM-YOLOv11-neck, where it increased by 0.7%. For mAP@0.5, the YOLOv5 model showed the largest increase at 1.89%, and YOLOv9t showed the smallest increase at 1.1%. The precision, recall, and mAP@0.5 values for the proposed YOLOv11n+DNMS are 97.05%, 96.81%, and 96.48%, respectively. The proposed lightweight model GM-YOLOv11+DNMS achieved precision, recall, and mAP@0.5 values of 97.04%,

96.38%, and 95.56%, respectively. Compared to the methods in the literature [28,29], this approach has advantages. The introduction of DNMS improves the accuracy of different versions of the YOLO algorithm in chip detection. Although the idea of soft NMS has been proposed before [35], it differs from our DNMS algorithm, which is specifically designed considering the unique characteristics of our application. The DNMS algorithm presented in this paper first identifies larger objects in the image—the workpiece—and then, based on the actual machining scenario, where chips may or may not be near the workpiece, it sets different NMS thresholds, thereby enhancing the detection accuracy.

Building on the model that accurately identified chips in images, we also designed the VPPA, which enables the statistical counting of chips in videos. In turning operations, as chips are cut from the workpiece and ejected into the air, their shape continuously changes. This paper introduces a specialized triggering mechanism to count the number of ejected chip particles. From Table 5, it can be seen that GM-YOLOv11+DNMS+VPPA and YOLOv11n+DNMS+VPPA accurately identified 393 and 394 out of 436 ejected chip particles, achieving an accuracy rate of 90.14% and 90.37%, respectively. Compared to other models, these two models achieved the highest accuracy, demonstrating the value of introducing GM and DNMS into YOLOv11. In general, the actual processing time in real-world scenarios is much shorter than in the video tested in this paper. This accuracy rate indicates that the chip count is very close to the true value. Additionally, GM-YOLOv11+DNMS+VPPA applies post-processing based on the GM-YOLOv11 model.

The hardware environment in this study was as follows: an Intel i5-13500HX CPU, an NVIDIA GTX 4050 GPU, and 16 G of memory. This configuration represents a relatively common computer setup. There are many models of industrial control computers and industrial edge devices that are equivalent to this configuration. This study evaluated different models based on the number of model parameters, FLOPs, and FPS, as shown in Table 5. FPS directly reflects the inference speed, and FLOPs indicates the computational load required for a single inference. When processing images, GM-YOLOv11 achieved FLOPs and FPS values of 5.72 G and 173.41, respectively. Although the FLOPs value remained relatively high, suggesting potential gaps in its application to mobile devices, the GM-YOLOv11 model has only 2 MB of parameters and a FLOPs value close to 5 G, with an FPS value significantly exceeding 30. This makes it a promising candidate for deployment on embedded devices or industrial edge devices.

5. Conclusions

During machining, workers pay attention to the quantity and shape of the chips generated in the process to assess the machining quality. Therefore, a machine vision-based method for the counting and monitoring of chips produced during turning operations may help to promote process optimization. However, detecting chips using the current object detection algorithms is challenging due to their generally small sizes during machining. This paper aimed to develop a lightweight model that accurately identified chips and their locations and counted the number of chip particles appearing in videos. First, upon introducing the GM module, the FPS value of YOLOv11 was improved, the number of parameters was reduced, and the FLOPs value was decreased, with only a limited drop in accuracy. Then, by using the workpiece as a reference during machining and identifying it, different NMS thresholds were applied to process the bounding boxes of chips based on their positions relative to the workpiece, leading to the development of DNMS. The results demonstrate that DNMS significantly enhances the accuracy of multiple versions of YOLO in chip detection. Finally, we also designed the video-level post-processing algorithm (VPPA), which implements a rising edge signal trigger mechanism to refine the counting rules for crack chips. Its purpose is to count the number of broken

chip particles appearing in the video. This approach effectively monitors the quantity of crack chips produced during the machining processes of machine tools. It allows for the quantitative analysis of the monitored data, facilitating the assessment of the surface quality of processed workpieces. The model size and frames per second (FPS) serve as critical metrics in evaluating hardware resource consumption in models. According to the Ultralytics community documentation, YOLOv11n demonstrates compatibility with diverse embedded devices, including the RK3576 and RV1106 platforms. The model proposed in this study enhances the YOLOv11n architecture, achieving a more compact structure while maintaining comparable FPS. This optimized model significantly reduces the computational complexity without compromising the detection accuracy, exhibiting strong potential for the real-time monitoring of turning processes in embedded systems.

However, this research presents the following limitations: (1) the absence of dual label assignment and consistent matching metric strategies necessitates a reliance on non-maximum suppression (NMS) during inference, limiting the detection speed (FPS); (2) machining tests were conducted without cutting fluid, leaving the algorithm's performance under cutting fluid interference unverified.

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

Funding: This research was funded by the Center for Balance Architecture, Zhejiang University, grant number K-20203312C and Construction Scientific Research Project in Zhejiang Province, grant number 2018K068.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Conflicts of Interest: Author Ruofei Liu was employed by the company The Architectural Design & Research Institute of Zhejiang University Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Appendix A

Table A1. Architecture of GM-YOLOv11-Backbone.

Section	ID	From	Repeats	Module	Args	Parameter Explanation
Backbone	0	−1	1	GhostConv	[64, 3,3,2,2]	Input channels auto-inferred, output 64 channels, 3 × 3 kernel for both primary convolution and linear operation, s = 2, stride 2
	1	−1	1	GhostConv	[128, 3,3,2,2]	Output 128 channels, 3 × 3 kernel for both primary convolution and linear operation, s = 2, stride 2
	2	−1	2	C3k2	[256, False, 0.25]	Output 256 channels, no shortcut connection, bottleneck width reduction ratio 0.25
	3	−1	1	GhostConv	[256, 3,3,2,2]	Output 256 channels, 3 × 3 kernel, stride 2
	4	−1	2	C3k2	[512, False, 0.25]	Output 512 channels, no shortcut connection, bottleneck width reduction ratio 0.25
	5	−1	1	GhostConv	[512, 3,3,2,2]	Output 512 channels, 3 × 3 kernel for both primary convolution and linear operation, s = 2, stride 2

Table A1. *Cont.*

Section	ID	From	Repeats	Module	Args	Parameter Explanation
Backbone	6	−1	2	C3k2	[512, True]	Output 512 channels, with shortcut connection
	7	−1	1	GhostConv	[1024,3,3,2,2]	Output 1024 channels, 3 × 3 kernel for both primary convolution and linear operation, s = 2, stride 2
	8	−1	2	C3k2	[1024, True]	Output 1024 channels, with shortcut connection
	9	−1	1	SPPF	[1024, 5]	Output 1024 channels, max pooling kernel size 5 × 5
	10	−1	2	C2PSA	[1024]	Output 1024 channels
Head	0	−1	1	nn.Upsample	[None, 2, “nearest”]	Upsample with scale factor 2, nearest-neighbor interpolation
	1	[−1, 6]	1	Concat	[1]	Concatenate current layer with backbone layer 6 along channel dim
	2	−1	2	C3k2	[512, False]	Output 512 channels, no shortcut connection
	3	−1	1	nn.Upsample	[None, 2, “nearest”]	Second 2× upsampling
	4	[−1, 4]	1	Concat	[1]	Concatenate with backbone layer 4
	5	−1	2	C3k2	[256, False]	Output 256 channels, no shortcut connection
	6	−1	1	Conv	[256, 3, 2]	Output 256 channels, 3 × 3 kernel, stride 2
	7	[−1, 13]	1	Concat	[1]	Concatenate with head layer 13
	8	−1	2	C3k2	[512, False]	Output 512 channels, no shortcut connection
	9	−1	1	Conv	[512, 3, 2]	Output 512 channels, 3 × 3 kernel, stride 2
	10	[−1, 10]	1	Concat	[1]	Concatenate with backbone layer 10
	11	−1	2	C3k2	[1024, True]	Output 1024 channels, with shortcut connection
	12	[16, 19, 22]	1	Detect	[2]	Number of classes is 2

Table A2. Architecture of GM-YOLOv11-Head.

Section	ID	From	Repeats	Module	Args	Parameter Explanation
Backbone	0	−1	1	Conv	[64, 3, 2]	Input channels auto-inferred, output 64 channels, 3 × 3 kernel, stride 2
	1	−1	1	Conv	[128, 3, 2]	Output 128 channels, 3 × 3 kernel, stride 2
	2	−1	2	C3k2	[256, False, 0.25]	Output 256 channels, no shortcut connection, bottleneck width reduction ratio 0.25
	3	−1	1	Conv	[256, 3, 2]	Output 256 channels, 3 × 3 kernel, stride 2
	4	−1	2	C3k2	[512, False, 0.25]	Output 512 channels, no shortcut connection, bottleneck width reduction ratio 0.25
	5	−1	1	Conv	[512, 3, 2]	Output 512 channels, 3 × 3 kernel, stride 2
	6	−1	2	C3k2	[512, True]	Output 512 channels, with shortcut connection
	7	−1	1	Conv	[1024, 3, 2]	Output 1024 channels, 3 × 3 kernel, stride 2
	8	−1	2	C3k2	[1024, True]	Output 1024 channels, with shortcut connection
	9	−1	1	SPPF	[1024, 5]	Output 1024 channels, max pooling kernel size 5 × 5
	10	−1	2	C2PSA	[1024]	Output 1024 channels
Head	0	−1	1	nn.Upsample	[None, 2, “nearest”]	Upsample with scale factor 2, nearest-neighbor interpolation
	1	[−1, 6]	1	Concat	[1]	Concatenate current layer with backbone layer 6 along channel dim
	2	−1	2	C3k2	[512, False]	Output 512 channels, no shortcut connection
	3	−1	1	nn.Upsample	[None, 2, “nearest”]	Second 2× upsampling
	4	[−1, 4]	1	Concat	[1]	Concatenate with backbone layer 4

Table A2. *Cont.*

Section	ID	From	Repeats	Module	Args	Parameter Explanation
Head	5	−1	2	C3k2	[256, False]	Output 256 channels, no shortcut connection
	6	−1	1	GhostConv	[256, 3,3,2,2]	Output 256 channels, 3 × 3 kernel for both primary convolution and linear operation, s = 2, stride 2
	7	[−1, 13]	1	Concat	[1]	Concatenate with head layer 13
	8	−1	2	C3k2	[512, False]	Output 512 channels, no shortcut connection
	9	−1	1	GhostConv	[512,3,3,2,2]	Output 512 channels, 3 × 3 kernel for both primary convolution and linear operation, s = 2, stride 2
	10	[−1, 10]	1	Concat	[1]	Concatenate with backbone layer 10
	11	−1	2	C3k2	[1024, True]	Output 1024 channels, with shortcut connection
	12	[16, 19, 22]	1	Detect	[2]	Number of classes is 2

Table A3. GM-YOLOv11.

Section	ID	From	Repeats	Module	Args	Parameter Explanation
Backbone	0	−1	1	GhostConv	[64, 3,3,2,2]	Input channels auto-inferred, output 64 channels, 3 × 3 kernel for both primary convolution and linear operation, s = 2, stride 2
	1	−1	1	GhostConv	[128, 3,3,2,2]	Output 128 channels, 3 × 3 kernel for both primary convolution and linear operation, s = 2, stride 2
	2	−1	2	C3k2	[256, False, 0.25]	Output 256 channels, no shortcut connection, bottleneck width reduction ratio 0.25
	3	−1	1	GhostConv	[256, 3,3,2,2]	Output 256 channels, 3 × 3 kernel, stride 2
	4	−1	2	C3k2	[512, False, 0.25]	Output 512 channels, no shortcut connection, bottleneck width reduction ratio 0.25
	5	−1	1	GhostConv	[512, 3,3,2,2]	Output 512 channels, 3 × 3 kernel for both primary convolution and linear operation, s = 2, stride 2
	6	−1	2	C3k2	[512, True]	Output 512 channels, with shortcut connection
	7	−1	1	GhostConv	[1024,3,3,2,2]	Output 1024 channels, 3 × 3 kernel for both primary convolution and linear operation, s = 2, stride 2
	8	−1	2	C3k2	[1024, True]	Output 1024 channels, with shortcut connection
	9	−1	1	SPPF	[1024, 5]	Output 1024 channels, max pooling kernel size 5 × 5
	10	−1	2	C2PSA	[1024]	Output 1024 channels
Head	0	−1	1	nn.Upsample	[None, 2, “nearest”]	Upsample with scale factor 2, nearest-neighbor interpolation
	1	[−1, 6]	1	Concat	[1]	Concatenate current layer with backbone layer 6 along channel dim
	2	−1	2	C3k2	[512, False]	Output 512 channels, no shortcut connection
	3	−1	1	nn.Upsample	[None, 2, “nearest”]	Second 2× upsampling
	4	[−1, 4]	1	Concat	[1]	Concatenate with backbone layer 4
	5	−1	2	C3k2	[256, False]	Output 256 channels, no shortcut connection
	6	−1	1	GhostConv	[256, 3,3,2,2]	Output 256 channels, 3 × 3 kernel for both primary convolution and linear operation, s = 2, stride 2
	7	[−1, 13]	1	Concat	[1]	Concatenate with head layer 13
	8	−1	2	C3k2	[512, False]	Output 512 channels, no shortcut connection

Table A3. *Cont.*

Section	ID	From	Repeats	Module	Args	Parameter Explanation
	9	−1	1	GhostConv	[512,3,3,2,2]	Output 512 channels, 3 × 3 kernel for both primary convolution and linear operation, s = 2, stride 2
Head	10	[−1, 10]	1	Concat	[1]	Concatenate with backbone layer 10
	11	−1	2	C3k2	[1024, True]	Output 1024 channels, with shortcut connection
	12	[16, 19, 22]	1	Detect	[2]	Number of classes is 2

References

1. García Plaza, E.; Núñez López, P.J.; Beamud González, E.M. Multi-Sensor Data Fusion for Real-Time Surface Quality Control in Automated Machining Systems. *Sensors* **2018**, *18*, 4381. [CrossRef] [PubMed]
2. Fowler, N.O.; McCall, D.; Chou, T.-C.; Holmes, J.C.; Hanenson, I.B. Electrocardiographic Changes and Cardiac Arrhythmias in Patients Receiving Psychotropic Drugs. *Am. J. Cardiol.* **1976**, *37*, 223–230. [CrossRef] [PubMed]
3. Chen, T.; Guo, J.; Wang, D.; Li, S.; Liu, X. Experimental Study on High-Speed Hard Cutting by PCBN Tools with Variable Chamfered Edge. *Int. J. Adv. Manuf. Technol.* **2018**, *97*, 4209–4216. [CrossRef]
4. Cheng, Y.; Guan, R.; Zhou, S.; Zhou, X.; Xue, J.; Zhai, W. Research on Tool Wear and Breakage State Recognition of Heavy Milling 508III Steel Based on ResNet-CBAM. *Measurement* **2025**, *242*, 116105. [CrossRef]
5. Vorontsov, A.L.; Sultan-Zade, N.M.; Albagachiev, A.Y. Development of a New Theory of Cutting 8. Chip-Breaker Design. *Russ. Eng. Res.* **2008**, *28*, 786–792. [CrossRef]
6. Zaidi, S.S.A.; Ansari, M.S.; Aslam, A.; Kanwal, N.; Asghar, M.; Lee, B. A Survey of Modern Deep Learning Based Object Detection Models. *Digit. Signal Process.* **2022**, *126*, 103514. [CrossRef]
7. Pagani, L.; Parenti, P.; Cataldo, S.; Scott, P.J.; Annoni, M. Indirect cutting tool wear classification using deep learning and chip colour analysis. *Int. J. Adv. Manuf. Technol.* **2020**, *111*, 1099–1114. [CrossRef]
8. Rehman, A.U.; Nishat, T.S.R.; Ahmed, M.U.; Begum, S.; Ranjan, A. Chip Analysis for Tool Wear Monitoring in Machining: A Deep Learning Approach. *IEEE Access* **2024**, *12*, 112672–112689. [CrossRef]
9. Chen, S.H.; Lin, Y.Y. Using cutting temperature and chip characteristics with neural network BP and LSTM method to predicting tool life. *Int. J. Adv. Manuf. Technol.* **2023**, *127*, 881–897. [CrossRef]
10. Shen, D.; Chen, X.; Nguyen, M.; Yan, W.Q. Flame Detection Using Deep Learning. In Proceedings of the 2018 4th International Conference on Control, Automation and Robotics (ICCAR), Auckland, New Zealand, 20–23 April 2018; pp. 416–420.
11. Cai, C.; Wang, B.; Liang, X. A New Family Monitoring Alarm System Based on Improved YOLO Network. In Proceedings of the 2018 Chinese Control and Decision Conference (CCDC), Shenyang, China, 9–11 June 2018; pp. 4269–4274.
12. Badgujar, C.M.; Poulose, A.; Gan, H. Agricultural Object Detection with You Only Look Once (YOLO) Algorithm: A Bibliometric and Systematic Literature Review. *Comput. Electron. Agric.* **2024**, *223*, 109090. [CrossRef]
13. Liu, J.; Zhu, X.; Zhou, X.; Qian, S.; Yu, J. Defect Detection for Metal Base of TO-Can Packaged Laser Diode Based on Improved YOLO Algorithm. *Electronics* **2022**, *11*, 1561. [CrossRef]
14. Banda, T.; Jauw, V.L.; Farid, A.A.; Wen, N.H.; Xuan, K.C.W.; Lim, C.S. In-process detection of failure modes using YOLOv3-based on-machine vision system in face milling Inconel 718. *Int. J. Adv. Manuf. Technol.* **2023**, *128*, 3885–3899. [CrossRef]
15. Redmon, J.; Farhadi, A. Yolov3: An Incremental Improvement. *arXiv* **2018**, arXiv:1804.02767.
16. Bochkovskiy, A.; Wang, C.-Y.; Liao, H.-Y.M. Yolov4: Optimal Speed and Accuracy of Object Detection. *arXiv* **2020**, arXiv:2004.10934.
17. Wang, C.-Y.; Bochkovskiy, A.; Liao, H.-Y.M. YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Online, 27 June 2023; pp. 7464–7475.
18. Swathi, Y.; Challa, M. YOLOv8: Advancements and Innovations in Object Detection. In *Smart Trends in Computing and Communications*; Springer Nature: Singapore, 2024; pp. 1–13.
19. Wang, C.-Y.; Yeh, I.-H.; Mark Liao, H.-Y. YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information. In *Computer Vision—ECCV 2024*; Springer Nature: Cham, Switzerland, 2024; pp. 1–21.
20. Kshirsagar, V.; Bhalerao, R.H.; Chaturvedi, M. Modified YOLO Module for Efficient Object Tracking in a Video. *IEEE Lat. Am. Trans.* **2023**, *21*, 389–398. [CrossRef]
21. Doherty, J.; Gardiner, B.; Kerr, E.; Siddique, N. BiFPN-YOLO: One-Stage Object Detection Integrating Bi-Directional Feature Pyramid Networks. *Pattern Recognit.* **2025**, *160*, 111209. [CrossRef]

22. Yan, J.; Zeng, Y.; Lin, J.; Pei, Z.; Fan, J.; Fang, C.; Cai, Y. Enhanced Object Detection in Pediatric Bronchoscopy Images Using YOLO-Based Algorithms with CBAM Attention Mechanism. *Heliyon* **2024**, *10*, e32678. [CrossRef] [PubMed]
23. Wang, J.; Wang, W.; Zhang, Z.; Lin, X.; Zhao, J.; Chen, M.; Luo, L. YOLO-DD: Improved YOLOv5 for Defect Detection. *Comput. Mater. Contin.* **2024**, *78*, 759–780. [CrossRef]
24. Wang, M.; Fu, B.; Fan, J.; Wang, Y.; Zhang, L.; Xia, C. Sweet Potato Leaf Detection in a Natural Scene Based on Faster R-CNN with a Visual Attention Mechanism and DIoU-NMS. *Ecol. Inf.* **2023**, *73*, 101931. [CrossRef]
25. Xue, C.; Xia, Y.; Wu, M.; Chen, Z.; Cheng, F.; Yun, L. EL-YOLO: An Efficient and Lightweight Low-Altitude Aerial Objects Detector for Onboard Applications. *Expert Syst. Appl.* **2024**, *256*, 124848. [CrossRef]
26. Wang, Y.; Wang, B.; Fan, Y. PPGS-YOLO: A Lightweight Algorithms for Offshore Dense Obstruction Infrared Ship Detection. *Infrared Phys. Technol.* **2025**, *145*, 105736. [CrossRef]
27. Han, K.; Wang, Y.; Tian, Q.; Guo, J.; Xu, C.; Xu, C. GhostNet: More Features from Cheap Operations. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Online, 24 June 2020; pp. 1577–1586.
28. Wang, H.; Wang, G.; Li, Y.; Zhang, K. YOLO-HV: A Fast YOLOv8-Based Method for Measuring Hemorrhage Volumes. *Biomed. Signal Process. Control* **2025**, *100*, 107131. [CrossRef]
29. Dong, X.; Yan, S.; Duan, C. A Lightweight Vehicles Detection Network Model Based on YOLOv5. *Eng. Appl. Artif. Intell.* **2022**, *113*, 104914. [CrossRef]
30. Rashmi; Chaudhry, R. SD-YOLO-AWDNet: A Hybrid Approach for Smart Object Detection in Challenging Weather for Self-Driving Cars. *Expert Syst. Appl.* **2024**, *256*, 124942. [CrossRef]
31. Cui, M.; Lou, Y.; Ge, Y.; Wang, K. LES-YOLO: A Lightweight Pinecone Detection Algorithm Based on Improved YOLOv4-Tiny Network. *Comput. Electron. Agric.* **2023**, *205*, 107613. [CrossRef]
32. Li, J.; Su, Z.; Geng, J.; Yin, Y. Real-Time Detection of Steel Strip Surface Defects Based on Improved YOLO Detection Network. *IFAC-Pap.* **2018**, *51*, 76–81. [CrossRef]
33. Zhang, Y.; Chen, W.; Li, S.; Liu, H.; Hu, Q. YOLO-Ships: Lightweight Ship Object Detection Based on Feature Enhancement. *J. Vis. Commun. Image Represent.* **2024**, *101*, 104170. [CrossRef]
34. Huangfu, Z.; Li, S.; Yan, L. Ghost-YOLO v8: An Attention-Guided Enhanced Small Target Detection Algorithm for Floating Litter on Water Surfaces. *Comput. Mater. Contin.* **2024**, *80*, 3713–3731. [CrossRef]
35. Chen, J.; Chen, H.; Xu, F.; Lin, M.; Zhang, D.; Zhang, L. Real-Time Detection of Mature Table Grapes Using ESP-YOLO Network on Embedded Platforms. *Biosyst. Eng.* **2024**, *246*, 122–134. [CrossRef]

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.