Optimized Object Detection Based on The Improved Lightweight Model Mini Net

*Qi Chen, Xinyi Gao, Renjie Li, Yong Zhang**

Information Engineering Department, Tianjin University of Commerce, China
chenqi@tjcu.edu.cn, 1275475886@qq.com, 1300189009@qq.com, zhangyong@tjcu.edu.cn

Abstract

This paper proposes a Mini Net lightweight model that can be used for real-time detection. This model works together with Mini Lower and Mini Higher, which greatly improves the detection efficiency while ensuring the accuracy. The Mini module designs both the batch normalization layer and the excitation function at the front end of the module, which realizes efficient convolution, greatly reduces the amount of parameters and computation, and introduces the nonlinearity brought by more layers in the spatial dimension, which can improve the performance of the module extraction capacity. Based on the Mini convolution module, a multi-stage training strategy is proposed. The first stage makes the system fast and stable. In order to improve the overfitting phenomenon of the system, the second and third stages use finer features to improve the detection of small targets, thereby improving the Model training efficiency and detection accuracy.

Keywords: Convolutional neural network, Lightweight model, Object detection, Image recognition

1 Introduction

Object detection is a kind of image segmentation based on geometric and statistical features of objects. It combines target segmentation and recognition, and its accuracy and real-time performance is an important capability of the whole system. With the development of computer technology and the wide application of computer vision principle, it has wide application value to use computer image processing technology to track and locate the target dynamically in real time.

In addition, the development of computer technology makes it possible to realize recognition and classification through machine learning, and has achieved a good target recognition effect. To construct a new method of object recognition and classification that can deal with large-scale data through machine learning has become one of the hot spots of people's urgent attention. Deep learning algorithms based on Convolutional Neural Network (CNN) have been widely used in machine learning. In order to improve the efficiency and accuracy of common neural network systems, this paper proposes a lightweight model for real-time detection, and validates the effectiveness of the model design through a series of ablation experiments.

2 Research Background

2.1 Reserch Status

The classic Le Net was born in 1998. Then the edge of CNN began to be overshadowed by hand-designed features such as SVM. With the proposal of ReLU and Dropout, as well as the historical opportunities brought by GPU and big data, CNN ushered in a historical breakthrough in 2012 - Alex Net [1].

Since then, Deep Learning has continued to develop, and the ImageNet Large-Scale Network Visual Recognition Challenge (ILSVRC) is ranked by Deep Learning every year.

As the model is researched more and more deeply, the top-5 error rate is also getting lower and lower, and by 2017, it dropped to around 2.25%. Similarly, on the ImageNet data set, the recognition error rate of the human eye is about 5.1% [2]. In other words, the recognition ability of the current Deep Learning model has surpassed that of the human eye. The model representative shown in Figure 1 is also a milestone representative of Deep Learning's visual development.

The main classical structures of CNN include Le Net, Alex Net, ZF Net, VGG, NIN, Google Net, Res Net, SE Net, etc., which are the oldest CNN models. In 1985, Rumelhart and Hinton and others proposed the BP neural network algorithm, which made the training of neural network simple and feasible [3]. At present, Deep Learning is still a little behind Cortes and Vapnic's Support-Vector Networks, but its development prospects are very impressive.

2.2 Research Purpose and Significance

Nowadays, many devices and equipment pay attention to whether the system can respond in real time. Accordingly, the timely response of the system means that the effectiveness of the system is good. At present, the focus of many research models is to build a good model. From a macro perspective, it is obvious that most of the time is spent on training, and the efficiency of the system becomes a problem; from a micro perspective, the redundancy of the convolution itself needs to be improved [4]. This paper focuses on the study of lightweight models, essentially analyzing the effectiveness of convolution parameters, and further improving the efficiency of the system.

*Corresponding Author: Yong Zhang; E-mail: zhangyong@tjcu.edu.cn
DOI: 10.53106/160792642024032502005

3 Convolutional Neural Network

3.1 Neural Network

The M-P neural network originated in 1943 is an artificial neural network widely used in machine learning. It is an abstract and simplified model constructed according to the structure and working principle of biological neurons. Each neuron is a multi-input single-output information processing unit, and there is a fixed time delay between the neuron input and output due to the synaptic delay. The existing neural network is an adaptive system with learning ability composed of a large number of neurons connected to each other.

3.1.1 Perception

In artificial neural networks, neurons have both excitation and perception properties. Similar to the nervous system, the perceptron proposed by Frank Rosenblatt is determined by the Equation (1). The n-dimensional input X_n and the weight K_n are multiplied and summed, and an adjustable bias b is added. After mapping by the excitation function $f(x)$, the output Y is obtained.

$$Y = f(\sum_{i=1}^{n} K_i X_i + b). \tag{1}$$

A single-layer perceptron can be regarded as the simplest forward neural network, consisting of an input layer, an output layer, and a set of trainable weight parameters. The multi-layer perceptron is composed of input layer, output layer and hidden layer. It has nonlinear characteristics and effectively solves the problem that single-layer perceptron is difficult to deal with linear inseparability. Its structure is shown in Figure 1.

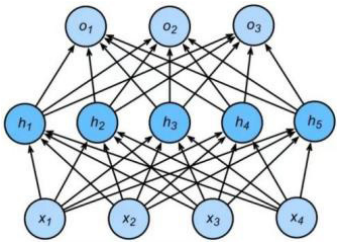


Figure 1. Multi-layer perceptron

3.1.2 BP Neural Network

The BP algorithm is a supervised learning algorithm, which is usually used to train a multi-layer perceptron. The data is input into the multi-layer perceptron network, and the actual value is found by forward propagation to the hidden layer until the output layer. The deviation function between the theoretical value and the theoretical value is back-propagated according to the gradient descent method; then the updated weight is used to minimize the deviation function to obtain the minimum value of the deviation, so that the model data can fit the real value as much as possible. The actual error value of the algorithm depends on the weight parameters during training [5]. Before the training starts, the initial weight will be randomly assigned, and a set of weight values that minimize the error will be obtained after multiple effective backpropagations. Since the initial weight value is randomly assigned by the system, there is also a certain error, and multiple fittings are required to find the optimal value.

The output of each neuron is Equation (2), n input O_i, multiplying and summing the weights to get net_j and the output O_j is obtained by mapping the excitation function ϕ.

$$O_j = \phi\left(net_j\right) = \phi\left(\sum_{i=1} \omega_{ij} O_i\right). \tag{2}$$

The weight is updated to formula (3), and the weight is adjusted to formula (4).

$$\omega_{ij} = \omega_{ij} + \Delta\omega_{ij}, \tag{3}$$

$$\Delta\omega_{ij} = -lr \times \frac{\partial L}{\partial \omega_{ij}}, \tag{4}$$

where L is the deviation function.

The partial derivative obtained by the chain rule is the Equation (5), and then the neuron is obtained. If the neuron is located in the output layer, then the output is equal to the predicted value p, which can be obtained by directly performing the partial calculation; if the neuron is located in the hidden layer, a recursive operation must be performed.

$$\frac{\partial L}{\partial \omega_{ij}} = \frac{\partial L}{\partial O_j} \times \frac{\partial O_j}{\partial net_j} \times \frac{\partial net_j}{\partial \omega_{ij}} = \delta_j \times O_i. \tag{5}$$

Among them,

$$\delta_j = \frac{\partial L}{\partial O_j} \times \frac{\partial O_j}{\partial net_j} = \begin{cases} \left(\frac{\partial L}{\partial O_j}\right) \times \frac{\partial}{\partial net_j} \phi\left(net_j\right), j = output\ neuron \\ \left(\sum_l \delta_l \omega_{jl}\right) \times \frac{\partial}{\partial net_j} \phi\left(net_j\right), j = hidden\ neuron \end{cases}$$

3.2 Convolution

With the continuous development of neural network and on the basis of adaptive learning system, the multi-level neural network realized by gradient descent method can effectively solve the problem of nonlinear processing of the system. In the early days of neurons, the full connection method was used to fit the data. When processing high-pixel images, the model is prone to overfitting.

3.2.1 Convolution Neural Networks

Convolutional neural network utilizes local perceptual field of view, weight sharing and spatial or temporal down-sampling to achieve invariance of translation, scaling and deformation, further improving the defects of fully connected network in the field of image recognition.

In order to perform high-level feature extraction, the topological structure of the input image is used to extract local features from the convolution kernel, and then the high-level features are obtained through the combination of step-by-step filtering [6]. The neurons in the feature map are obtained by convolving a group of local neurons in the previous layer with a single convolution kernel.

Setting a single convolution kernel as a set of weights and an optional bias, the convolution kernel can detect the same features in different regions, and then multiplies and sums to obtain a flat feature map. All neurons of the feature map share weights, thereby reducing the complexity of the feature map.

The calculated feature maps are constructed alternately through convolutional layers and down-sampling layers, thereby reducing the spatial resolution of the feature maps, and finally, at the end of the network, the fully connected layers and filters are combined to output the prediction results.

3.2.2 Neural Network Structure

The most intuitive way to improve the accuracy of the network is to increase the depth and width of the network, but this will produce a huge number of parameters, and easy to have overfitting problems. In recent years, many scholars have conducted researches on neural network structures, among which the achievements of deep learning in image recognition have been greatly improved, and many neural network structures have appeared, such as LeNet, AlexNet, VGGNet, ResNet, DenseNet, etc. [7]. The evolution of these structures is nothing more than pushing the structure deeper, or making it wider and wider, and these operations do lead to a significant increase in accuracy. However, when the structure reaches a certain depth, the problem of information loss often occurs. In DenseNet, the concatenation of the information of the previous layer is used to prevent information loss, which has a very good effect, but also causes a large number of parameters.

The initial structure of Inception is shown in Figure 2, which is to perform 1×1 convolution operation, 3×3 convolution operation, 5×5 convolution operation and 3×3 pooling operation on the input feature graph. However, this approach will cause too much computation. After the improvement of the model, 1×1 convolution operation is added before 3×3 convolution operation, 5×5 convolution operation and 3×3 pooling operation. This 1×1 convolution operation is helpful to reduce the thickness of the feature map and reduce the amount of calculation [6].

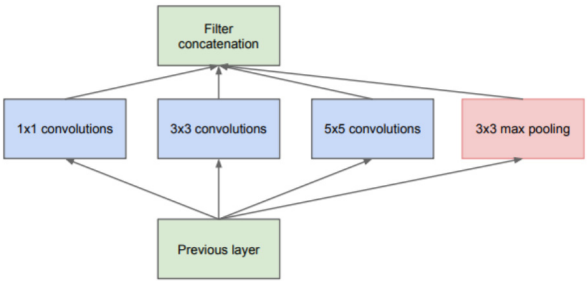


Figure 2. Inception structure

In the high-level feature space, the distance between them is relatively far, so the number of large-sized convolution kernels is relatively large, and it is difficult to avoid the increase in the amount of calculation. Therefore, before convolution of 3×3 and 5×5, it is pooled with 3×3, and then 1×1 convolution that can interact with channels and reduce the amount of data calculation is added. While extracting high-level features, this model can control its space and time complexity within a reasonable range and has a certain accuracy. The model is shown in Figure 3.

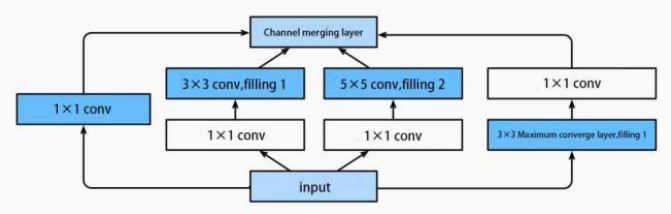


Figure 3. Improved convolution model

3.2.3 Residuals

Previous studies have found that the deeper the neural network structure, the lower the accuracy, so simply deepening the network layer is not the best way. ResNet proposed a residual connection, also known as shortcut connection, which transfers the information of the previous layer to the next layer [8]. In logic, when more than one layer, if the extra layer is redundant layer, at least will retain the information of the previous layer, so as to prevent the problem of training degradation.

For residual learning, Figure 4 is the residual learning unit. Let the input be x, the result after one layer is $F(x)$. At this time, the network structure without residual connection, the next layer of input is only $F(x)$, and it is easy to degenerate to a certain depth training. However, there is a shortcut path under the ResNet network structure, so the input of the next layer is $F(x) + x$. If this layer does not learn new features, there is at least the previous layer of information to ensure that the network model will not degenerate [9].

The residual block is shown in Figure 5.

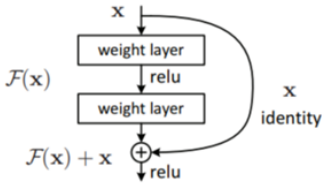


Figure 4. Residual learning unit

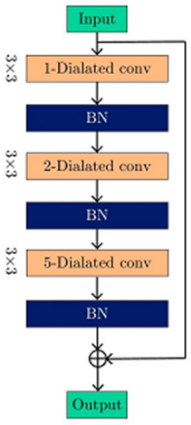


Figure 5. Residual module

It is connected to the original input layer x through a shortcut method, and the residual function expression $F(x) = H(x) - x$ is obtained, and then $H(x)$ is obtained by element-level addition, and the corresponding output value is obtained through the excitation function mapping. The introduction of residual function mapping can highlight small changes in branches, making the weights more sensitive to branch changes, thereby reducing the training difficulty of the model.

3.3 Lightweight Convolution

In order to solve the efficiency problem of the system, lightweight convolution is introduced. The lightweight model is mainly composed and designed by the convolutional layer of the model. Usually, group convolution and depthwise convolution, which are different from traditional convolution, are introduced in the convolution layer.

3.3.1 Group Convolution

The traditional convolution is to perform convolution processing on all feature channels of the previous layer, as shown in Figure 6. The group convolution is to first group the feature channels so that they can be operated on different GPUs, and different convolution kernels process the grouped channels of the previous layer, as shown in Figure 7.

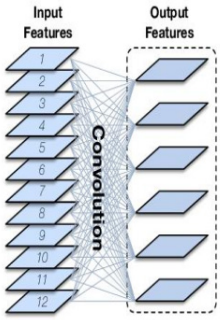


Figure 6. Traditional convolution

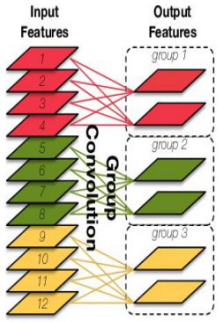


Figure 7. Group convolution

Assuming that the number of feature channels in the previous layer is C_{in}, after the convolution kernel operation of this layer, the output channel is C_{out}, the width of the convolution kernel is K_w, and the height is K_h, then its single convolution kernel size is: $K_w \times K_h \times C_{in}$, ignoring the bias set parameters, the parameter quantity of this layer is: $K_w \times K_h \times C_{in} \times C_{out}$; if the group convolution kernel of equal height and width is used, the channels of the previous layer are divided into G groups, and the parameter quantity is: $(K_w \times K_h \times \frac{C_{in}}{G} \times \frac{C_{out}}{G}) \times G = K_w \times K_h \times \frac{C_{in}}{G} \times C_{out}$. Compared with the general convolution kernel, the parameter amount of group convolution is $\frac{1}{G}$ times of the original.

3.3.2 Depth Wise Convolution

The depth wise convolution model is shown in Figure 8. If the number of feature channels in the upper layer is C_{in}, and a depth wise convolution kernel of equal height and width is used, then the size of a single convolution kernel is only $K_w \times K_h \times 1$, and the parameter quantity of this layer is also only $K_w \times K_h \times 1 \times C_{in}$, the parameters are greatly reduced.

Figure 8. Depth wise convolution

3.4 Object Detection

3.4.1 Two Stages Algorithm

The main algorithm of Two Stages is R-CNN, which converts the detection problem into a classification problem, uses a selective hierarchical grouping method to extract candidate regions, obtains multiple regions through an image segmentation algorithm, and merges multiple candidate frames layer by layer according to the similarity [10]. Then scale each candidate frame to a fixed size, input the convolutional neural network for feature extraction, and then send it to the SVM for classification to obtain the accurate position.

FAST-R-CNN improved the shortcomings of R-CNN, input the original image into the convolutional neural network at one time, and sent the finally obtained features to the pooling layer to extract the corresponding feature regions, and realized the candidate frame. Maximum pooling, which outputs a fixed-size feature map, solves the problem that the fully-connected layer needs a fixed input, and the scaling of the feature area causes distortion [11].

Using selective search to extract candidate regions will take a lot of detection time. Extract the candidate regions into the convolutional neural network, introduce the region generation network, and judge the category and background of the feature map output by the convolutional layer. According to the obtained candidate box corresponding to the output feature map by the previous convolution network, it is input into the pooling layer, and then sent to the softmax classifier and the correction boundary filter respectively to obtain the final prediction result [12].

3.4.2 SSD Algorithm

The SSD is one of the representative algorithms in object detection. References 15 discussed the method of SSD implementation. In this paper, the feature extractor is VGG16. After the Conv4_3 layer in the VGG16 architecture, each dimension reduction feature map will be used for object detection. A total of six branches of the feature map in the SSD are used for object detection, and the objects are detected from large to small. Because the small objects will disappear after reducing the dimension too much, the large part of the feature map is mainly used to detect small objects, while the small part of the feature map is used to detect large objects.

For the selection of boxes, SSD has a set of Default boxes. SSD contains two parameters, one is scale and the other is ratio. There will be different Default boxes for different sizes of feature maps. The Figure 11 is the SSD structure diagram. In the SSD architecture, the feature maps of 6 sizes have different numbers of Defaultboxes, which are 4, 6, 6, 6, 4, 4, respectively [13]. Most objects are in the middle of the image. For SSD output results from the past classifier can only obtain the probability of various categories, developed to obtain the location, location information are x coordinates, y coordinates, width and height.

3.4.3 One Stage Algorithm

The One Stage algorithm in target detection directly regresses the category probability and position coordinate value of the object, which is faster than the control of pre-extracting candidate frames by the hierarchical grouping feature extraction method, and can realize real-time detection. One Stage algorithm is representative of the YOLO series algorithm, which is a single pipeline as a whole and directly returns the category and accurate position of the bounding box from a single image [14]. Its advantages are fast detection speed, easy training, and higher accuracy than the R-CNN series. YOLO neural network structure is shown in Figure 9.

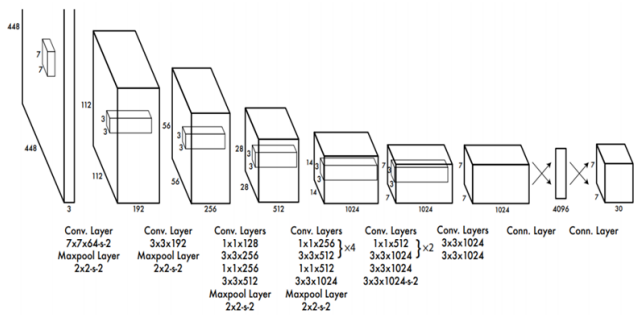


Figure 9. YOLO neural network structure

YOLOv3 scales the original image to $H \times W$ and inputs it into a single network [15]. After processing by the convolutional layer and the pooling layer, the feature map is divided into $S \times S$ cells, and each cell predicts B bounding boxes, each boundary Box prediction $(5 + C)$ values, including relative center coordinates of the box (d_{cx}, d_{cy}), relative offset width and height (d_w, d_h), confidence score c and C class conditional probabilities. d represents relative cell offset.

During testing, as shown in formula (6), the confidence score box_{score} of a specific category is obtained by multiplying the conditional probability C of the category of the prediction box with the confidence score c, and the final prediction result is obtained by filtering and non-maximum discrimination based on the confidence score of all prediction boxes.

$$\begin{aligned} &\Pr(class_i \mid object) \times \Pr(object) \times (IOU_{pred}^{truth}) \\ &= \Pr(class_i) \times (IOU_{pred}^{truth}) \end{aligned} \tag{6}$$

In formula (6), if the target is included, its center point falls into the cell and $\Pr(object) = 1$; if the target is not included, $\Pr(object) = 0$. IOU_{pred}^{truth} represents the area ratio of the ground-truth box to the predicted box.

In order to avoid the problem that the required convergence time is too long due to unstable training, YOLOv2 and YOLOv3 use the relative position of the center coordinate corresponding to the upper left corner of the cell to obtain the sigmoid function ϕ, so that the center coordinate can fall within the cell [16].

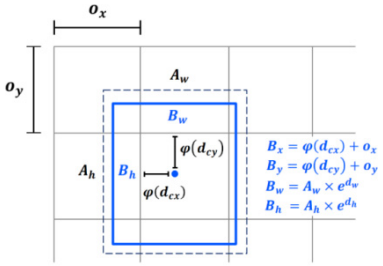


Figure 10. Exact position of bounding box in micrograph

As shown in Figure 10, the width of the feature map is B_w, the height is B_h, the width of the bounding box is A_w, the height is A_h, the coordinates of the center of the bounding box are (d_{cx}, d_{cy}), and the offset width and height are (d_w, d_h). The distance of the cell from the upper left corner feature map is (o_x, o_y), and (B_x, B_y) can be determined by formula (7) and formula (8), and then the exact position of the center coordinate of the prediction frame on the feature map can be obtained.

$$B_x = \phi(d_{cx}) + o_x. \tag{7}$$

$$B_y = \phi(d_{cy}) + o_y. \tag{8}$$

4 Mini Net Model

Based on the design of the Mini convolution module, this research uses convolution modules with different properties in the high-level and low-level layers to extract features. By reducing the amount of parameters and computation of the model, real-time detection is achieved, and a certain accuracy is guaranteed.

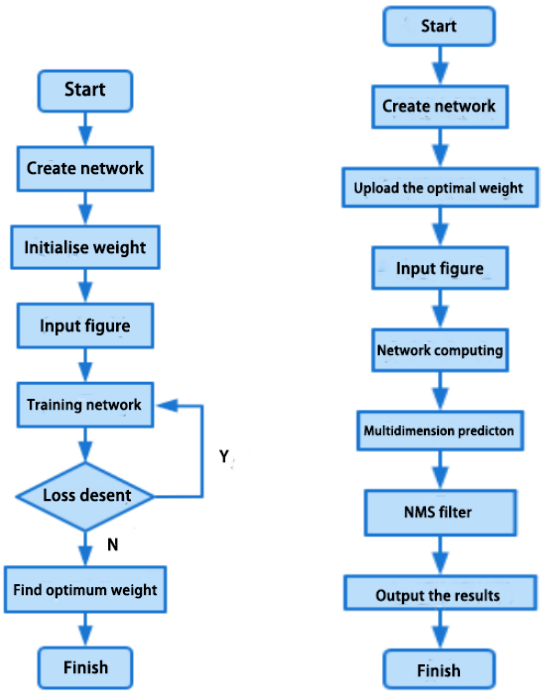
4.1 Detection System

The original image is scaled and converted to a fixed size and input into the system. After being processed by the Mini Net model, the exact position of the target is directly output, and the single-channel system mode is used to improve the detection efficiency. The input images are all color RGB data, and the original image is not subjected to grayscale processing to reduce the dimension.

4.1.1 System Flow

The Mini Net detection system is divided into a training phase and a testing phase. The training phase is shown in Figure 11(a). The Mini Net network is established and the

weights are initialized. The scaled images of the training set are input to the network for forward propagation, and the deviation function is calculated. Then the gradient descent method is used for backward propagation to adjust the weight value. After many times of training, the weight parameter of the feature map is finally obtained. In the test stage, as shown in Figure 11(b), a Mini Net network is established, the optimal weight is input into the network, the scaled image is input into the network for forward propagation to realize multi-scale prediction, and the final detection result is obtained by NMS filtering.



(a) Training phase (b) Test phase

Figure 11. Mini net detection system

4.1.2 Input Preprocessing

Firstly, the width-height ratio of the input size of the model to the original size is calculated, and the minimum value is taken as the scaling ratio. In order to avoid the scaled boundary being larger than the original image boundary, crop the original image was cropped and then multiplied by the scaling ratio to get the new size. Since the width and height of the input image are both 416, but the aspect ratio of the original image is not necessarily 1:1, the image is proportionally scaled and placed on a grayscale negative whose width and height are both 416 and RGB is specified as (128, 128, 128) [17]. In order to avoid overlapping effect in the training process, the intermediate value 128 is selected as the background color for training.

4.1.3 Model Architecture

The Mini Net model is composed of different convolution modules in the high and low stages. In the low-level stage of the model, the Mini Lower module is used to extract low-level features, and in the high-level stage, the Mini Higher module is used to extract high-level features. Finally, the predictions from the two stages are taken as output. The overall framework is shown in Figure 12.

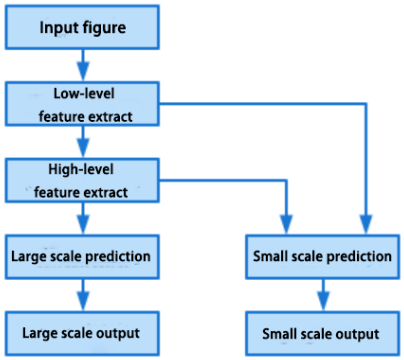


Figure 12. Mini Net architecture

4.2 Operation Model

The operation mode of the Mini Net model is YOLOv3 based on the One Stage algorithm. The entire task is regarded as a regression problem. There is no need to pre-extract candidate frames. The image is directly input into the neural network for processing, and the feature map is divided into grids. The global target detection is carried out in the feature space. Considering the computational complexity of the lightweight model, two scales of YOLOv3-tiny are used for output prediction, and the bias regression control of the YOLOv3 prior box is used for training and testing.

4.2.1 Traning Phase

The Mini Net model uses supervised learning for mode training, including two inputs I_1 and I_2, and calculates the output end O of the deviation function. The first input I_1 is the label value of the real box G corresponding to the target in the picture, and I_2 is the *RGB* value of the scaled image.

i) Input terminal I_1

The ground-truth box corresponding to each target in the training set contains five label values, namely: the upper left corner coordinate of the boundary ($rx_{\min}$, $ry_{\min}$), the lower right corner coordinate ($rx_{\max}$, $ry_{\max}$), and the category C_{id}. First, the mark values of the real box are transformed: $G_{cx} = \frac{rx_{\min} + rx_{\max}}{2}$, $G_{cy} = \frac{ry_{\min} + ry_{\max}}{2}$, $G_w = rx_{\max} - rx_{\min}$, $G_h = ry_{\max} - ry_{\min}$, and then the transformed mark values are normalized, as shown in Equation (9):

$$\left(G_{ncx}, G_{ncy}, G_{n\omega}, G_{nh}\right) = \left(\frac{G_{cx}}{\omega}, \frac{G_{cy}}{h}, \frac{G_{\omega}}{\omega}, \frac{G_h}{h}\right). \tag{9}$$

After the format conversion is completed, each ground-truth box G corresponds to the a priori box A, so as to determine the training order of the a priori boxes.

ii) Input terminal I_2

The scaled original image is normalized to [0, 1] before model training, which helps to stabilize the training, and then the normalized data is input into the Mini Net network, and y_{pred} is output after forward propagation. Finally, the two convolutional layers output feature maps of different sizes respectively. The tensor shape of small-scale output is: [*bs*, 13, 13, 18], and the tensor shape of large-scale output is: [*bs*, 26, 26, 18], 18 means that three prediction frames are output, and each prediction frame predicts six values.

iii) Input terminal *O*

The output of the entire system is the loss layer. The loss layer receives two inputs, namely the actual label value y_{true} and the model predicted value y_{pred}, and then calculates the error value between the two through the deviation function. After several trainings, the most suitable weight is found. Dimension conversion is required before the loss function is calculated, as shown in Table 1.

Table 1. Dimension transformation

Predicted and labeled values		The tensor shape of small-scale	The tensor shape of large-scale	Item system of tensors
y_{pred}	Before conversion	[*bs*,13,13,18]	[*bs*,26,26,18]	-
	After conversion	[*bs*,13,13,3,6]	[*bs*,26,26,3,6]	[d_{cx}, d_{cy}, d_w, d_h, c_{po}, C_{po}]
y_{true}		[*bs*,13,13,3,6]	[*bs*,26,26,3,6]	[G_{ncx}, G_{ncy}, G_{nw}, G_{nh}, c_t, C_t]

Here, the cross entropy is used as the deviation function, and the value of the cross entropy is expressed in the form of probability. Because the real value and the predicted value of the loss layer are both six items, the multi-task training mode is adopted.

4.2.2 Test Phase

The test stage of Mini Net is calculated by the weight obtained in the training stage. First, the original image is scaled to the size required by the model, *RGB* is normalized and forwarded, and the prediction results of different sizes are output, and the confidence score c_p is multiplied with the conditional probability C_p to get a specific confidence score box_{score}, filter out the prediction frame of the confidence score $box_{score} < 0.2$, and then perform NMS processing on the filtered image to eliminate the overlapping frame until all the prediction frames are processed and the prediction result is obtained.

4.3 Mini Block Design

Here, different forms of lightweight models are designed according to the features extracted at different stages, namely the basic features Mini Lower based on low-level extraction and the high-level features Mini Higher based on high-level extraction, and then the two are combined to obtain a lightweight model Mini Net.

4.3.1 Mini Lower Block

The Mini Lower block mainly uses group convolution, and the convolution kernels of different groups act on the grouped feature maps respectively. First, 1×1 convolution is used for information fusion on the features of the input block, and the number of convolutions is set as half of the number of input feature channels, so that feature interaction can be realized and the amount of parameter calculation can be reduced; then the processed features are processed. The Group convolution operation is performed. Considering that too many groups will lead to fragmentation of features, so it is only divided into two groups, and a 3×3 convolution operation is performed. The number of convolution kernels depends on the number of output channels after the combination. The channel-level merging strategy is adopted here, which on the one hand can greatly reduce the amount of parameters, and on the other hand can reduce the amount of unnecessary parameter calculations caused by convolution [18].

4.3.2 Mini Higher Block

The Mini Higher block mainly uses depth wise convolution. Each convolution kernel operates on a single feature channel corresponding to it, and uses a single convolution kernel for operation processing. Using depth wise convolution can greatly reduce the amount of computation [19]. Based on the efficiency brought by Mini Lower's introduction of 1×1 convolution at the front of the block with pooling layer and excitation function, a 1×1 convolution combination is also used here, and the number of convolution kernels is set to reduce the $0.5C_{in}$ parameter. The amount of calculation is followed by depth wise convolution, and the last 1×1 convolution is equivalent to fusing the features output by depth wise convolution to fit the desired position of the target feature.

4.4 Detection Model

After the image is input into the model, the upper and left boundaries of the input image are filled with 0, so that the width and height of the feature map are reduced to half of the original; then five Mini Lower blocks are used to extract basic features, and the feature channels are doubled, interspersed with four pooling layers are used to reduce the size of the feature map. After this stage is completed, the size of the feature map is reduced to 13×13, and the number of channels is increased to 512; Mini Higher is used for extraction of high-level features; the output ends of the final two scales use 1×1 convolution for prediction.

5 Experimental Result

5.1 Introduction to The Development Environment

In this study, all operations are performed on the central processing unit, instead of a circular processor with a large number of parallel computing capabilities, the real-time detection system is designed with Python, the neural network is built based on TensorFlow and the clustering algorithm Keras, and the data set WIDER FACE is used for supervised learning evaluation of the model. No additional other data sets were used.

5.2 Data Preprocessing

5.2.1 WIDER FACE

The dataset is a subset of WIDER, and all images are obtained through Google and Bing searches, which are processed by category, and images with high similarity are deleted to ensure the richness of the samples. Figure 13 shows large-scale data with diverse attributes, which can fully guarantee positive and negative samples, and does not require additional data sets.

Figure 13. WIDER FACE data set

5.2.2 Dataset Preprocessing

The characteristics of the training set itself will affect the generalization ability of the model. It is necessary to filter and screen the data set to ensure that the data is real and effective.

5.2.3 Filtering and Screening

First, the problematic data were screened, ten inconsistent images with a mark value of 0 were deleted, and bounding boxes with width $w \leq 0$ or height $h \leq 0$ were eliminated. Table 2 shows the preliminary processing of the dataset and validation set.

Table 2. Preliminar processing of dataset and verification set

Wider face		Original quantity	Number of deletions	Effective number
Training sets	Image	12,880	4	12,876
	Bounding box	159,420	27	159,393
Validation set	Image	3,226	14	3,212
	Bounding box	39,422	7	39,415

5.2.4 Statistic and Clustering

First, the bounding boxes of the training set are counted, and the six a priori boxes required for the experiment are clustered. By using the Euclidean distance function of K-means, the distances between all data points and the centers of each cluster are calculated. In order to reduce errors caused by statistics and clustering, the selection of initial values is to select six points from all data points, rather than random arbitrary values [17].

5.3 Training Method

Based on the effective design of the Mini convolution module, the model is more stable during the overall training process, so the detection model is trained on the detection data set. A multi-stage training strategy is adopted on the data set, and hyperparameter adjustment is matched in a specific stage, thereby improving the training efficiency of the model and the accuracy of detection.

5.3.1 Hyperparameters and Optimizer

This study does not use pre-fixing the period, but after traversing each period of the entire training set, the weight value is updated by the period, and the average error is calculated on the verification set to judge the training effect. Common setting methods of batch size *bs* are BGD, SGD and MBGD. The BGD method is to input all samples into the network and traverse all the samples once to obtain the updated weights. This method requires too much calculation and the convergence speed is very slow; in the SGD method, only one sample is selected for input into the network for each training, which avoids the need for a large number of calculations, and the model can usually converge by traversing a few samples; the MBGD method selects a batch of m samples for each training to input the network, divides the overall data into several batches, and then determines the gradient direction of the batch weight update, so that The data is stable and does not cause a large amount of calculation problems. The effectiveness of the weight update will affect the data fitting ability of the model. The weight with high nonlinearity will be adjusted more to fit each data point, but this is prone to overfitting, as shown in Figure 14.

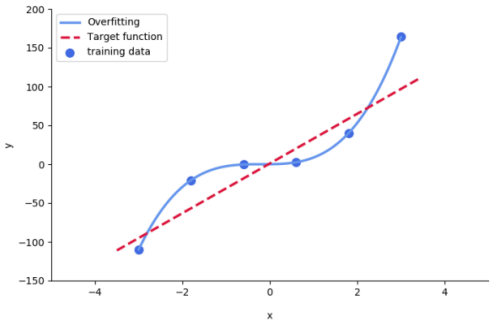


Figure 14. Over fitting phenomenon

The optimizer uses MBGD with L2 for weight update, and then uses Adam with Momentum characteristics and RMS adaptive supervised learning as the optimizer.

5.3.2 Data Augmentation

Here, different processing strategies are used for different training stages. Data augmentation is equivalent to increasing the amount of more diverse data, thus avoiding overfitting and improving detection accuracy. Multi-scale scaling is divided into three steps: the first step is to randomly generate a new aspect ratio within the valid range, so that the ratio of the detected image and the original image will not be too different; the second step is to randomly obtain a scaling value s to avoid too large or too small scaling ratio, which will affect the boundary of the input size and cut off too much original data; the third step is to set the letterbox mode of RGB = (128, 128, 128) to find the most suitable position for the negative.

5.4 Analysis of Experimental Results

First, in order to illustrate the effectiveness of Mini Lower, a comparative experiment is conducted between Mini Lower and other modules, and it is discussed whether the model can improve the overall detection effect after incorporating more refined features. All algorithms were tested on a dataset jointly constructed by WIDER FACE and LFW. Mini Net designed with Mini Lower module can detect more accurately and instantly. The following control experiments are designed according to Table 3. The Mini Net-A group has no residual branch, so the number of convolution kernels increases; the Mini Net-B group changes the channel combination in the Mini Lower module to the element addition in the residual module. The Mini

Net-C group guarantees that the number of combined channels is 384 instead of the 416 channels in Mini Net. The experimental results are shown in Table 4.

Table 3. Multistage training

Stage of training	Training sets	Number of batches	Initial learning rate /%	Data augmentation
The first stage	First subset	32	0.1	-
The second stage	Second subset	16	0.01	-
The third stage	Third subset	16	0.01	√

Table 4. Model comparison

Model	Number of parameters	Accuracy
Mini Net	923,420	0.9308
Mini Net-A	1,021,876	0.9235
Mini Net-B	2,334,292	0.9321
Mini Net-C	914,876	0.9125

Secondly, the Mini Net is evaluated and compared with YOLOv3-tiny, Faster R-CNN, SSD-Lite and YOLOv5. The comparison results are shown in Table 5. Mini Net has more advantages than YOLOv3-tiny in terms of parameter number and accuracy. Compared with YOLOv5, although the accuracy is lower than YOLOv5l and YOLOv5x, the number of parameters and testing time are more advantageous. Faster R-CNN, a large two-stage network, has low detection rate but high detection accuracy in face detection task. Although the detection accuracy of this method is lower than that of Faster R-CNN, its speed is nearly twice as fast. Mini Net reduces model parameters and computational complexity to better meet the needs of detection speed in practical applications.

Table 5. Experimental results

Model	Number of parameters	Testing time (ms)	Accuracy
YOLOv3-tiny	8756342	725.2	0.9179
YOLOv5s	1029734	834.8	0.9213
YOLOv5m	3053482	928.3	0.9299
YOLOv5l	5857324	1046.4	0.9554
YOLOv5x	9837264	1241.83	0.9421
SSD-Lite	8854632	723.5	0.9295
Faster R-CNN	9925463	1347.6	0.9772
Mini Net	924327	728.1	0.9308

6 Conclusion

For the lightweight model Mini Net, the target features can be effectively extracted with only 0.92 × 106 parameters. Due to the redundancy problem of convolution itself, compared with full convolution using a large number of parameters to learn features, Mini Lower and Mini Higher, which are designed according to low-order features and high-order features, can learn features more accurately. Adding any operation to the convolution module will increase the computational load of the model, which in turn affects the detection speed. In the lightweight model, both the batch normalization layer and the excitation function are designed at the front end of the module. In the process of data stacking, the interaction of features makes the data shared among various parts, thereby reducing unnecessary data calculations.

Compared with YOLOv3-tiny, SSD-Lite, Faster R-CNN and YOLOv5 models, the lightweight model Mini Net has far fewer parameters than these models, and has strong data analysis ability and better computing speed. The Mini Net is a high performance convolutional neural network suitable for low specification mobile devices.

Acknowledgment

This work was supported in part by the Tianjin Natural Science Foundation No. 20JCYBJC00320 and No. 23JCQNJC00840. Tianjin Science and Technology Project of Special Correspondent No. 22YDTPJC00290; Tianjin Postgraduate Research and Innovation Project (Intelligent Connected Vehicles) No. 2021YJSO2S37 and No. 2022SKYZ389.

References

[1] A. Krizhevsky, I. Sutskever, G. Hinton, ImageNet classification with deep convolutional neural networks, *Communications of the ACM*, Vol. 60, No. 6, pp. 84-90, June, 2017.

[2] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. Berg, F. Li, ImageNet large scale visual recognition challenge, *International Journal of Computer Vision*, Vol. 115, No. 3, pp. 211-252, December, 2015.

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, *2015 IEEE Conference on Computer Vision and Pattern Recognition*, Boston, MA, USA, 2015, pp. 1-9.

[4] L. Sun, Z. Liu, X. Sun, L. Liu, R. Lan, X. Luo, Lightweight image super-resolution via weighted multi-scale residual network, *IEEE/CAA Journal of Automatica Sinica*, Vol. 8, No. 7, pp. 1271-1280, July, 2021.

[5] J. Soh, N. Cho, Lightweight single image super-resolution with multi-scale spatial attention networks, *IEEE Access*, Vol. 8, pp. 35383-35391, February, 2020.

[6] K. He, X. Zhang, S. Ren, J. Sun, Spatial pyramid pooling in deep convolutional networks for visual recognition, *IEEE transactions on pattern analysis and machine intelligence*, Vol. 37, No. 9, pp. 1904-1916, September, 2015.

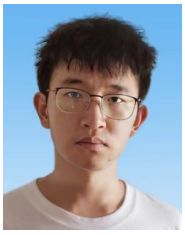
[7] Y. Guo, J. Y. Li, Z. H. Zhan, Efficient Hyperparameter Optimization for Convolution Neural Networks in Deep Learning: A Distributed Particle Swarm Optimization Approach, *Cybernetics and Systems*, Vol. 52, No. 1, pp. 36-57, 2021.

[8] R. Girshick, Fast R-CNN, *Proceedings of the IEEE International Conference on Computer Vision*, Santiago, Chile, 2015, pp. 1440-1448.
[9] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask R-CNN, *2017 IEEE international conference on computer vision*, Venice, Italy, 2017, pp. 2980-2988.
[10] Q. Zhu, P. Zhang, Z. Wang, X. Ye, A New Loss Function for CNN Classifier Based on Predefined Evenly-Distributed Class Centroids, *IEEE Access*, Vol. 8, pp. 10888-10895, December, 2019.
[11] Y. Guo, L. Du, G. Lyu, SAR Target Detection Based on Domain Adaptive Faster R-CNN with Small Training Data Size, *Remote Sensing*, Vol. 13, No. 21, Article No. 4202, November, 2021.
[12] J. Han, D. Choi, S. Park, S. Hong, Hyperparameter Optimization Using a Genetic Algorithm Considering Verification Time in a Convolutional Neural Network, *Journal of Electrical Engineering & Technology*, Vol. 15, No. 2, pp. 721-726, March, 2020.
[13] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A. C. Berg, SSD: Single Shot MultiBox Detector, in: B. Leibe, J. Matas, N. Sebe, M. Welling (Eds.), *European Conference on Computer Vision*, Springer, Cham, 2016, pp. 21-37.
[14] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You Only Look Once: Unified, Real-Time Object Detection, *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 779-788.
[15] Z. Wang, X. Zhang, J. Li, K. Luan, A YOLO-Based Target Detection Model for Offshore Unmanned Aerial Vehicle Data, *Sustainability*, Vol. 13, No. 23, Article No. 12980, December, 2021.
[16] A. Alshehri, Y. Bazi, N. Ammour, H. Almubarak, N. Alajlan, Deep attention neural network for multi-label classification in unmanned aerial vehicle imagery, *IEEE Access*, Vol. 7, pp. 119873-119880, August, 2019.
[17] T. Lin, P. DollarR, R. Girshick, K. He, B. Hariharan, S. Belongie, Feature pyramid networks for object detection, *2017 IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, 2017, pp. 936-944.
[18] F. Gao, B. Li, L. Chen, Z. Shang, X. Wei, C. He, A Softmax Classifier for High-precision Classification of Ultrasonic Similar Signals, *Ultrasonics*, Vol. 112, Article No. 106344, April, 2021.
[19] S. Zhou, C. Su, Efficient Convolutional Neural Network for Pest Recognition-ExquisiteNet, 2020 *IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE)*, Yunlin, Taiwan, 2020, pp. 216-219.

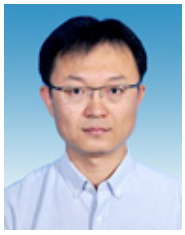
Biographies

Qi Chen was born in Tinjin, China. She received the B.E. degree in Automatic control from Harbin University of Science and Technology, China, in 1992, the master's degree from Tianjin University, China, in 2002, and the Ph.D. degree from Tianjin University, China, in 2008. Since 2008, she has been an Associate Professor with Information Engineering College of Tianjin University of Commerce. Her research interests include information processing of Industrial Internet of Things, production process control and optimization algorithm.

Xinyi Gao was born in Wuwei, China. She is currently pursuing the bachelor's degree in Automation with Information Engineering College of Tianjin University of Commerce, China. Her research interest includes automation, algorithm design and program writing.



Renjie Li was born in Hefei, China. He is currently pursuing the master's degree in Information and Communication Engineering in Tianjin university of commerce. He graduated from Wuyi University in 2017, majoring in communication engineering. His research interest includes intelligent detection and information processing.



Yong Zhang was born in Yanshan, China. He received the B.E. degree in Automation from the Liaoning Shihua University, China, in 2000, the M.E. degree in Computer application technology from Yanshan University, China, in 2005, and the Ph.D. degree in Inspection technology and automation from Tianjin University, China, in 2013. Since 2021, he has been a Professor with Information Engineering College of Tianjin University of Commerce. His research interests include wireless sensor networks, intelligent detection and information processing, source detection and localization.