electronics

MDPI

Article

Static Gesture Recognition Algorithm Based on Improved YOLOv5s

Shengwang Wu, Zhongmin Li *, Shiji Li, Qiang Liu and Weiyu Wu

School of Information Engineering, Nanchang Hangkong University, Nanchang 330063, China
* Correspondence: zhongmli@nchu.edu.cn

Abstract: With the government's increasing support for the virtual reality (VR)/augmented reality (AR) industry, it has developed rapidly in recent years. Gesture recognition, as an important human-computer interaction method in VR/AR technology, is widely used in the field of virtual reality. The current static gesture recognition technology has several shortcomings, such as low recognition accuracy and low recognition speed. A static gesture recognition algorithm based on improved YOLOv5s is proposed to address these issues. The content-aware re-assembly of features (CARAFE) is used to replace the nearest neighbor up-sampling method in YOLOv5s to make full use of the semantic information in the feature map and improve the recognition accuracy of the model for gesture regions. The adaptive spatial feature fusion (ASFF) method is introduced to filter out useless information and retain useful information for efficient feature fusion. The bottleneck transformer method is initially introduced into the gesture recognition task, reducing the number of model parameters and increasing the accuracy while accelerating the inference speed. The improved algorithm achieved an mAP(mean average precision) of 96.8%, a 3.1% improvement in average accuracy compared with the original YOLOv5s algorithm; the confidence level of the actual detection results was higher than the original algorithm.

Keywords: YOLOv5s; gesture recognition; ASFF; CARAFE; bottleneck transformer

Citation: Wu, S.; Li, Z.; Li, S.; Liu, Q.; Wu, W. Static Gesture Recognition Algorithm Based on Improved YOLOv5s. *Electronics* **2023**, *12*, 596. https://doi.org/10.3390/electronics12030596

Academic Editor: Seong G. Kong

Received: 16 December 2022
Revised: 17 January 2023
Accepted: 21 January 2023
Published: 25 January 2023

1. Introduction

In recent years, the field of artificial intelligence represented by deep learning has developed rapidly; this efficient machine learning method has also injected strong vitality into the field of computer vision and promoted the application of target detection in gesture recognition. As a direct and immediate interaction method, gesture recognition can be applied in VR/AR, smart homes, smart cars, physical games, and other emerging technology fields. Gesture recognition tasks require both real-time and accuracy, so improving the detection speed and accuracy of gesture recognition is especially important. There are three approaches to gesture recognition. One is collecting and identifying hand information from external hardware devices [1]. This approach has a high accuracy rate, but it relies heavily on external hardware devices and has a high design cost. The second one is based on traditional computer vision; this approach is mainly divided into three steps [2]: hand gesture segmentation, analysis, and detection. This method has high accuracy but low robustness. The last one is the deep learning-based method.

In recent years, deep learning has achieved good results in various fields of computer vision, such as image fusion, image classification, and target detection. Gesture recognition methods based on deep learning are mainly divided into two categories: one is a one-stage target detection algorithm based on SSD [3], YOLOv3 [4], YOLOv4 [5], and the other is Fast R-CNN [6], Faster R-CNN [7], Mask R-CNN [8]-based two-stage object detection algorithm. The YOLOv5 algorithm proposed by Glenn Jocher inherits the high-speed feature of YOLO series algorithms. As a lightweight model, it also has high detection accuracy, which is ideal for deployment on some edge computing devices.

After the above analysis, the YOLOv5 algorithm, which is easy to deploy in mobile terminals and has high accuracy as well as detection speed, is selected for gesture recognition. Aiming at the problem that YOLOv5s is directly applied to gesture recognition, its accuracy is not high enough to improve the accuracy of its recognition without excessively increasing the number of its model parameters. The main contributions of this paper are as follows.

1. We collect and collate 2850 images of different gestures taken from various angles and manually label them.

2. CARAFF is used to fuse the weights of the feature layers at each scale. As a lightweight and efficient up-sampling method, it expands the receptive field and enhances the perception of gestural regions, improving the model recognition accuracy.

3. ASFF is applied in the detect layer to solve the scale inconsistency problem of CSP + PAN. This strategy optimizes the feature fusion process of the network and improves its recognition accuracy.

4. The bottleneck transformer is introduced into gesture recognition for the first time, by using which backbone network can reduce model parameters and increase certain accuracy while speeding up inference. The experimental results show that the introduction of this structure improves the model recognition speed and accuracy.

2. Related Work

At present, gesture recognition tasks can be divided into three categories. One uses an external hardware device, another uses the traditional computer vision method, and the third uses the deep learning method to detect and recognize the gesture.

2.1. External Hardware Device-Based Recognition

Lin [9] used a data glove equipped with a flexible stretch sensor and an inertial sensor to collect gesture data and, after pre-processing the data, transferred it to a recognition model for feature extraction and classification recognition, thereby completing the detection of gestures. Huang et al. [10] used the gesture acquisition module to collect gesture information with the support of the Leap Motion SDK, used the two-finger angular domain feature extraction method for feature extraction based on the weighted cardinality distance fuzzy KNN classification algorithm for the detection of various gestures. This method has high accuracy but relies heavily on external hardware devices and has a high design cost.

2.2. Traditional Computer Vision-Based Recognition

Gesture detection based on traditional computer vision is mainly divided into three steps: gesture segmentation, analysis, and detection. Gesture segmentation usually uses skin color information to model the gesture in a color space, such as YUV [11], HSV [12], or YCbCr [13]; gesture analysis mainly realizes the detection of gesture by analyzing the hand contour or hand motion trajectory. Bai et al. [14] used an image edge algorithm to extract the gesture edges, then fused the extracted edge information with the skin color model to achieve the analysis and detection of gestures. Wang et al. [15] combined RGB and depth image information to fuse and analyze the captured hand trajectory features to achieve the detection of gesture actions. These methods have good accuracy but not high robustness.

2.3. Deep Learning-Based Recognition

Xie et al. [16] proposed an improved SSD gesture detection method that fuses shallow visual features with high-level semantics. Replacing the convolutional layer of the original SSD with the newly fused feature layer and improving the loss function, this method improves the accuracy and recognition speed of the SSD to some extent; however, its recognition accuracy is still not high due to the limitations of the SSD itself. Ling et al. [17] designed a gesture detection system based on the YOLOv3 algorithm, which has a faster recognition speed but only a 76.76% mAP value in the dataset. Yu et al. [18] designed a gesture detection method based on the Fater R-CNN algorithm for complex scenes

using a five-layer neural network architecture; the average accuracy of the trained model recognition reached 94.3%. Wu et al. [19] proposed a perturbed intersection rate algorithm based on the improved Faster R-CNN algorithm to avoid the underfitting problem of the model; moreover, they designed a new gesture detection method, which achieved 84.33% recognition accuracy and 68ms inference time, with low detection accuracy and slow detection speed. The YOLOv8 algorithm proposed by Ultralytics replaced all the C3 structures in YOLOV5 with the C2f structure, which has richer gradient flow, adjusted the different number of channels for different scale models, added more hopping connections and extra Split operations, introduced Distribution Focal Loss to calculate the loss. As the state-of-the-art (SOTA) algorithm, the mAP value of the YOLOv8 algorithm tested on the coco dataset is much improved compared with other YOLO algorithms, but the number of model parameters is significantly increased, which causes it to become a bit slower than most of YOLOV5 models for inference.

2.4. Yolov5 Algorithm

In June 2020, Glenn Jocher of Ultralytics released the YOLOv5 target detection algorithm with both high accuracy and inference speed. The YOLOv5 algorithm can be divided into five versions, YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x, whose network depth and feature map width increase successively. According to the demand for fast speed, high accuracy, and small model volume, YOLOv5s is selected as the baseline model for experimental improvement in this paper. The YOLOv5s algorithm can be divided into four parts in the overall network structure: input, backbone, neck, and output.

2.4.1. Input

The input side of YOLOv5s mainly consists of three parts: data enhancement, automatic image deflation and scaling, and adaptive anchor calculation. In terms of data enhancement, YOLOv5s uses a mixture of data enhancement methods, mainly Mosaic data enhancement and other methods such as HSV gamut transformation, panning, scaling, horizontal flipping, etc. Mosaic data enhancement is done by randomly selecting four images from the dataset; these images are then randomly cropped, scaled, arranged, and finally stitched together in a single image, enriching the dataset and reducing the number of GPUs required for training. Automatic image deflation refers to conducting rectangle aspect ratio transformations in images of different sizes to deflate them to a uniform size and adaptively adjusting them so that the black edges are as small as possible to reduce computation and accelerate model training. The adaptive anchor calculation is done by re-clustering the targets in the training set to generate the anchor during training, outputting predicted data based on the initial anchor set by the YOLOv5s algorithm based on different datasets, and then comparing and calculating the difference between the real data and the predicted data to update the network parameters continuously.

2.4.2. Backbone

The backbone feature network of YOLOv5s consists of CSP-Darknet53. In the backbone network of YOLOv5s, it is mainly a stack of C3 modules and CBS modules. The C3 module mainly refers to the removal of one of the four convolutional layers of BottleneckCSP, which is the main structure for learning features. The CBS module refers to the sequential connection of ordinary convolutional layers, BatchNorm2d layers, and SiLU layers, which serves to perform convolution, batch normalization, and activation function operations on the input feature map. The specific structure is shown in Figure 1.

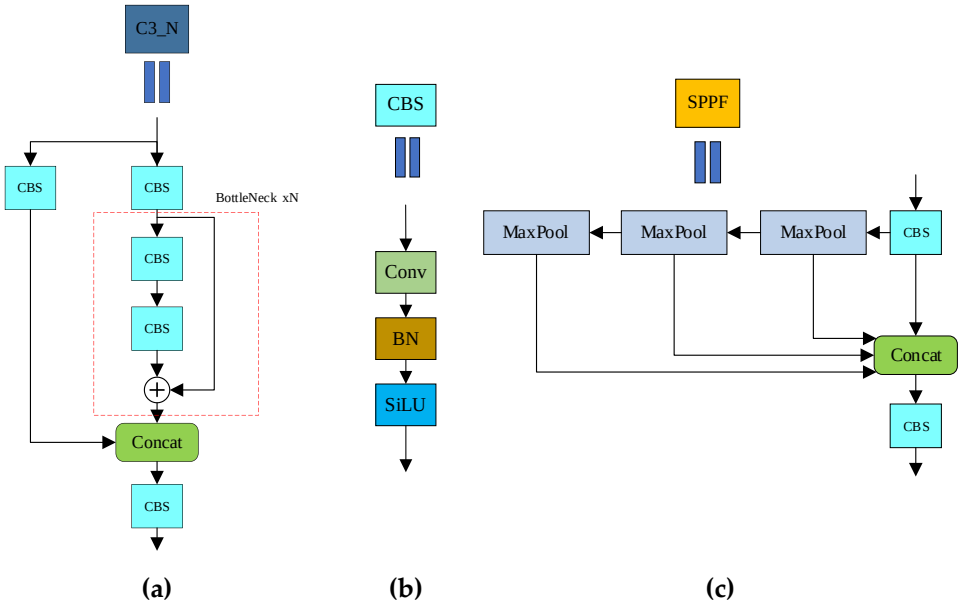

Figure 1. Structure of C3, CBS, and SPPF modules. (**a**) C3; (**b**) CBS; (**c**) SPPF.

2.4.3. Neck

The neck of YOLOv5s consists of the SPPF module and Cross Stage Partial (CSP) + Path Aggregation Network (PAN) module. The core idea of the Spatial Pyramid Pooling–Fast (SPPF) module is to use three different sizes of pooling layers to fuse the features in a comprehensive way, which extends the perceptual field of the feature map and improves the expressiveness of the feature layers. Figure 1 shows the structure of the SPPF module. The CSP + PAN module uses the top-down CSP structure to fuse detailed information such as texture, the edge contour of the bottom features, and the high semantic information of the top features. At the same time, the bottom-up PAN structure is used to supplement the CSP, and the bottom features are down-sampled to transmit the strong localization information of the lower layer. The combination of the two enhances the ability of the model to fuse features. Figure 2 shows the structure of CSP + PAN.

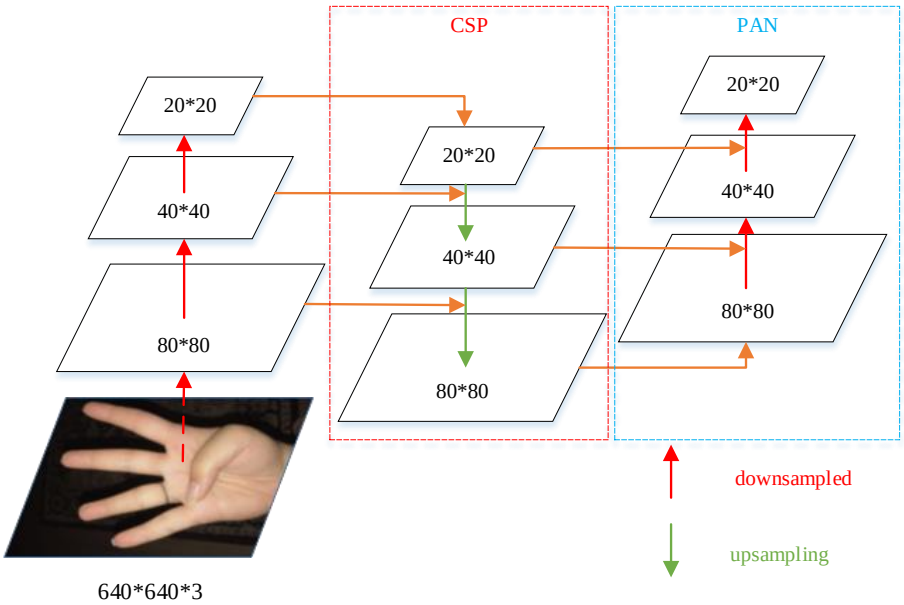

Figure 2. Structure of CSP and PAN.

2.4.4. Output

The loss calculation at the output of YOLOv5s consists of three main components: classes, objectness, and location loss. Both classes and objectness loss use binary cross-entropy loss; the difference is that the classes loss calculates the loss of positive samples, whereas objectness loss is calculated for all samples. The location loss uses Complete Intersection over Union (CIoU) loss, which only calculates the loss of positive samples.

3. Materials and Methods

3.1. Improvement of the Up-Sampling Method

Gesture images are not quite the same as those used for object detection; a gesture image in datasets generally includes a single medium and large targets, with relatively little variability. Up-sampling using the nearest neighbor or bilinear approach employed by traditional target detection methods, the receptive field is too small, and the ability to extract detailed information such as contours of gestures is insufficient. Changing the up-sampling method to content-aware re-assembly of features (CARAFE) expands the receptive field of feature extraction while linking the up-sampling kernel with the semantic information of the feature map to enhance the perception of important contents, and by virtue of its fewer amount of operations, it is also able to keep the model lightweight.

The CARAFE method is divided into two modules: sampling kernel prediction and content-aware feature reorganization. The sampling kernel prediction module is mainly used to generate the reorganization kernel, assuming that the input feature map size is $H \times W \times C$ and the sampling multiplicity is σ(=2). Firstly, a 1*1 convolutional layer is used to compress the input feature channels from C to C_m to reduce the number of parameters and computational cost. Then, content encoding is performed to generate recombination kernels. This is done by using a convolutional layer of size $k_{encoder} \times k_{encoder}$ to predict the sampled kernels of the compressed feature map to obtain a sampled kernel with the number of output channels $\sigma^2 \times k_{up}^2$. We set k_{up} = 5 and $k_{encoder}$ = 3 as a tradeoff between performance and efficiency. Expanding it in the spatial dimension, a recombination kernel of size $\sigma H \times \sigma W \times k_{up} \times k_{up}$ can be obtained. Finally, each of the generated recombination kernels of size $k_{up} \times k_{up}$ are normalized in the spatial dimension using the Softmax function so that the sum of their kernel values is restricted to one. The content-aware feature reassembly module maps each location in the output feature map back to the input feature map, takes out a region of size k_{up} with its center, and makes a dot product with the predicted up-sampling kernel of that point to obtain the output value. For a target location l' and the corresponding square region $N(\chi_l, k_{up})$ centered at $l = (i, j)$, the calculation method is as follows.

$$\chi'_{l'} = \sum_{n=-r}^{r} \sum_{m=-r}^{r} w_{l'(n,m)} \cdot \chi_{(i+n,j+m)} \quad (1)$$

where, χ' denotes the new feature map,$w_{l'}$ denotes the reassembly kernel, $r = \lfloor k_{up}/2 \rfloor$. Figure 3 shows the structure of CARAFE.

3.2. Introducing the Adaptive Spatial Feature Fusion Method

The neck of YOLOv5s uses the CSP + PAN structure to fuse features of different scales; this fusion approach cannot achieve consistent processing at multiple scales of a one-stage target detection task. For example, high-level features are only associated with larger targets, while bottom-level features are associated with smaller ones. This variability in different feature layers will have an impact on the gradient calculation during training, thus reducing the accuracy of the model recognition.

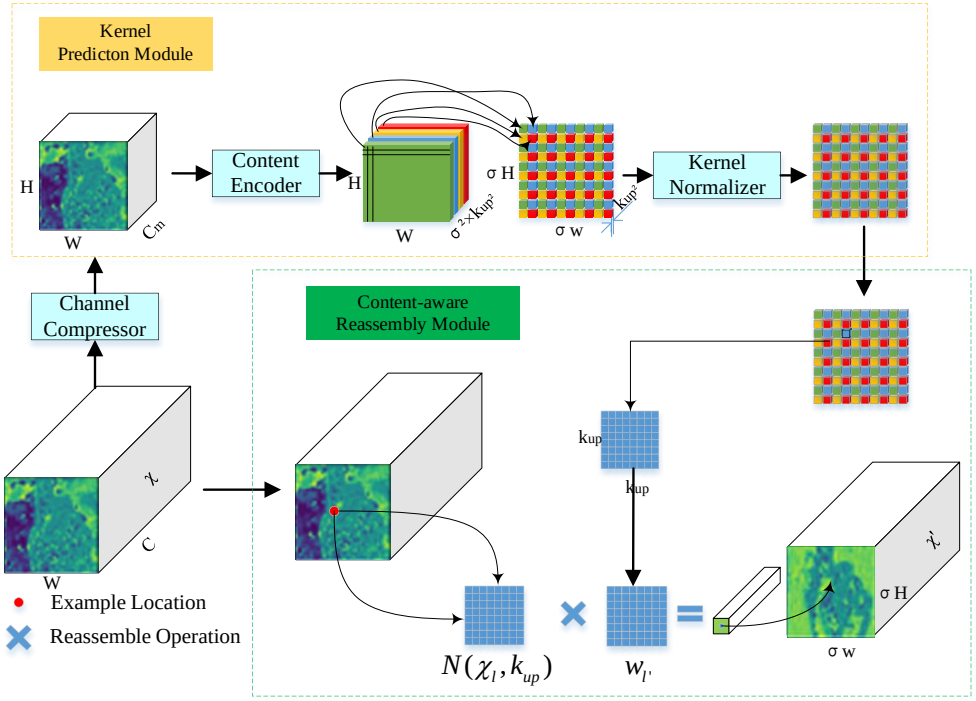

Figure 3. Structure of CARAFE.

This paper introduces an adaptive spatial feature fusion method to solve the problem of inconsistency at different scales when YOLOv5s performs feature fusion. Figure 4 shows the structure of adaptive feature fusion. Since the output of the three-level feature by the neck has different resolutions and channel numbers, such channel differences cannot be used directly for feature fusion. Therefore, it is necessary to modify the sampling scheme of each feature layer accordingly to make the resolution and number of channels the same for each scale for the next stage of feature fusion. In order to compress the feature layers uniformly to level-l, for the up-sampling layer, use a 1*1 convolution layer to compress the feature channels and then apply the interpolation to upscale the resolutions. For one-half down-sampling channels, use 3*3 convolutional layers with a stride 2 to complement the resolution and number of compression channels. Finally, a maximum pooling layer with a step size of 2 plus a 3*3 normal convolution layer with a step size of 2 is used for the quarter down-sampling channels to complete the resolution complementation and modify the number of channels accordingly. Next, the features of each scale after constant scaling are weighted and summed, which is calculated as follows.

$$y_{ij}^{l} = \alpha_{ij}^{l} \cdot x_{ij}^{1\to l} + \beta_{ij}^{l} \cdot x_{ij}^{2\to l} + \gamma_{ij}^{l} \cdot x_{ij}^{3\to l} \tag{2}$$

where, y_{ij}^{l} denotes the (i,j)-th feature vector of the output feature map, $x_{ij}^{1\to l}$, $x_{ij}^{2\to l}$, and $x_{ij}^{3\to l}$ are the output feature maps from different scales, and α_{ij}^{l}, β_{ij}^{l}, and γ_{ij}^{l} denote the spatial weights of the unified-to-l layer feature maps from different scales. These weights are learned adaptively by the network and are shared across all channels; their weight values are restricted to [0, 1] and satisfy the constraint: $\alpha_{ij}^{l} + \beta_{ij}^{l} + \gamma_{ij}^{l} = 1$, which can be calculated by the normalized exponential function. The calculation method is as follows.

$$\alpha_{ij}^{l} = \frac{e^{\lambda_{\alpha_{ij}}^{l}}}{e^{\lambda_{\alpha_{ij}}^{l}} + e^{\lambda_{\beta_{ij}}^{l}} + e^{\lambda_{\gamma_{ij}}^{l}}} \tag{3}$$

where,$\lambda_{\alpha_{ij}}^{l}$, $\lambda_{\beta_{ij}}^{l}$ and $\lambda_{\gamma_{ij}}^{l}$ are the control parameters for these weights, respectively, a 1*1 convolutional layer can be used to compute these weight scalar maps, λ_{α}^{l},λ_{β}^{l}, and λ_{γ}^{l} on

$x^{1\to l}$, $x^{2\to l}$, and $x^{3\to l}$ respectively. In this way, the network model, through standard backpropagation, can learn these parameters adaptively.

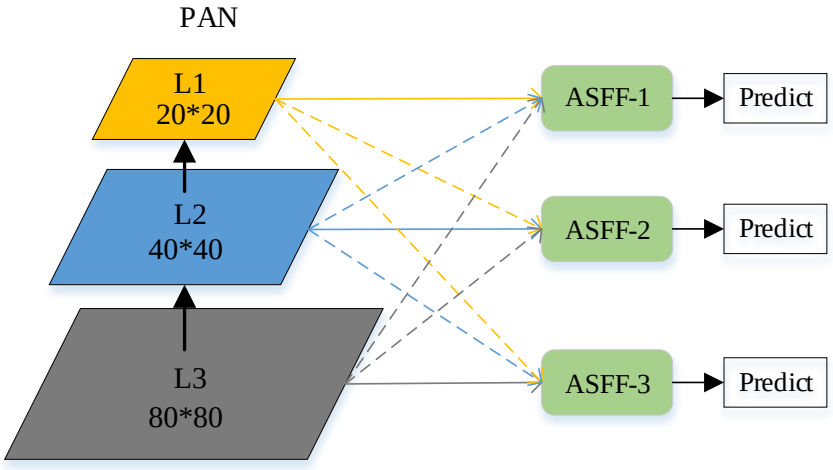

Figure 4. Structure of ASFF.

3.3. Introducing the Bottleneck Transformer Method

The gesture recognition task requires modeling long-range dependencies. The backbone network of YOLOv5s uses numerous convolutional layers to capture local information of the image, and the stack of convolutional layers improves the performance of the backbone network in extracting features but also increases the number of model parameters and reduces the inference speed of the model. In this paper, we introduce the bottleneck transformer method to optimize the backbone network of YOLOv5s. According to the experiments in the BotNet paper [20], replacing the convolution operator of the residual convolutional neural network with the multi-headed self-attentive operator can improve the recognition accuracy of the model and reduce the number of model parameters. We refer to this approach and improve the C3 module in the backbone of YOLOv5 to the C3BOT module. At the same time, considering the computer performance requirement of self-attention, replacing all C3 modules with C3BOT modules will bring huge training time consumption, so only the last layer of the backbone is replaced. Multi-Head Self-Attention (MSHA) treats position encoding as spatial attention, embeds two learnable vectors, which are viewed as spatial attention in both horizontal and vertical dimensions, and then multiplies the summed and fused spatial vectors in q to get content-position; moreover, content-position and content-content are summed to get the spatially sensitive similarity feature, which allows MHSA to focus on the appropriate region and makes the model converge more easily. Figure 5 shows the structure of MSHA. With the MHSA layer in the bottleneck transformer module, the global self-attentiveness is used to process and aggregate the information contained in the feature maps captured by the convolution, which reduces the parameters of the model, avoids the model bloat caused by the stacking of the convolutional neural network parameters, and increases the accuracy to a certain extent based on the speed of inference. Meanwhile, after the middle feature extraction layer (the sixth layer) of the backbone network, a jump-splicing layer is added to strengthen the feature fusion ability of the network for medium and large targets. The structure of the C3BOT module and optimized overall network structure is shown in Figure 6.

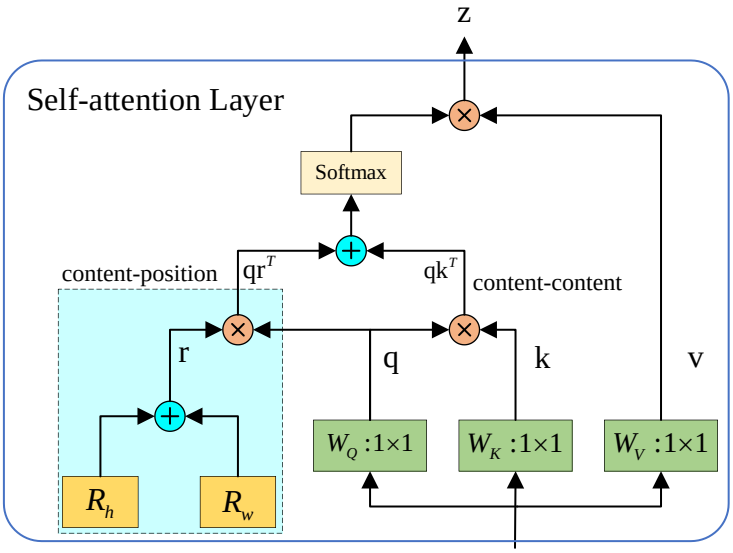

Figure 5. Structure of optimized MHSA. The attention logarithm is $qr^T + qk^T$, where q, k, r, and v represent query, key, position encoding, and value respectively. $\oplus$ and $\otimes$ denotes element-wise sum and matrix multiplication respectively, while 1×1 denotes point-by-point convolution.

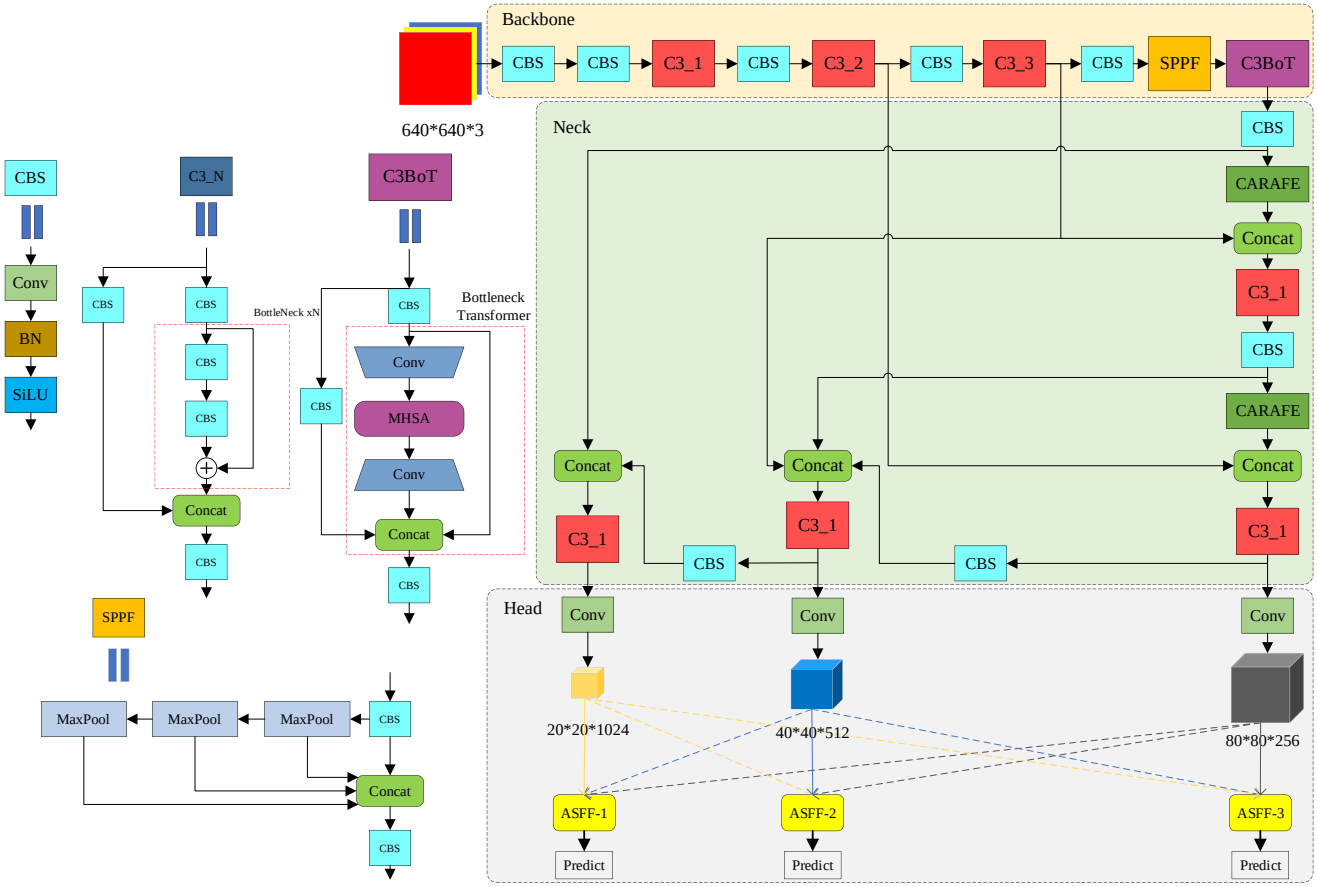

Figure 6. The C3BOT module and overall network structure of optimized YOLOv5s.

4. Results and Discussion

4.1. Experimental Environment and Dataset

The operating system used for the experiments in this paper is Windows 10. the CPU model is AMD Ryzen 5 2600X Six-Core Processor @3.6 GHz, the running memory is

16 GB, the GPU model is NVIDIA GeForce RTX 2060, the video memory size is 6 GB, and the memory size is 14 G. Use Pytorch deep learning framework, touch version 1.8.0. Use Python 3.8 as the experimental programming language, CUDA 11.1.102, and CUDNN 8.0.5 for graphics acceleration. Some specific parameter settings of the training are shown in Table 1.

Table 1. Training parameter settings.

Title 1	Parameter Value
Lr	0.01
Momentum	0.937
Weight_decay	0.0005
Batch_size	32
Epoch	1200

The original image data of the experiment come from the gesture data set in Baidu Paddle Paddle Developer Forum, including 2850 gesture pictures, and all images are manually labeled. In order to increase the diversity and complexity of the dataset, the selected gesture pictures include different gesture pictures taken from various angles, with 14 categories, namely 'one', 'five', 'fist', 'ok', 'heartSingle', 'yearh', 'three', 'four', 'six', 'Iloveyou', 'gun', 'thumbUp', 'nine', and 'pink'; each category is randomly divided into the training set, test set, and validation set in the ratio of 8:1:1, and LabelImg is used to annotate the images. The labeling information file is in text format, including the gesture category information and the coordinate information of the ground-truth bounding box, and the labeled images are shown in Figure 7.

Figure 7. Gesture datasets.

4.2. Experimental Environment and Dataset

The evaluation metrics of the experimental model in this paper use the precision (P) and the mAP. The precision (P) and recall (R) are calculated as follows.

$$P = \frac{TP}{FP + TP} \tag{4}$$

$$R = \frac{TP}{FN + TP} \tag{5}$$

where TP, FP, and FN denote the number of correctly recognized, incorrectly recognized, and unrecognized gesture pictures.

Take the recall (r) as the abscissa and the precision (p) as the ordinate, then draw a curve. The area enclosed by this curve, the abscissa, and ordinate axes is the AP value, and

finally, the arithmetic mean is calculated. The result of the calculation is the *mAP* value, and the specific calculation method is as follows.

$$AP = \int_0^1 p(r)dr \tag{6}$$

$$mAP = \frac{1}{m}\sum_{i=1}^{m} AP_i \tag{7}$$

4.3. Ablation Experiment

Ablation experiments with the addition of modules were conducted to verify the effectiveness of the above method in improving the gesture recognition results. The results obtained from the experiments with YOLOv5s as the baseline model were recorded as BASE, the results obtained from adding the ASFF module to the baseline model were recorded as AF, and the results obtained by replacing the CARAFE module were recorded as CA, the result obtained by improving the C3 to C3Bot module was recorded as BoT. The baseline model and the three improved models were trained for 1200 epochs under the condition that various training parameters were set consistently. The experimental results are shown in Figure 8.

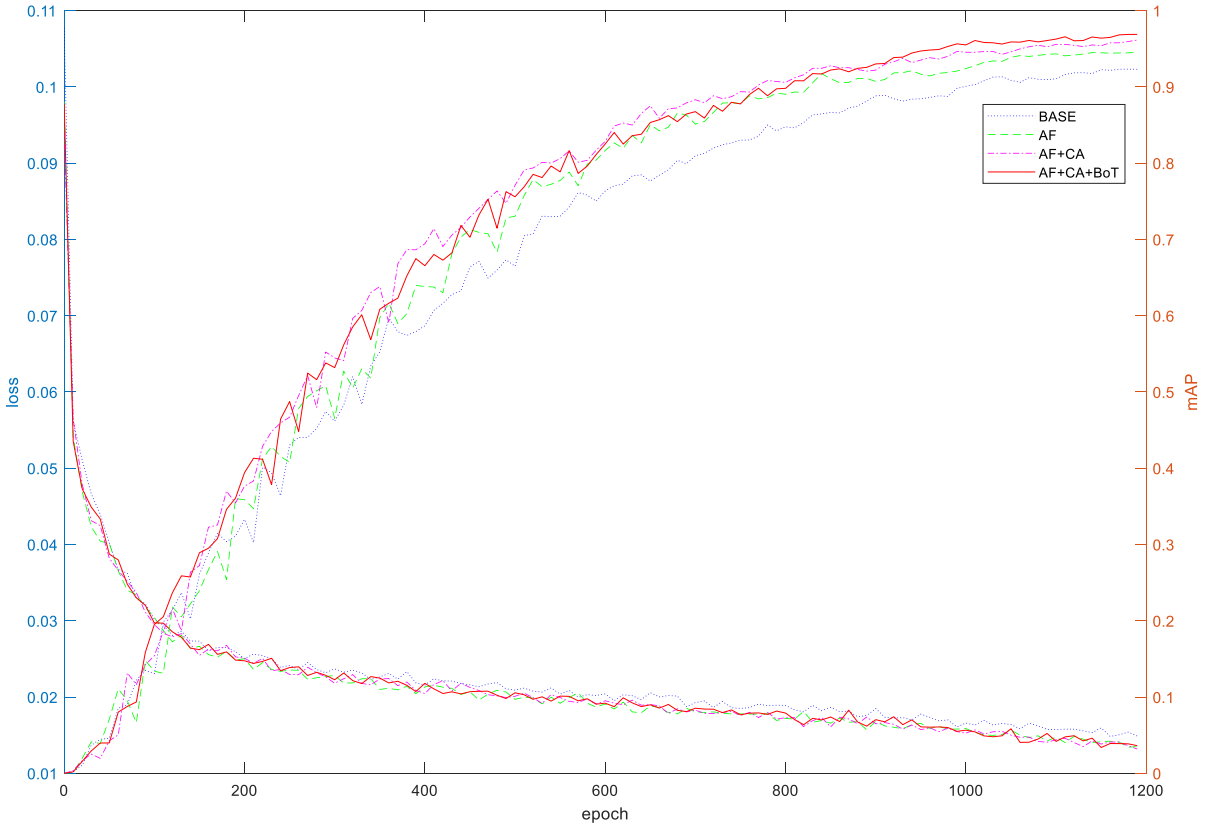

Figure 8. Results of the ablation experiment.

It can be seen from Figure 8 that the model converges faster in the first 2/3 training cycles at the beginning of the training, and the convergence speed is slower in the last 1/3 training cycles. After adding the ASFF module, the model converges faster, and the loss drops to 0.02 for the first time at about 425 epochs, which is about 150 epochs faster than the baseline model. The average accuracy does not increase significantly in the first 1/3 training epoch and is about 1.1% higher than the baseline model in the last 2/3 training epoch. On this basis, after replacing the nearest neighbor up-sampling with the CARAFE,

the model convergence speed is improved in the second half of the training, and the loss drops to 0.02 for the first time at about 470 epochs, which is about 100 epochs faster than the baseline model. The mAP value is better than the baseline model after 175 epochs, which is about 1.8% higher than that of the model with only ASFF added. Finally, after improving the C3 module into the C3BOT module, the convergence speed and MAP values of the model are basically the same in the first 2/3 training epoch as the AF + CAmodel; in the last 1/3 training epoch, the mAP value of the model gradually exceed the AF + CA model.

4.4. Results and Analysis

We named the improved algorithm YOLOv5-ACBoT. YOLOv5-ACBoT stabilized the accuracy and recall after training the hand gesture dataset for 1000 epochs, while the location loss and objectness loss decreased steadily after training for 200 epochs.

In order to verify the effectiveness of our improved model, we conducted comparative experiments using the YOLOv5-ACBoT model with the baseline model and some other commonly used object detection models. The experimental results are shown in Figure 9, Figure 10, and Table 2.

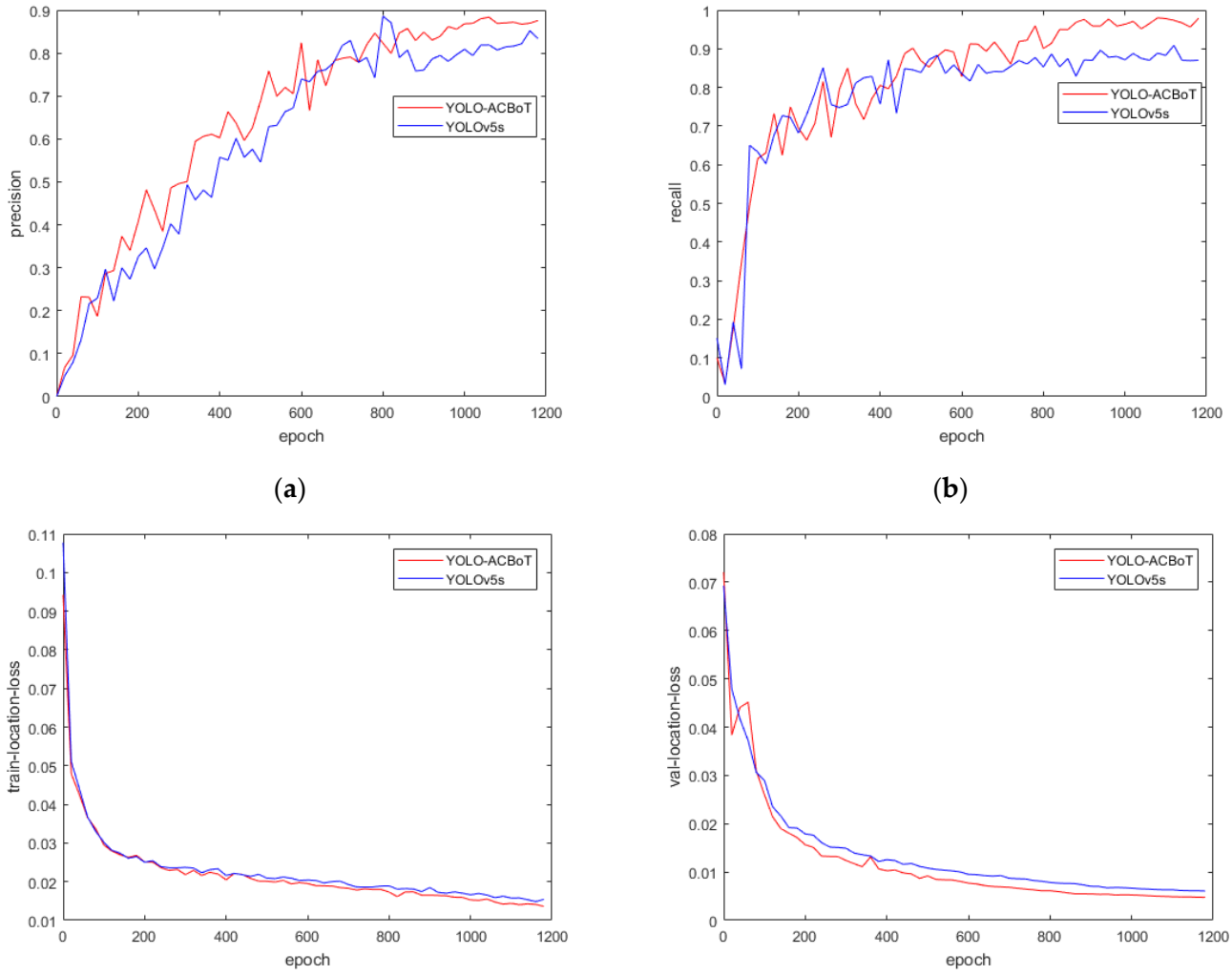

Figure 9. Four performance curves. (**a**) precision; (**b**) recall; (**c**) train-location-loss; (**d**) validation-location-loss.

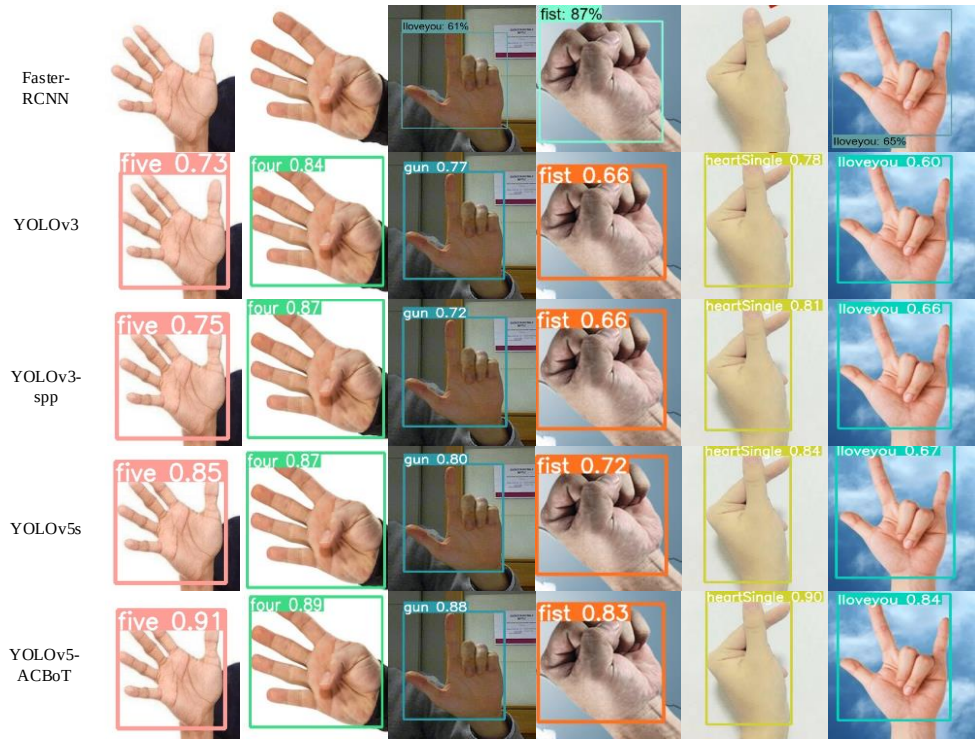

Figure 10. Comparison of partial detection results.

Table 2. Comparison table of experimental results.

Model	P/%	mAP/%	time/ms
SSD	61.3	74.5	56.1
Faster-RCNN	49.3	40.4	321.8
YOLOv3	74.3	84.2	16.8
YOLOv3-spp	82.0	89.2	27.5
YOLOv5s	82.6	93.7	16.6
YOLOv5-AF	82.1	94.8	17.3
YOLOv5-CA	83.1	94.6	13.4
YOLOv5-BoT	84.9	94.1	12.3
YOLOv5-CA + BoT	84.6	95.1	13.5
YOLOv5-CA + AF	84.4	96.6	17.8
YOLOv5-BoT + AF	85.6	95.3	17.2
YOLOv5-ACBoT	87.2	96.8	15.6
YOLOv8(SOTA)	92.7	97.9	16.1

From Figure 9, it can be seen that the precision and recall rates of the improved YOLOv5-ACBoT algorithm are higher than the original algorithm, the precision improvement process in training is relatively stable, and the loss of bounding box location and objectness loss are lower than the loss of the original algorithm.

As can be seen from Table 2, the inference time and model weight of the Faster-RCNN model is much larger than other one-stage detection methods, and its accuracy is low. The bottom half of the table (YOLOV5S-YOLOV5-ACBOT) shows the results of the ablation experiments for each module. Compared with the original YOLOv5s algorithm, the algorithms introducing ASFF, CARAFE, and BotNet have an average accuracy improvement of 1.1%, 0.9%, and 0.4%, respectively. The algorithm with the addition of ASFF has the largest accuracy improvement but reduces the algorithm inference speed. The algorithm with the introduction of BotNet has the smallest accuracy improvement but contributes the most in speeding up the model inference speed. The algorithm with the addition of CARAFE is in between the two in all aspects. This can also be seen in the following

dual-module addition experiments. After finally introducing all modules, the accuracy of our proposed YOLOv5-ACBoT model is the highest among these models, with a 3.1% improvement in mAP value and a 1ms speedup in inference compared to the baseline model. The SOTA method is higher in recognition accuracy compared to our proposed YOLOV5-ACBOT method. However, the large number of model parameters introduced by the new structure leads to a slower model inference speed of the SOTA method than our proposed YOLOV5-ACBOT method.

As shown in Figure 10, in the actual detection experiment, Faster-RCNN missed the detection of gestures 'five', 'fist', and 'heartSingle', and detected the gesture 'gun' as 'Iloveyou', so the detection results are poor. The actual detection effect of YOLOv3-spp is relatively good. Except that YOLOv5s has low confidence in 'Iloveyou' gesture detection and recognition, the rest of the gesture detection results are ideal. The improved YOLOv5-ACBoT has higher overall confidence for different gesture detection compared to the original YOLOv5s algorithm. It improves the mean average precision of YOLOv5s by 3.1% based on the accelerated detection speed, and the confidence and stability of the actual detection results are much higher than the original algorithm. Moreover, the overall detection performance of the algorithm is greatly improved.

5. Future Work

In this paper, we focus on model recognition accuracy and inference speed for gesture recognition. However, with the iteration of technology and various new methods proposed, there is still room for improvement in accuracy and inference speed. In the future, we will continue to optimize YOLOv5-ACBOT, use GSConv + Slim-neck to reduce model complexity and improve accuracy, and use reparameterization, pruning, and quantization to compress the number of model parameters to improve model inference speed. We will continue our research on gesture recognition and further optimize the model to expand the application of gesture recognition.

6. Conclusions

In this paper, we improve the YOLOv5s target detection algorithm and propose an improved YOLOv5-ACBoT algorithm. Changing the up-sampling method to the CARAFE method increases the receptive field and enhances the network's perception of the gesture area. We introduce the ASFF method to enable the network to better fuse features from various scales, the bottleneck transformer method to increase the accuracy while accelerating the inference speed, and add a skip splicing layer to enhance the feature fusion ability of the network for medium and large targets. The experimental results show that the improved algorithm improves the mean average precision by 3.1% based on t the faster detection speed, and the confidence of the overall detection results are improved significantly in the actual gesture detection, besides, the inference results are stable.

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

Funding: This research was funded by the National Natural Science Foundation of China (Grant Nos. 61263040 and 62262043), the Natural Science Foundation of Jiangxi Province of China (Grant No. 20202BABL202005), and the Nanchang Hangkong University "three small" extracurricular academic and scientific projects for students (Grant NO. 2022XG253, 2022XG280, 2022XG299).

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

References

1. Guo, Z.; Han, H.; He, L. Gesture Recognition Algorithm and Application Based on Improved YOLOV4. *J. North Univ. China (Nat. Sci. Ed.)* **2021**, *42*, 223–231.
2. Wu, X.; Zhang, Q.; Xu, Y. An Overview of Hand Gestures Recognition. *Electron. Sci. Technol.* **2013**, *26*, 171–174.
3. 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 European Conference on Computer Vision—ECCV 2016, Amsterdam, The Netherlands, 8–16 October 2016; Springer: Cham, Switzerland, 2016; pp. 21–37. [CrossRef]
4. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv Prepr.* **2018**, arXiv:180402767. [CrossRef]
5. Bochkovskiy, A.; Wang, C.-Y.; Liao, H.-Y.M. Yolov4: Optimal speed and accuracy of object detection. *arXiv Prepr.* **2020**, arXiv:200410934. [CrossRef]
6. 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.
7. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Adv. Neural Inf. Process. Syst.* **2015**, *28*. [CrossRef] [PubMed]
8. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask r-cnn. In Proceedings of the IEEE international conference on computer vision, Venice, Italy, 22–29 October 2017; pp. 2961–2969.
9. Lin, L. Research on Kinect Dynamic Gesture Recognition Technology Based on Multi-modal Feature Fusion. Master's Thesis, Wuhan University of Science and Technology, Wuhan, China, 2021.
10. Huang, S.; Chen, P.; Yang, T.; Hen, P.; Gong, X. Gesture recognition based on leap motion and its application in virtual installation of large structural parts. *Mod. Radar* **2021**, 1–6.
11. Liu, C.; Yang, S.; Zhao, S.; Bai, Y. A Method of Hand Image Seamentation from Complex Backgrounds. *J. Hebei Norm. Univ. Sci. Technol.* **2007**, *21*, 46–49.
12. Tan, T.; Guo, Z. Research on location and gesture recognition of hand based on binocular stereo-vision. *Comput. Eng. Des.* **2012**, *33*, 259–264.
13. Ma, K.; Zhang, Q. Research on hand gesture recognition based on structure analysis. *Inf. Technol.* **2012**, 81–83. Available online: https://wenku.baidu.com/view/d390868fb9d528ea81c779e3.html?_wkts_=1674624210801&bdQuery=%E5%9F%BA%E4%BA%8E%E7%BB%93%E6%9E%84%E5%88%86%E6%9E%90%E7%9A%84%E6%89%8B%E5%8A%BF%E8%AF%86%E5%88%AB%E7%A0%94%E7%A9%B6 (accessed on 15 December 2022).
14. Bai, L.; Pen, Y.; Lu, A.; Yu, S.; Zhang, X. Gesture recognition based on convolutional neural network in complex background. *Comput. Eng. Des.* **2020**, *41*, 3199–3203.
15. Wang, J.; Zhu, X.; Wu, X. Gesture interaction method based on fusion of static gesture features and gesture track features. *Foreign Electron. Meas. Technol.* **2021**, *40*, 14–18.
16. Xie, L.; Zhong, Z.; Qiao, D.; Gao, X. SSD Gesture Recognition Algorithm with Multi-scale Convolution Feature Fusion. *Comput. Technol. Dev.* **2021**, *49*, 79–87.
17. Lin, L.; Tao, J.; Wu, H. Gesture Recognition Technology Based on YOLOv3. *J. Jianghan Univ. (Nat. Sci. Ed.)* **2021**, *49*, 79.
18. Yu, X.; Yuan, Y. Hand Gesture Recognition Based on Faster-RCNN Deep Learning. *J. Comput.* **2019**, *14*, 101–110. [CrossRef]
19. Wu, X.; Zhang, J.; Xu, X. Hand Gesture Recognition Algorithm Based on Faster R-CNN. *J. Comput.-Aided Des. Comput. Graph.* **2018**, *30*, 468–476. [CrossRef]
20. Srinivas, A.; Lin, T.-Y.; Parmar, N.; Shlens, J.; Abbeel, P.; Vaswani, A. Bottleneck transformers for visual recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 16519–16529.