ISSN: 0067-3064

Lightweight and Scale Adaptive Efficient backbone Network for Recognition

ChaoWang[1,3] , Kaijie Zhang[1*] , Xiaoyong Yu[1], Xianpeng Xiong[1], Aihua Zheng[2]
1.School of Informatics and Engineering, Suzhou University, Suzhou 234000, PR China;
2.Information Materials and Intelligent Sensing Laboratory of Anhui Province, Anhui University, HeFei, People's Republic of China;
3. Institute of Machine Learning and Systems Biology, School of Electronics and Information Engineering, Tongji University, ShangHai 201804, PR China.
*1976523155@qq.com

Abstract: Vehicle refinement recognition related technology research is widely used in the field of mine monitoring and management systems, road traffic command and control, etc. As researchers develop and implement the target recognition technology system based on deep learning algorithms, designing a target recognition algorithm with excellent performance is a research priority within the field of vehicle monitoring. In this paper, we propose an Efficient Net algorithm based recognition method for vehicle front-end and vehicle rear-end recognition to address the shortcomings of the current methods used for vehicle front-end and vehicle rear-end recognition, and verify the reliability of the algorithm using experiments. Algorithm systematically investigates model scaling, the backbone network makes extensive use of the MBConv structure to extract the feature maps, which cuts short the time required for model training, and the structure introduces the SE module to perform global averaging pooling operations in the channel dimension direction to enhance model performance, so that the network has the dual advantages of network model size and recognition accuracy at the same time. Based on the above findings, we improve the inverse residual module of the backbone feature extraction network EfficientNet by introducing the coordinate attention mechanism (CA) to average the spatial feature information in X-axis and Y-axis dimensions respectively, with the feature layer size and number of channels unchanged, and change the residual edge to shorten the input and output of high-dimensional channels to improve the accuracy of model feature extraction. Meanwhile, this paper introduces a depth-separable convolutional neural network and agent-normalized activation in the mobile flip-flop convolutional module to offset the two different dimensions of X-axis and Y-axis between each convolutional layer but the two main sources of non-normalization, so as to achieve the improvement of the target detection rate and accuracy.
Keywords: Lightweight Network; Scale Adaptive; Pattern Recognition; CNN

ISSN: 0067-3064

1 Introduction

By the end of 2021, the number of intelligent mining workings nationwide has reached 687, including 431 intelligent coal mining workings, accounting for 63%, and 256 intelligent digging workings, accounting for 37%; 26 types of coal mine robots have been realized in coal mine sites with different degrees of application [1]. According to the implementation plan of the three-year action plan for special improvement of coal mine safety, in 2022, we should strive to reach more than 1000 intelligent working faces for mining and excavation, reach 1 billion to 1.5 billion tons of production capacity in mines with intelligence, and build a number of intelligent mine models with fewer than 100 people [2]. Compared with the traditional mine model, smart mines rely on the application of intelligent and digital technologies and mechanical equipment to achieve digital manipulation and automated linkage between mining production systems [3]. Smart mine construction is an important cornerstone for sustainable development of resources and a fundamental way to solve the safe and standardized transportation of mineral resources [4]; in recent years, mining enterprises have been deepening and extending the connotation of intelligent mine transportation vehicle identification management around their own development reality, from digital mine [5] to smart mine [6] and sensing mine [7], all with information, digitalization and automation as the core [8]. At present, China's mining enterprises are facing the refinement of the identification of mining vehicles [9] with the type [10] and weight of the mined material as well as the detailed information of the vehicles intelligent, digital, networked big transformation [11]; at the same time, the State Administration of Safety takes the lead to accelerate the organization and implementation of relevant scientific research and research, transformation of results, promotion and application and technology demonstration projects; and with China's image perception, pattern recognition, Internet of things

In terms of technology, driverless vehicles in domestic open-pit coal mines have reached (146) hundreds; intelligent violation control systems have been practically applied in several coal mines [15]. With the continuous development of technology, in the process of mineral work, the efficiency of the dredger is getting faster and faster, which leads to the increase of mineral production, in order to make the timely delivery of mineral resources to ensure the regular operation of the dredging operation, the fine identification and monitoring management supervision of the mine car, a proven method is to use the mine monitoring system for the fine identification technology research of the mine car at the time of the entry and exit of the mine car [16] [17] [18], this refinement identification technology research to a certain extent greatly reduces the labor consumed by the traditional based on manual data collection, thus realizing the detailed data record

and statistics of the mine transportation vehicles under certain conditions. The pattern recognition application appeared initially due to the inadequacy and imprecise recognition of the algorithms involved in this technology, along with the rapid development of computers and the continuous optimization of computer hardware and software, the artificial intelligence image pattern recognition took a new wave and a variety of algorithms began to contribute to the mine vehicle management system [19] [20]. In our previous work we used a vehicle detection method based on YOLOv5 image recognition and processing technology to carry out systematic recognition selection work for vehicles entering and leaving the mine area, the algorithm overcomes the difficult problem of high road traffic flow under natural non-anthropogenic influence conditions that lead to inaccurate recognition of vehicles, as well as various disturbing factors such as blurred vehicles at night and missing parts of vehicles that are not fully displayed or recognized. The experimental results show that the algorithms are able to accurately identify and segment the vehicles according to their edge contours [16]. When the mine environment supervision system captures the picture of a vehicle, sometimes it cannot accurately give whether the captured is the front or the rear or the roof of the vehicle, the captured picture contains the front of one vehicle and the rear of another vehicle, in the case that the front and the rear of the vehicle cannot be distinguished, generally according to the conventional left-in-right-out principle or custom rules, distinguish the current need to identify the license plate number of the front or the license plate number of the rear of the vehicle However, the system sometimes cannot distinguish between the front and the rear of the car, resulting in more false alarms. In this condition, we propose a deep convolutional neural network-based mining vehicle front-end and rear-end recognition research method, which systematically filters out the front-end and rear-end pictures as well as the data that are difficult for people to distinguish the front-end and rear-end of the vehicle, using a layer of convolution plus a layer of pooling, with a total of 5 layers, and using ReLU as the activation function of the neurons in the middle hidden layer, followed by 2 fully connected layers and a prediction layer for the convolutional tail processing, then the softmax activation function is used to classify the head and tail of the vehicle, and finally the parameters are updated using the Adam optimizer [17]. The algorithm is applied to identify the front-end and rear-end recognition rate of mining vehicles in surveillance images is somewhat unsatisfactory. The work of this paper is to solve or partially solve this problem of poor recognition accuracy of the head and tail of mining vehicles, and subsequently proposes an intelligent recognition method of the head and tail of vehicles based on the Efficient Net algorithm, which is getting more and more attention from scholars as well as related industries according to its powerful recognition accuracy, and the refined recognition of the head and tail of vehicles by this algorithm will provide The refined recognition of the head and tail of the

vehicle by this algorithm will have important practical significance for us to further capture information for the recognition of license plate.

This paper improves the EfficientNet backbone feature extraction network on the basis of EfficientNet algorithm, including changing the SE module of the original network model to CA attention module, and solving the loss of low-dimensional feature information after the convolutional compression channel in the process of residual connection by the idea of flipping the inverted residual. Deeply separable convolution is introduced to solve the hardware limitation of memory access speed throughput, and the number of dimensions of activation is increased or decreased by the "expansion factor" around the spatial depth convolution. The proposed agent normalized activation solves the problem of batch size limitation introduced by EfficientNet for normalizing the output of convolution and matrix multiplication operations, and the accuracy of the experimental results is greatly improved by optimizing the network model, as shown in Table 5.

The structure of the rest of this paper is organized as follows: Section 1 goes through the introduction of intelligent extraction work in China and the discussion of the related image pattern recognition technology research designed in this scenario; Section 2 reviews the related work on deep convolutional neural networks and the model introduction of EfficientNet; Section 3 elaborates the theoretical basis of the algorithmic models involved in this paper, including Section 3 describes the theoretical basis of the algorithmic model covered in this paper, including a detailed introduction of the 9 phases of EfficientNet-B0 network structure, Model Scaling, MBConv structure, and SE module; Section 4 introduces the improved EfficientNet backbone feature extraction network; Section 5 verifies the effectiveness of the model through the analysis of experimental results; Section 6 summarizes the work of this paper and gives an outlook on the future research direction of image Section 6 concludes this paper and gives an outlook on the future research direction of pattern recognition.

2 Related Work

The pioneering work of CNN is LeNet-5 proposed by LeCun [21], and its real explosive stage is when AlexNet achieved the championship of classification task in ImageNet competition in 2012, and the classification accuracy is far more than the classification results achieved by using traditional methods, and the reason for the success of this model is mainly the massive labeled training data i.e., the large scale labeled data set provided by Feifei Li's team The success of this model is mainly due to the large amount of labeled training data, i.e., ImageNet, a large-scale labeled dataset provided by Fei-Fei Li's team, and the support of computer hardware, with the advent of GPUs providing powerful support for complex computations [22]. Further, there are improvements in algorithms, including network structure deepening, data augmentation (data expansion),

ReLU, Dropout, etc. The network ZF Net, designed by Matthew Zeiler and Rob Fergus at New York University in 2013, the article reveals what exactly the layers of the neural network are doing and what role they play by using visualization techniques [23]. The subsequent deep convolutional neural network VGGNet, developed by the Oxford University Computer Vision Group (Visual Geometry Group) together with researchers at Google DeepMind, used 33 convolutional kernels and 22 pooling kernels [24] to improve performance by continuously deepening the network structure, and InceptionNet, which appeared in the same year In the same year, InceptionNet emerged to control the amount of computation and number of parameters while obtaining very good classification performance with a top-5 error rate of 6.67% [25]. 2014 GoogLeNet reduced the error rate to 6.656%, with the prominent feature of greatly increasing the depth of the convolutional neural network, replacing the last fully connected layer with a 1x1 convolutional layer, greatly accelerating the training rate. resNet ResNet was introduced in the residual network structure (residual network) was proposed in 2015 [26], and won first place in the ImageNet competition classification task, after which many methods are built on the basis of ResNet50 or ResNet101, detection, segmentation, recognition and other fields have been used. ResNet. mobileNet, proposed by Google in 2017, is a lightweight CNN neural network focused on on mobile and embedded devices, and quickly derived v1, v2, and v3 versions, which greatly reduce the model parameters and the amount of operations with a small reduction in accuracy compared to traditional CNN networks [27]. With the continuous optimization of deep convolutional neural networks people slowly found that improving the performance of neural networks does not only lie in the number of stacked layers, but more important points are: 1. the network should be trainable and converge; 2. the number of parameters should be relatively small to facilitate training and also increase the speed; 3. the structure of innovative neural networks to learn more important things. As with other neural networks, one of the key issues in designing a cnn network is model scaling, such as deciding how to increase the size of the model to provide better accuracy. This is a lengthy process that requires manual hits and trials until a sufficiently accurate model is produced that satisfies the resource constraints. This process is resource- and time-consuming and often produces models with suboptimal accuracy and efficiency.

Model efficiency is becoming increasingly important in computer vision, and EfficientDet: Scalable and Efficient Object Detection systematically investigates the design choices of various neural network structures for target detection and proposes several key optimization methods to improve efficiency [28]. First, we propose a weighted bi-directional feature pyramid network (BiFPN) that allows simple and fast multi-scale feature fusion; second, we propose a composite scale expansion method that scales the resolution, depth, and width of all backbone,

feature, and prediction networks in a uniform manner. Based on these optimizations, we developed a new family of object detectors called EfficientDet. How to perform efficient multi-scale feature fusion (EFMF) is also one of the research directions that we are eager to focus on [29], mentioning that multi-scale fusion, when fusing different input features Most of the previous studies (FPN and some improvement works on FPN) just add up the features without distinction; however, since these different input features have different resolutions, we observe that their contributions to the fused output features are often unequal, and to solve this problem, the authors propose a simple and efficient weighted (similar to attention) bi-directional feature pyramid network (BiFPN), which introduces learnable weights to learn the importance of different input features, while iteratively applying top-down and bottom-up multi-scale feature fusion. Based on EfficientNet, the authors propose model scaling for networks such as backbone of detectors and propose a new family of detectors called EfficientDet in combination with the proposed BiFPN. The proposed detectors in the paper mainly follow the one-stage design idea and can achieve higher efficiency and accuracy by optimizing the network structure. EfficientNet originated from Google Brain's paper EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks [30]. In the paper, model scaling is systematically investigated and it is found that finely balancing the depth, width and respectively rate of the network can achieve more performance. Based on these findings, we propose a new scaling method that uses a simple and efficient compound coefficient to uniformly scale the depth, width, and separateness dimensions. The authors used neural structure search to build an efficient network structure efficientnet - b0. It achieves 77.3% accuracy on ImageNet with only 5.3M parameters and 0.39B latency. (Resnet-50 provided 26M parameters and 4.1B FLOPS 76% accuracy). The main component of this network is the MBConv with additional compressed excitation optimization. the MBConv is similar to the inverted residual blocks used in MobileNet v2 [31]. These form shortcut connections between the beginning and end of the convolutional blocks. The input activation mapping is first extended using 1x1 convolution to increase the depth of the feature mapping. This is followed by a 3x3 Depth-wise and Point-wise convolution that reduces the number of channels in the output feature map. The structure of fast-connected narrow layers with jump connections between wider layers helps to reduce the overall number of required operations as well as the model size. It has higher accuracy and efficiency compared to previous convolutional neural networks. The most important innovation of this paper is Model compound Scaling, which improves the effectiveness of the network by uniformly scaling the resolution, depth and width of all backbone, feature and prediction networks in a hybrid manner.The effectiveness of EfficientNet is significantly improved compared to the previous neural network accuracy results, making it an extremely powerful

network in today's era algorithm.

3 Core elements of efficient-net algorithm

3.1 Introduction to efficient-net networks

The paper EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks focuses on the NAS (Neural Architecture Search) technique [32] to search the network for the image input resolution r, the depth of the network depth and the rationalization of the three parameters width of the channel [55]. Usually, scaling up convolutional neural networks is a widespread approach in order to obtain better accuracy. For example, ResNet can be scaled up from ResNet-18 to ResNet-200 by using more layers of convolutional neural networks; GPipe obtained 84.3% top-1 accuracy on the ImageNet dataset by scaling up the baseline model four times, however the process of scaling up CNNs has never been well understood and used, before the algorithm was produced In some related papers, the network performance was rationalized by changing one of the three parameters: the image input resolution r, the depth depth of the network, and the width of the channel [33][34][35], but arbitrary scaling requires tedious manual tuning of the parameters and may yield a suboptimal accuracy and efficiency. This paper is to explore the substantive impact of these three parameters simultaneously. In the paper, it is mentioned that EfficientNet-B7 achieves the highest accuracy of 84.3% on Imagenet top-1 for the year, with only 1/8.4 of the number of parameters and 6.1 times faster inference compared to the previous highest accuracy GPipe. parameters to achieve state-of-the-art accuracy - CIFAR-100 (91.7%), Flowers (98.8%) [36][37]. Although the algorithm appears to be fast and lightweight in terms of overall data, the algorithm still suffers from some shortcomings in practice, such as a large runtime memory consumption. The structure of EfficientNet-B0 obtained by searching with Neural Architecture Search technique, the whole network framework consists of a series of stage stages, which indicates the operation of the corresponding stage, and the number of repetitions in the stage, the network framework of EfficientNet-B0 (B1- B7 is the modification of Resolution, Channels and Layers on the basis of B0) [38], it can be seen that the network is divided into a total of 9 Stages, the first Stage is a convolutional kernel size of 3x3 step 2 ordinary convolutional layer (containing BN and activation function Swish), Stage2 to Stage8 are in Repeatedly stacking MBConv structures (the Layers in the last column indicate how many times the Stage repeats the MBConv structure), while Stage9 consists of a normal 1x1 convolutional layer (containing BN and activation function Swish) an average pooling layer and a fully connected layer. Each MBConv in the table is followed by a number 1 or 6, where 1 or 6 is the multiplicity factor n. The first 1x1 convolutional layer in MBConv expands the channels of the input feature matrix by a factor of n, where k3x3 or k5x5 indicates the size of the convolutional kernel

used by Depthwise Conv in MBConv. Channels of the output feature matrix after passing this Stage, as shown in Table 1.

Table 1. EfficientNet-B0 base network

Stage	Operator	Resolution	#Channels	#Layers
1	Conv3x3	224 × 224	32	1
2	MBConv1, k3x3	112 × 112	16	1
3	MBConv6, k3x3	112 × 112	24	2
4	MBConv6, k5x5	56 × 56	40	2
5	MBConv6, k3x3	28 × 28	80	3
6	MBConv6, k5x5	14 × 14	112	3
7	MBConv6, k5x5	14 × 14	192	4
8	MBConv6, k3x3	7 × 7	320	1
9	Conv1x1 & Pooling & FC	7 × 7	1280	1

3.2 EfficientNet-B0 network structure

The EfficientNet-B0 network structure has 9 stages, each of which is convolved first, then batch normalization layer, and then computed using Swish activation function, etc. Overall, it consists of 16 moving flipping bottleneck convolution modules, 2 convolutional layers, 1 global average pooling layer and 1 classification layer. As shown in the following figure, where a picture is inserted for description (different colors represent different stages in Figure 1 below) [39].

In the first stage, the following operations are performed sequentially on the input 224x224x3 image to obtain the result of the first stage. 1. Convolution (convolution kernel is 32 kernel 3×3×3, step size is 2×2, padding is "same" i.e. the output width and height are reduced by half), the output of this convolution operation is a feature map with dimension (112× 112×32) feature map. Since this layer does not contain bias terms, the total number of parameters to be trained and learned in this layer is 864 (32x3x3x3). 2. Batch Normalization (BN), the input of this layer is a feature map of (112×112×32), so the total number of parameters in this layer is 128 (32x4), of which 64 parameters need to be trained and learned. 3. Finally, after the Swish activation function ends the first stage, the total number of parameters in the first stage is 128 + 864 = 992, and the number of parameters that need to be trained and learned is 928.

In the second stage, the 112x112x32 feature map outputted in the previous stage is subjected to moving flip bottleneck convolution (expansion ratio of 1, depth convolution kernel size of 3x3, kernel step size of 1x1, containing compression and excitation operations, no connection deactivation and connection skip), and the results of the second stage are output. Since the dilation ratio is 1, we skip the point-by-point convolution at the beginning and perform the deep convolution directly (the convolution kernel is 32 kernels of 3×3×3, the step size is 1×1, and the padding is "same", i.e., the width and height of the output are unchanged). The

output of the deep convolution is a feature map of dimension (112×112×32). Since this layer does not contain bias terms, the total number of parameters to be trained and learned in this layer is 288 (32x3x3x1). 2. Batch Normalization (BN), the input of this layer is a feature map of (112×112×32), so the total number of parameters in this layer is 128 (32x4), of

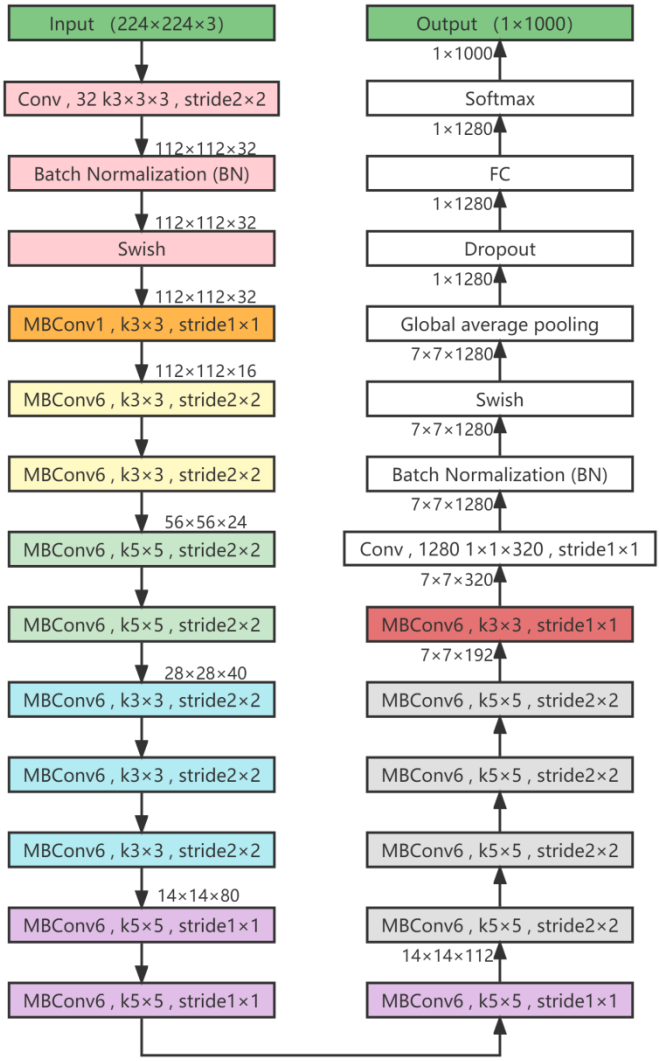

Fig. 1 Structure of EfficientNet-B0 model

which 64 parameters need to be trained and learned. 3. The Swish activation function leads to the global average pooling layer, which performs global average pooling in the channel dimension direction and outputs a (1x1x32) feature map. 4. Convolution (the first convolution in the compression and excitation module, the convolution kernel is 8 kernels 1x1x32, the step size is 1×1, and the padding is "same" i.e., the output is constant in width and height), the output of this convolution operation is a feature map of dimension (1×1×8). Since this layer

contains bias terms, the total number of parameters to be trained and learned in this layer is 264 (8x1x1x32+8). 5. The second convolution in the compression and excitation module is the second convolution with 32 kernels 1x1x8 and a step size of 1×1, filled with "same" i.e. the output The output of this convolution operation is a feature map with dimension (1×1×32). Since this layer contains bias terms, the total number of parameters to be trained in this layer is 288 (32x1x1x8+32).6. Multiply the Sigmoid activation function with the result of step 3 to obtain a 112x112x32 feature map.7. Convolution point by point (the convolution kernel is 16 kernels 1×1×32, the step size is 1×1, and the padding is " same" i.e., the output width and height are unchanged) The output of this convolution operation is a feature map of dimension (112×112×16). Because this layer does not contain bias terms, the total number of parameters to be trained and learned in this layer is 512 (16x1x1x32).8. Batch Normalization (BN), the input of this layer is a feature map of (112×112×16), so the total number of parameters in this layer is 64 (16x4), of which 32 parameters need to be trained and learned. The total number of parameters in the second stage is 288+128+264+288+512+64=1544, and 1448 parameters need to be trained and learned.

The third stage performs two moving flip bottleneck convolutions on the 112x112x16 feature map output from the previous stage, the first (expansion ratio of 6, depth convolution kernel size of 3x3, kernel step size of 2x2, containing compression and excitation operations, no connection deactivation kernel connection jumpover), the second (expansion ratio of 6, depth convolution kernel size of 3x3, kernel step size of 1x1, containing compression and excitation 1. point-by-point convolution with a dilation ratio of 6 (convolution kernel of 96 kernels 1×1×16, step size 1×1, padding "same", i.e., the output width and height are unchanged) The output of this convolution operation is a feature map of dimension (112×112×96) The output of this convolution operation is a feature map of dimension (112×112×96). Because this layer does not contain bias terms, the total number of parameters to be trained and learned is 1536 (96x1x1x16). 2. Batch Normalization (BN), the input of this layer is a feature map of (112×112×96), so the total number of parameters contained in this layer is 384 (96x4), of which 192 parameters need to be trained and learned. 3. Through the Swish activation function into the deep convolution (convolution kernel is 6 kernel 3 × 3 × 16, step size is 2 × 2, the fill is "same" that is, the output width and height reduced by half). The output of the deep convolution is a feature map of dimension (56×56×96). Since this layer does not contain bias terms, the total number of parameters to be trained and learned in this layer is 864 (96x3x3x1). 4. Swish activation function into the global average pooling layer, the layer in the direction of the channel dimension global average pooling, the output is (1x1x96) feature map. 6. convolution (the first convolution in the compression and excitation module, the convolution kernel is 4 cores 1x1x96, the step size is 1 × 1,

the fill is "same" i.e., the output is constant in width and height), the output of this convolution operation is a feature map of dimension (1×1×4). Since this layer contains bias terms, the total number of parameters to be trained in this layer is 388 (4x1x1x96+4).7. The second convolution in the compression and excitation module is connected to the convolution by the Swish activation function (with a 96-kernel 1x1x4 convolution kernel and a step size of 1×1, filled with "same", i.e., the output of The output of this convolution operation is a feature map with dimension (1×1×96). Since this layer contains bias terms, the total number of parameters to be trained and learned in this layer is 480 (96x1x1x4+96). 8. Multiply the result of step 5 with the Sigmoid activation function to obtain a 56x56x96 feature map. 9. Convolution point by point (the convolution kernel is 24 kernels 1×1×96, the step size is 1×1, the padding is "same "The output of this convolution operation is a feature map of dimension (56×56×24). Since this layer does not contain bias terms, the total number of parameters to be trained and learned in this layer is 2304 (24x1x1x96).10. Batch Normalization (BN), the input of this layer is a feature map of (56×56×24), so the total number of parameters contained in this layer is 96 (24x4), of which 48 parameters need to be trained and learned. The second moving flip bottleneck convolution in this stage (expansion ratio of 6, depth convolution kernel size of 3x3, kernel step size of 1x1, contains compression and excitation operations with connection deactivation and connection skip), where the ending connection deactivation and connection skip do not contain parameters. Except for the change in the step length of the depth convolution, the rest of the operation is the same as the first moving flip bottleneck convolution, and the output is (56x56x24) so the total sum of parameters of the second moving flip bottleneck convolution is 3456+576+1296+576+870+1008+3456+96=11334, of which 10701 parameters need to be trained, and the total of the third order 17770 parameters, and 16705 parameters need to be trained and learned.

In the fourth stage, the 56x56x24 feature map output in the previous stage is moved twice to flip the bottleneck convolution, the first one (expansion ratio of 6, depth convolution kernel size of 5x5, kernel step size of 2x2, containing compression and excitation operations, no connection deactivation kernel connection jumpover), the second one (expansion ratio of 6, depth convolution kernel size of 5x5, kernel step size of 1x1, containing compression and excitation operations, with connection deactivation and connection skip), the output is a 28x28x40 feature map. The total number of parameters is 48336 and 46640 parameters need to be trained and learned.

In the fifth stage, the 28x28x40 feature map output from the previous stage is convolved with three moving flip bottlenecks, the first one (expansion ratio of 6, depth convolution kernel size of 3x3, kernel step size of 2x2, including compression and excitation operations, no connection deactivation kernel

connection skip), the second one (expansion ratio of 6, depth convolution kernel size of 3x3, kernel step size of 1x1, including compression and excitation with connected deactivation kernels connected to hop-over), and the third (dilation ratio of 6, depth convolution kernel size of 3x3, kernel step size of 1x1, including compression and excitation operations, connected deactivation kernels connected to hop-over), the output is a 14x14x80 feature map. The total number of parameters is 248,210 and 242,930 parameters need to be trained and learned.

In the sixth stage, the 14x14x80 feature map output from the previous stage is convolved with three moving flip bottlenecks, the first one (expansion ratio of 6, depth convolution kernel size of 5x5, kernel step size of 1x1, including compression and excitation operations, no connection deactivation kernel connection skip), the second one (expansion ratio of 6, depth convolution kernel size of 5x5, kernel step size of 1x1, including compression and excitation with connected deactivation kernels), and the third (dilation ratio of 6, deep convolution kernel size of 5x5, kernel step size of 1x1, containing compression and excitation operations, with connected deactivation kernels connected to skip), the output is a 14x14x112 feature map. The total number of parameters is 551116, and 543148 parameters need to be trained and learned.

In the seventh stage, the 14x14x112 feature map output from the previous stage is convolved with four moving flip bottlenecks, the first one (expansion ratio of 6, depth convolution kernel size of 5x5, kernel step size of 2x2, including compression and excitation operations, no connection deactivation kernel connection skip), the second one (expansion ratio of 6, depth convolution kernel size of 5x5, kernel step size of 1x1, including compression and excitation with connected deactivation kernel connection hopping), the third (expansion ratio of 6, depth convolution kernel size of 5x5, kernel step size of 1x1, containing compression and excitation operations, with connected deactivation kernel connection hopping), and the fourth (expansion ratio of 6, depth convolution kernel size of 5x5, kernel step size of 1x1, containing compression and excitation operations, with connected deactivation kernel connection hopping), the output is a 7x7x192 feature map. The total number of parameters is 2044396, and 2026348 parameters need to be trained and learned.

In the eighth stage, the 7x7x192 feature map output from the previous stage is convolved with a moving flip bottleneck (expansion ratio of 6, depth convolution kernel size of 3x3, kernel step size of 1x1, including compression and excitation operations, no connection deactivation kernel connection jumpover) and the output is a 7x7x320 feature map. In total, there are 722,480 parameters and 717,232 parameters to be trained and learned.

In the ninth stage, the following operations are performed sequentially on the input 7x7x320 image to obtain the final result of the model. 1. Convolution (convolution kernel is 1280 kernel 1×1×320, step size is 1×1, padding is "same"

i.e. the output width and height are constant), the output of this convolution operation is a feature map with dimension (7×7× 1280) feature map. Because this layer does not contain bias terms, the total number of parameters to be trained and learned in this layer is 409600 (1280x1x1x320). 2. Batch Normalization (BN), the input of this layer is a feature map of (7×7×1280), so the total number of parameters in this layer is 5120 (1280x4), of which 2560 parameters need to be trained and learned. 3. The global average pooling layer is accessed by the Swish activation function, which performs global average pooling in the channel dimension direction and outputs a feature map of (1x1x1280). 4. The random deactivation dropout function is used to connect the fully connected layer, which has 1000 neurons. Since this layer contains bias terms, the total number of parameters is 1281000 (1000x1280+1000). 5. Finally, the Softmax activation function is used to output the classification results, with a total of 1695720 parameters in the ninth stage and 1693160 parameters to be trained and learned.

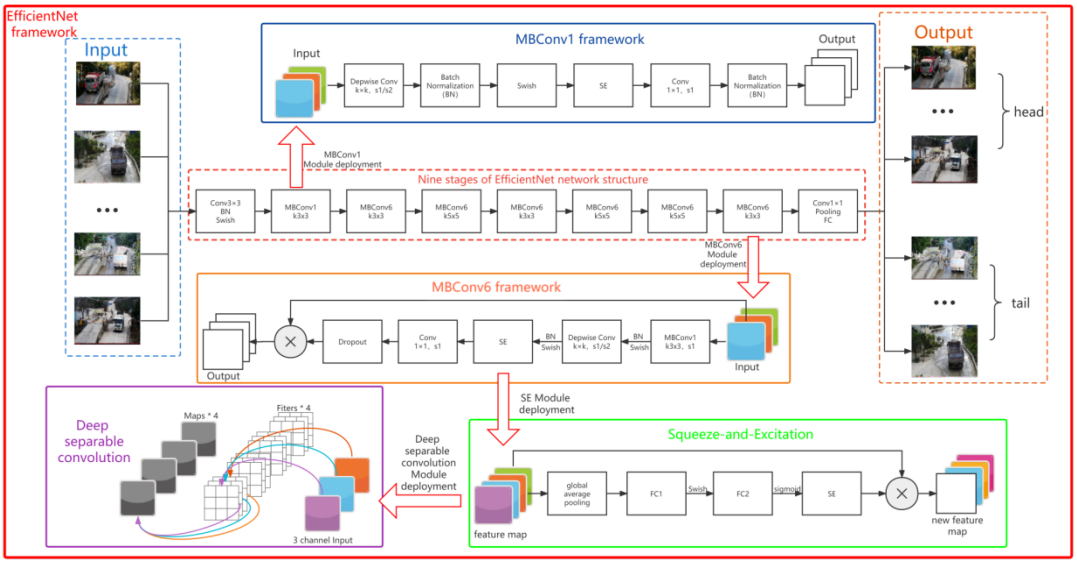

Fig. 2 The structure of backbone framework. The network architecture contains of the MBConv module, SE module, and depth-separable convolution module

The algorithm of Efficient Net has the ability to quantize and adjust the complex network, through the comprehensive adjustment of network depth, width and resolution of the input image, to obtain the optimal network parameters for specific needs, so that the network has the advantages of both network size and recognition accuracy. The overall workflow is to transform the image into the input dimension required by the MBConv module after the first Conv3x3 layer; the image goes through a series of MBConv modules to extract the feature map [40]; the parameters of each MBConv module are finely tuned to suit the current usage environment; the combined scale optimization approach allows the network to obtain The combined scale optimization approach allows the network to obtain

a better perceptual field; the feature map adaptive connection method based on Fully-Convlutional-Neural-Network is utilized. Using Conv1x1 network will be able to adapt to various feature maps of different sizes and unify them into the dimensions we need; finally, the output feature maps will be used to complete the classification and recognition detection of images.

3.3 Model Scaling

The mobile flipping bottleneck convolution is also obtained by searching the neural network architecture [41], and the structure of this module is similar to depthwise separable convolution [42], which first performs a 1x1 point-by-point convolution of the input and changes the output channel dimension according to the expansion ratio (If the expansion ratio is 3, the channel dimension will be increased by a factor of 3. However, if the expansion ratio is 1, the 1x1 point-by-point convolution and its subsequent batch normalization and activation functions are omitted). Then a depthwise convolution of kxk is performed. If a compression and excitation operation is to be introduced, this operation is performed after the depthwise convolution. The original channel dimension is restored with a 1x1 point-by-point convolution ending. Finally, a drop connect and a skip connection of the inputs are performed, a practice derived from the paper Deep networks with stochastic depth [43], which gives the model a random depth, clipping the time required for model training and improving the model performance (note that in EfficientNets, connection deactivation and input jump-over connections are performed only when the same moving flipping bottleneck convolution is repeated and also change the depth convolution step in it to 1), connection deactivation is a random deactivation-like (dropout) operation and incorporates constant jump-over at the beginning and end of the module. Note that each convolution operation in this module is followed by a batch normalization, and the activation function uses the Swish activation function.

The compression and excitation operations in the moving flipping bottleneck convolution module, hereafter referred to as the SE module, is an attention-based feature map operation. The SE module first performs a compression operation on the feature map and global average pooling in the channel dimension direction to obtain global features in the channel dimension direction of the feature map. Then the global features are excited and convolved with a 1x1 convolution of the global feature dimension (C) using the activation ratio (R, which is a floating point number) (the original method uses a fully connected layer) to learn the relationship between the channels, and then the weights of the different channels are obtained by the sigmoid activation function, and finally the final features are multiplied by the original feature map. Essentially, the SE module does (attention) attention or (gate control) gating operations on the channel dimensions. This attention mechanism allows the model to focus more on the most informative

channel features and suppress those channel features that are not important. Another point is that the SE module is generic, which means that it can be embedded in other existing network architectures. The structure is shown in Figure 3. Note that in the moving flip bottleneck convolution module, it is the input channel dimension of the moving flip bottleneck convolution module that is multiplied with the activation ratio, not the output channel dimension after deep convolution in the module.

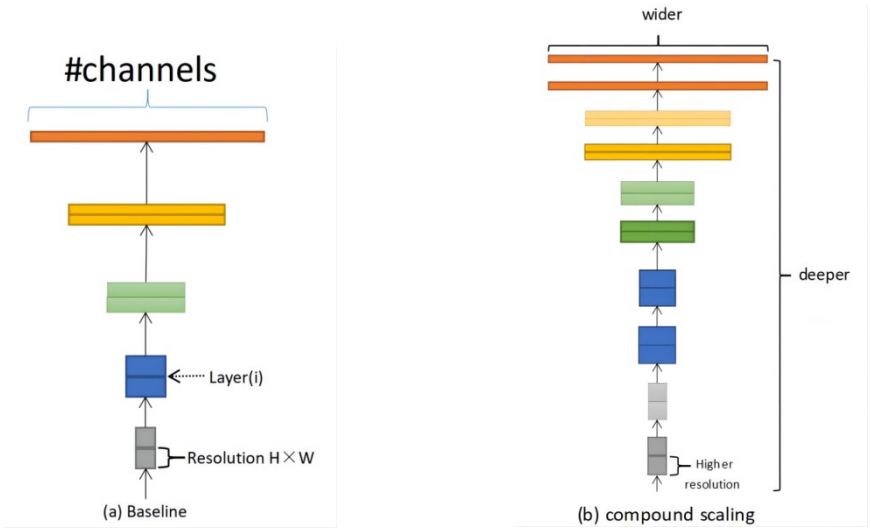

Fig. 3 Model Scaling

Increasing the depth of the network can get richer and more complex features and can be well applied to other tasks, but too deep a network will face the problem of gradient loss and training difficulties. Increasing the width of the network gives higher fine-grained features and is easier to train, but it is often difficult to learn deeper features for networks with large widths and shallow depths. Increasing the image resolution of the input network can potentially lead to higher fine-grained feature templates, but the accuracy gain is reduced for very high input resolutions, and large resolution images increase the computational effort. In order to explore the impact of the three factors d,r,w on the final accuracy, as shown in Equation (1), d,r,w are added to the equation, and we can obtain the abstracted optimization problem under the specified resource constraints, where s.t. represents the constraints, d is used to scale the depth $\widehat{L_i}$, r is used to scale the resolution i.e. the impact $\widehat{H_i}$ and $\widehat{W_i}$, w is the channel used to scale the feature matrix i.e. $\widehat{C_i}$, target_memory is the memory limit, target_flops is the FLOPs limit, $\bigodot_{i-1\ldots s}$ represents the concatenated multiplication operation, F_i represents an operation (such as the Operator in the above table EfficientNet-B0 base network), then indicates that the $F_i^{L_i}$ operation F_i is repeatedly executed L_i times in the Stagei, X represents the input Stagei feature matrix (input tensor). (H_i, W_i, C_i) denotes the

height, width, and Channels (shape) of X.

$$\max_{d,w,r} \quad \text{Accuracy}(N(d,w,r)) \tag{1}$$

$$\text{s.t.} \quad N(d,w,r) = \bigodot_{i-1\ldots s} \widehat{F_i^{d*\widehat{L_i}}}(X_{(r*\widehat{H_i},r*\widehat{W_i},r*\widehat{C_i},)}) \tag{2}$$

$$\text{Memory}(N) \leq \text{target\_memory} \tag{3}$$

$$\text{FLOPs}(N) \leq \text{target\_flops} \tag{4}$$

$$\text{depth}: d = \alpha^{\Phi} \tag{5}$$

$$\text{width}: w = \beta^{\Phi} \tag{6}$$

$$\text{resolution}: r = \gamma^{\Phi} \tag{7}$$

$$\text{s.t.} \quad \alpha * \beta^2 * \gamma^2 \approx 2 \tag{8}$$

$$\alpha \geq 1, \quad \beta \geq 1, \quad \gamma \geq 1 \tag{9}$$

The relationship between FLOPs (theoretical computation) and depth is that when depth is doubled, FLOPs are doubled, and FLOPs and width are related in that when width is doubled (i.e., channal is doubled), FLOPs are quadrupled, because the FLOPs of a convolutional layer are approximately equal to $\text{feature}_w \times \text{feature}_h \times \text{feature}_c \times \text{kernel}_w \times \text{kernel}_h \times \text{kernel}_{\text{number}}$ (Assuming constant height and width of the input and output feature matrices) , When the width is doubled, both the channels of the input feature matrix (feature_c) and the number of channels or convolution kernels of the output feature matrix ($\text{kernel}_{\text{number}}$) are doubled , So the FLOPs will be quadrupled. The relationship between FLOPs and resolution is that when resolution is doubled, FLOPs will also be doubled 4 times, similar to the above because the width feature_w of the feature matrix and the height feature_h of the feature matrix will both be doubled.

So the total FLOPs multiplier can be approximated by $(\alpha * \beta^2 * \gamma^2)^{\Phi}$. When restricted, the FLOPs are rather increased by a factor of 2^{Φ} for any one Φ. Next the authors used NAS on the benchmark network EfficientNetB-0 to search for the three parameters α,β,γ. [30] Firstly, fixing Φ=1 and searching based on equation (1) given above, the authors found that for EfficientNet-B0 the optimal parameters are α= 1.2 ,β= 1.1 ,γ= 1.15. Then fixing α= 1.2 ,β= 1.1 ,γ= 1.15, and using different Φ on the base of EfficientNetB0 to obtain respectively EfficientNetB1 to EfficientNetB7, as shown in Table (2). It is important to note that the α,β,γ searched for different benchmark networks are not the same. It should also be noted that in the original paper, the authors also stated that if the search for α,β,γ is done directly on the large model, better results may be obtained, but the search cost is too large in the larger model, i.e., the smaller EfficientNetB0 model is used for the search.

Table 2. Details of the data obtained using different Φ on the basis of

EfficientNetB0

Model	input_size	width_coefficient	depth_coefficient	drop_connect_rate	dropout_rate
EfficientNetB0	224x224	1.0	1.0	0.2	0.2
EfficientNetB1	240x240	1.0	1.1	0.2	0.2
EfficientNetB2	260x260	1.1	1.2	0.2	0.3
EfficientNetB3	300x300	1.2	1.4	0.2	0.3
EfficientNetB4	380x380	1.4	1.8	0.2	0.4
EfficientNetB5	456x456	1.6	2.2	0.2	0.4
EfficientNetB6	528x528	1.8	2.6	0.2	0.5
EfficientNetB7	600x600	2.0	3.1	0.2	0.5

input_size represents the image size of the input network when training the network, width_coefficient represents the multiplication factor in the channel dimension, for example, the number of convolutional kernels used in the 3x3 convolutional layer of Stage1 in EfficientNetB0 is 32, then in B6 it is 32×1.8=57.6 and then rounded to the nearest integer multiple of 8, that is 56. depth_coefficient represents the multiplicity factor in depth dimension (only for Stage2 to Stage8), for example, if $\widehat{L}_i$=4 for Stage7 in EfficientNetB0, then it is 4×2.6=10.4 in B6, and then rounded up to 11. drop_connect_rate is the drop_rate used by the dropout layer in the MBConv structure. In the official keras module implementation the drop_rate of the MBConv structure is incremented from 0 to drop_connect_rate [44] (see the official source code for the specific implementation, note that in the source code (Note that the Dropout layer is only available in the source code when using shortcut). It should also be noted that the Dropout layer here is Stochastic Depth, i.e., it randomly drops the main branch of the entire block (only the shortcut branch remains, which is equivalent to skipping the block directly) and can be interpreted as reducing the depth of the network. The dropout_rate is the dropout_rate of the dropout layer before the last fully connected layer (between Pooling and FC in stage9).

3.4 MBConv Structure

The MBConv has great similarity to the Inverted Residual Block in the MobileNetV3 network, but there are also slight differences [45].The Swish

activation function is used in all the MBConv in EfficientNet, and MobileNetV3 is the one that incorporates in each MBConv the SE (Squeeze-and-Excitation) module in each MBConv [46].The MBConv structure is shown in Figure 4.

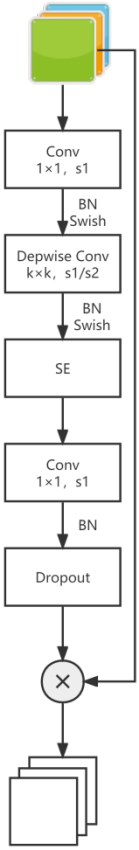

Fig. 4 MBConv model structure

The MBConv structure mainly consists of a 1x1 ordinary convolution (ascending role, containing BN and Swish), a kxk Depthwise Convolution (containing BN and Swish), the exact value of k can be seen in the network framework of EfficientNet-B0 mainly in the cases of 3x3 and 5x5, an SE module, a 1x1 ordinary convolution (dimensionality reduction role, containing BN), and a Droupout layer composition. The first dimensionalized 1x1 convolution layer has n times the number of convolution kernels of the input feature matrix channel, where $n \in \{1,6\}$. When n=1, the first upscaled 1x1 convolutional layer is not needed, i.e., the MBConv structure in Stage2 does not have the first upscaled 1x1 convolutional layer (similar to the MobileNetV3 network). shortcut connections exist only when the input MBConv structure has the same feature matrix as the output feature matrix shape (the code can be passed The SE module is shown below and consists of a global average pooling and two fully connected layers. The number of nodes in the first fully-connected layer is one-fourth of the input MBConv feature matrix channels, and the Swish activation function is used. The number of nodes in the

second fully-connected layer is equal to the depthwise Conv layer's output feature matrix channels, and uses the Sigmoid activation function. dropout_rate in the dropout layer corresponds to drop_connect_rate in tensorflow's keras source code, and is only available in the source implementation when Dropout layer is only available when using shortcut.

3.5 SE module

Squeeze-and-Excitation (SE) This module is proposed mainly considering the interdependence between model channels [49].Squeeze (compression) obtains the global compressed feature volume of the current Feature Map by performing Global Average Pooling on the Feature Map layer. Excitation (excitation) obtains the weights of each channel in the Feature Map through a two-layer fully connected bottleneck structure, and uses the weighted Feature Map as the input to the next layer of the network. In order to solve the loss problem caused by the different importance occupied by different channels of the feature map in the convolutional pooling process. In the traditional convolutional pooling process, each channel of the feature map is equally important by default, while in the actual problem, the importance of different channels varies. the Squeeze-and-Excitation module acts on the EfficientNet model structure and its extended structure as shown in Figure 5.

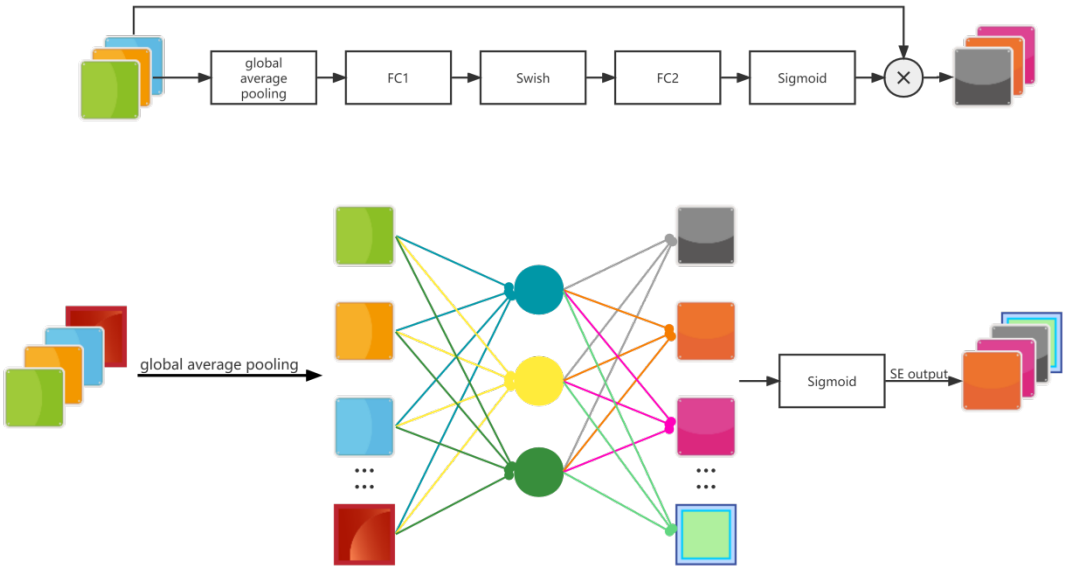

Fig. 5 Squeeze-and-Excitation module and its extended structure

The SE module first performs the Squeeze operation on the feature map obtained by convolution to get the global features at the channel level, and then performs the Excitation operation on the global features to learn the relationship between each channel and also get the weights of different channels, and finally multiplies the original feature map to get the final features. Essentially, the SE module is doing the attention or the gating operation on the channel dimension. This

attention mechanism allows the model to pay more attention to the most informative channel features and suppress those unimportant channel features. Another point is that the SE module is generic, which means that it can be embedded into existing network architectures. Since convolution only operates in a local space, it is difficult to obtain enough information to extract the relationship between channels, which is more serious for the front layers in the network, because the perceptual field is smaller. In order to, SENet proposes the Squeeze operation, which encodes the entire spatial features on a channel as a global feature, using global average pooling to achieve (in principle, a more sophisticated aggregation strategy can also be used).

$$Z_c = F_{sq}(u_c) = \frac{1}{H * W}\sum_{i=1}^{H}\sum_{j=1}^{W} u_c(i,j), z \in R^C \quad (10)$$

After the Sequeeze operation has obtained the global description of the features, we next need another operation to capture the relationships between channels. This operation needs to satisfy two criteria: firstly, it should be flexible, it should be able to learn the nonlinear relationships between the individual channels; the second point is that the learned relationships are not mutually exclusive, because here multi-channel features are allowed instead of the one-hot form. Based on this, the gating mechanism in sigmoid form is used here.

$$s = F_{ex}(z, W) = \sigma(g(z, W)) = \sigma(W_2 \mathrm{ReLU}(W_1 z)) \quad (11)$$

Where $W_1 \in R^{\frac{C}{r}*c}, W_2 \in R^{C*\frac{C}{r}}$. In order to reduce the model complexity and improve the generalization ability, a bottleneck structure with two fully connected layers is used here, where the first FC layer plays the role of dimensionality reduction, and the coefficient of dimensionality reduction is r as a hyperparameter, and then ReLU activation is used. The final FC layer restores the original dimensionality. Finally, the learned activation values (sigmoid activation, values 0 to 1) of each channel are multiplied by the original features.

$$\tilde{x}c = \mathrm{Fscale}(u_c, s_c) = s_c * u_c \quad (12)$$

4 Improving EfficientNet target detection model

4.1 Improving the EfficientNet backbone feature extraction network

The inverse residual module in the EfficientNet backbone feature extraction network can effectively reuse the data features in the data stream.The improved inverse residual module in MobileNeXt improves the accuracy of the model feature extraction by connecting the high-dimensional features with short residuals compared to the inverse residual module in EfficientNet. Q Hou. et al [52] proposed a Coordinate Attention (CA) mechanism, which averages pooling in the horizontal x-direction and vertical y-direction, encodes spatial information

on the transforms, and weighted fusion of spatial information on the channels. The structure is shown in Figure 7. Compared with the Squeeze and Excitation (SE) attention mechanism used by EfficientNet, the CA attention mechanism takes into account not only the relationship between channels but also the spatial feature information in X and Y dimensions to optimize the model training and thus improve the detection accuracy.

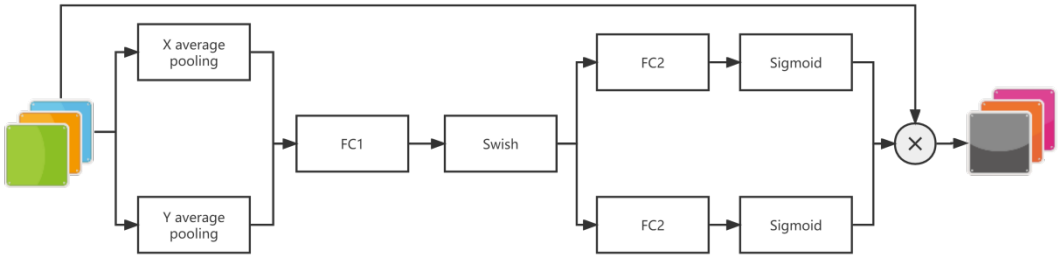

Fig. 6 CA attention module

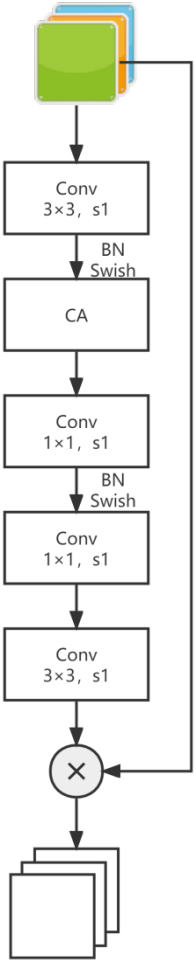

Fig. 7 CA Attention Module Improvement Module

As in the original network inverted residual module of the SE module in Fig. 4, the residuals are connected to the low-dimensional features after point convolution and compression of the channels, and some useful information may be lost in the process, leading to poor results. To solve the above problem, we use an improved inverse residual module with an imposed CA attention mechanism as shown in Fig. 7, which solves the above problem by the idea of flipping the inverse residual. the CA attention mechanism performs two lighter depth convolutions to give more spatial information than Squeeze-and-Excitation, which encodes more spatial information. In Fig. 7, firstly, the high-dimensional layer is subjected to 3x3 or 5x5 deep convolution to extract features, then the CA attention mechanism is added, followed by 1x1 channel compression and 1x1 channel dilation, and finally another 3x3 or 5x5 deep convolution, while the input and output are summed to form a "convolution-compression-dilation-convolution" data flow [56]. In the experiments, the EfficientNet reduced-dimensional backbone feature layer is extracted at Blockcore after adjusting the size and number of channels, which has the same feature layer size and number of channels compared with the original EfficientNet, but the residual edges are shorted to the input and output of high-dimensional channels, thus passing more information from the input tensor to the output tensor [54].

4.2 Introduction of depthwise separable convolution

Some lightweight networks, such as mobilenet, will have depthwise separable convolution, which is a combination of depthwise (DW) and pointwise (PW) components [47][48], and is used to extract feature maps. compared to conventional convolution operations , its number of parameters and operational cost are relatively low. For a 5×5×3 input, if we want to get a 3×3×4 feature map, then the shape of the convolution kernel is 3×3×3×4; if padding=1, then the output feature map is 5×5×4. There are four Filters in the convolution layer, and each Filter contains three Kernels, and the size of each Kernel is 3×3. Therefore, the number of parameters of the convolution layer can be calculated by the following formula (i.e., convolution kernel W × convolution kernel H × number of input channels × number of output channels): N_std = 4 × 3 × 3 × 3 = 108. The amount of computation (i.e., convolution kernel W × convolution kernel H × (picture W - convolution kernel W + 1) × (picture H - convolution kernel H + 1) × number of input channels × number of output channels, with padding= 0, no padding for demonstration, the output is 3×3×4, if padding convolution kernel W× convolution kernel H×(picture W-convolution kernel W+2P+1)×picture H-convolution kernel H+2P+1)×number of input channels×number of output channels): C_std =3×3×(5-2)×(5-2)×3×4=972. Since each spatial kernel needs to be considered independently. Therefore the length of dot product operations, which are usually accelerated by vector-product type hardware, is limited. This means that the hardware cannot always be fully utilized, resulting in "wasted"

cycles. Depthwise convolutions also have very low computational efficiency because they require a large amount of data transfer relative to the number of FLOPs performed, which means that memory access speed is an important factor [53]. While this may limit the throughput on alternative hardware, the memory architecture in the processor provides high bandwidth memory access, which can significantly improve the performance of low computationally efficient operations like this one. Experimental data suggest that deep convolutions are found to be most efficient when they are sandwiched between two pointwise "projection" convolutions, forming an MBConv block. These pointwise convolutions increase or decrease the dimensionality of the activation by an "expansion factor" around the spatial depth convolution. While this scaling can lead to good task performance, it also creates a very large activation tensor, which can dominate memory requirements and ultimately limit the maximum batch size available.

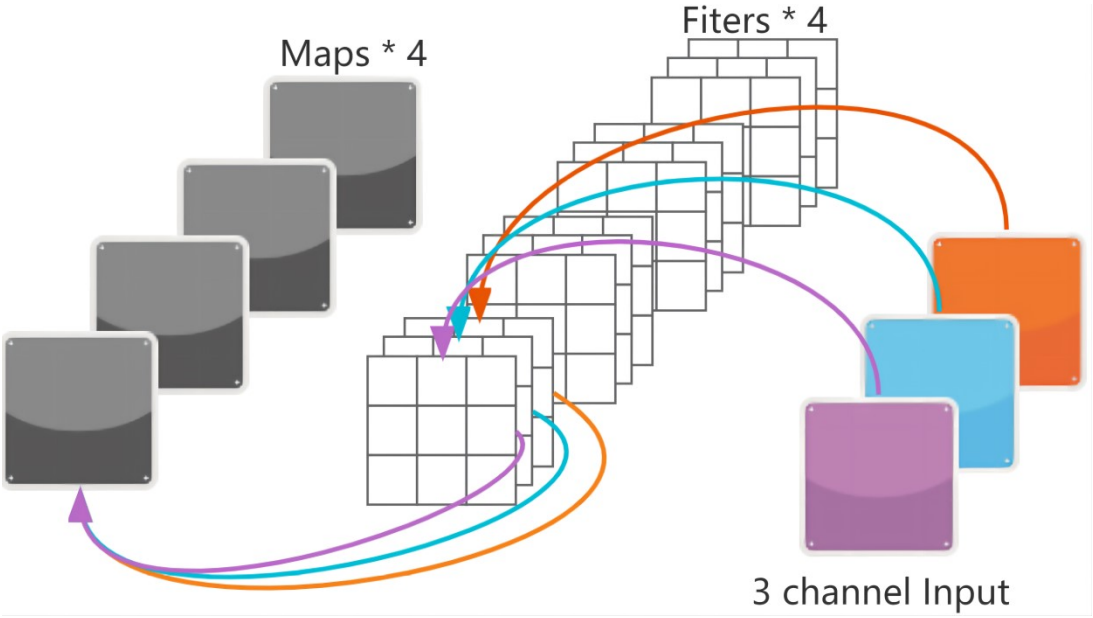

Fig. 8 Depth-separable convolutional neural network

Depthwise Convolution is divided into two main processes, Depthwise Convolution and Pointwise Convolution. In Depthwise Convolution, one convolution kernel is responsible for one channel, and a channel is convolved by only one convolution kernel, and this process produces exactly the same number of channels of the feature map as the number of channels of the input [47]. A 5 × 5 pixel, three-channel color input image (with a SHAPE of 5 × 5 × 3), Depthwise Convolution first goes through the first convolution operation, and DW is performed entirely in the 2D plane. The number of convolution kernels is the same as the number of channels in the previous layer (channels and convolution kernels correspond one-to-one). So a three-channel image is generated after the operation of three Feature map (if there is a same padding, then the size is the same as the input layer is 5 × 5), (the shape of the convolution kernel is:

convolution kernel W × convolution kernel H × number of input channels) where a Filter contains only a size of 3 × 3 Kernel, the number of parameters of the convolution part is calculated The number of parameters of the convolution part is calculated as follows (i.e.: Convolution Kernel W x Convolution Kernel H x number of input channels): N_depthwise = 3 x 3 x 3 = 27. The amount of computation is (Convolution Kernel W x Convolution Kernel H x (Picture W - Convolution Kernel W+1) x (Picture H - Convolution Kernel H+1) x number of input channels) C_depthwise = 3 x 3 x (5-2) x (5-2) x 3 = 243. Depthwise Convolution completes the same number of Feature maps as the number of channels in the input layer, which cannot expand the Feature maps. and this operation performs the convolution operation independently for each channel in the input layer, which does not effectively use the Feature information of different channels in the same spatial location. Therefore, Pointwise Convolution is needed to combine these Feature maps to generate a new Feature map.

The Pointwise Convolution operation is very similar to the regular convolution operation, which has a convolution kernel of size 1 × 1 × M, with M being the number of channels in the previous layer [48]. So the convolution operation here will combine the maps of the previous step in the depth direction with weighting to generate a new Feature map. several convolution kernels will have several output Feature maps.(The shape of the convolution kernel is: 1×1×number of input channels×number of output channels). The number of parameters involved in the convolution in this step can be calculated as (i.e., 1×1×number of input channels×number of output channels) N_pointwise = 1×1×3×4 = 12, and the amount of computation (i.e., 1×1×feature layer W×feature layer H×number of input channels×number of output channels): C_pointwise = 1×1×3×3×4 = 108. After Pointwise Convolution, the same 4 Feature maps are output, with the same output dimension as the conventional convolution.

4.3 Proxy normalization activation

Normalization of the output of convolution and matrix multiplication operations has become a fundamental operation in CNNs at this stage, with Batch Normalization being the most common formal method used for this purpose [54]. However, the batch size limitation introduced by batch normalization is a well-known problem that has triggered a series of innovations in batch-independent alternative methods. Although many methods work well in the ResNet model, we found that none of them achieve the same performance as the batch normalization of EfficientNet. To address the lack of alternative methods for batch normalization, we utilize the new batch-independent normalization method introduced in the paper Proxy-Normalizing Activations to Match Batch Normalization while Removing Batch Dependence --Proxy Normalization. This approach builds on the already successful group (and layer) normalization methods. Group normalization and layer normalization suffer from the problem

that activations may become non-normalized on the channel. This problem becomes worse with increasing depth, as non-normalization is accentuated at each level. While this problem can be avoided by simply reducing the group size in group normalization, this reduction in group size will change expressivity and degrade performance. Proxy normalization provides a better solution by preserving expressivity while offsetting the two main sources of non-normalization: affine transformations and activation functions, following group normalization or layer normalization [56]. Specifically, the non-normalization is canceled by assimilating the output of the group parametrization or layer parametrization to a Gaussian "proxy" variable and applying the same affine transformation and the same activation function to this proxy variable. The statistics of the non-normalized proxy variable are then used to correct for the expected distribution bias in the true activation. Proxy normalization allows us to maximize the group size (i.e., using layer normalization), and proxy normalization maintains expressivity even without the channel non-normalization problem. It is important that this approach does not mimic any of the implied regular features of batch normalization. For this reason, additional regularization is needed in this work, and we use a combination of hybrid and shear blending. In comparing the performance of layer normalization + agent normalization (LN+PN) with two batch normalized (BN) baseline models under standard preprocessing and AutoAugment (AA) [58], we found that LN+PN matched or exceeded the standard preprocessing performance of BN over the entire range of model sizes. Moreover, LN+PN is almost as good as BN on AA, although AA requires an expensive process of training augmentation parameters. Figure 9 shows the agent normalized activation module, and the red convolution blocks indicate with additional agent normalized activation operations.

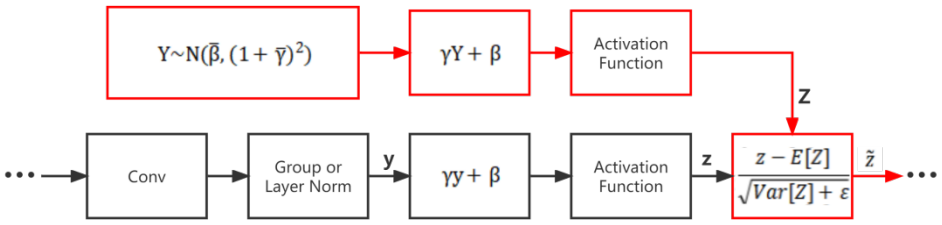

Fig. 9 Proxy normalized activation

5 Experiments

5.1 Data set composition and design ideas

This thesis will use efficientnet-b0 to recognize and classify the front end and rear end of the mine car with 12211 images of data, which contains 4073 images of the front end and 4065 images of the rear end. The first step is to divide the data set

by python code and divide the data set into training set and test set, which are saved in the target folder. Set the ratio of training set to test set to 3:2 for training and testing respectively. The second step is to download the model of efficientnet-b0 and put the model of efficientnet-b0 into the project, then install the third-party libraries such as torch,efficientnet_pytorch, etc., and test whether the installation is successful after the installation is completed, and check through various channels whether torch,efficientnet_ pytorch and other third-party libraries, Google and related efficientnet papers to learn, write for their own data set training and testing methods, and then use their own python code written on the divided data first training, get training model, and then use the trained model, test the data on the test set, test results, so as to achieve the mining car mine head and tail recognition. Figure 10 shows the distribution of the number of training set and test set in the dataset.

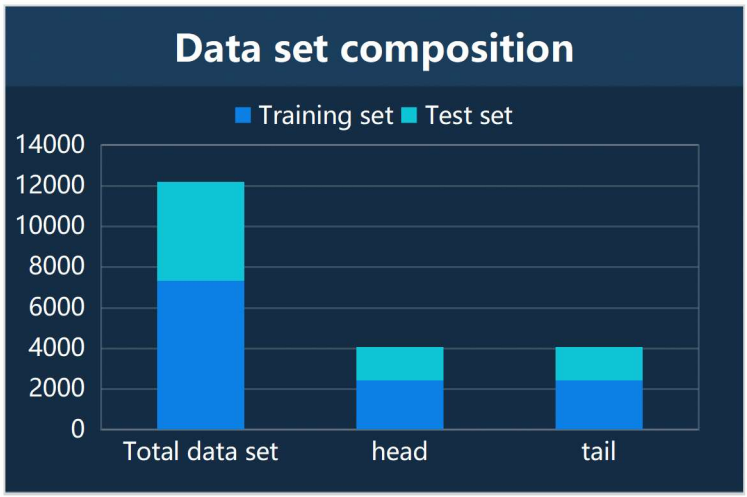

Fig. 10 Distribution of the number of training and test sets in the dataset

5.2 Development environment and network parameters

The platform is developed based on the python language, using PyCharm 2020.3.3 x64 and Anaconda3 2020.07 (Python 3.9 64-bit) as the development tools for the platform. Because Anaconda comes with many common third-party libraries such as Python, Anaconda Prompt. torch library is a third-party library for Python, which is used in large-scale machine learning applications and has great adaptability for image and video recognition. torchvision can mainly be used to load images and perform simple operations on them. The experimental training environment is shown in Table 3.

Table 3. Training environment

CPU：i7-8750H	OS：Window10
GPU:Nvidia Geforce GTX1050	Pytorch：1.5
内存：8G	CUDA：10.1
Python：3.9	Torchvision：0.6

Compiler：Anaconda、PyCharm	Numpy：1.18

Table (4) Network training parameters

Lr0	0.01
weight_decay	0.8
epoch	100
momentum	0.9
Img-size	256

As an important superparameter in supervised learning and deep learning, the learning rate of Ir0 determines whether and when the objective function can converge to a local minimum. A larger learning rate will accelerate the network learning in the early stage of training and make the model converge to the optimal solution more easily. However, in the later stage, there will be large fluctuations, and even the value of the loss function hovers around the minimum value, fluctuating greatly, and it is always difficult to reach the optimum. The appropriate learning rate can make the target function converge to the local minimum in a suitable time. Momentum is set to 0.9, momentum (momentum) is a common acceleration technique in gradient descent method to speed up the convergence, momentum method is mainly to solve the problem of Hessian matrix pathological condition; weight decay (weight_decay), to prevent overfitting. Define the training model train_model, define a parameter best_model_wts to store the weights and paranoia coefficients that need to be learned during the training process, then enable batch normalization and drop out, compress the dimensionality of the data, remove the dimensionality of dimension 1.

labels=torch. squeeze(labels.type(torch.LongTensor)) （13）

Variable can be seen as a wrapper for the tensor, which contains not only the content of the tensor, but also information such as the gradient, so the Variable data structure is often used in neural networks. Receive the returned values and take the value with the highest probability. Set the gradient to zero, calculate the loss to be returned after getting the loss, then update the parameters, and finally save the training model. The data width of 1920 and height of 1080 are not suitable for training, so we need to preprocess the data, process the data into width and height of 256, and put it into bishe-enddata2 folder, set the batch size to 8, set the training round to 10, set the dataset image processing size to 256, set the category to 3, and set the name of the model used to EfficientNet-B0.

5.3 Module Function Design

(1) mixup data augmentation

mixup is an unconventional data augmentation method, a simple data augmentation principle independent of the data [50], which constructs new training samples and labels by linear interpolation, and the final processing of the

labels is shown in the following equation.

$$\tilde{x} = \lambda x_i + (1-\lambda)x_j \quad (14)$$

$$\tilde{y} = \lambda y_i + (1-\lambda)y_j \quad (15)$$

The two data pairs (x_i, y_i), (x_j, y_j) are the training sample pairs (training samples and their corresponding labels) in the original dataset. Where λ is a parameter obeying the β distribution, $\lambda \sim \beta(\alpha, \alpha)$. The probability density function of the β distribution is shown in Figure 11 below, where $\alpha \in [0, +\infty]$. Thus α is a hyperparameter, and as α increases, the training error of the network increases, and its generalization ability is enhanced accordingly. And when $\alpha \to \infty$, the model degrades to the most primitive training strategy.

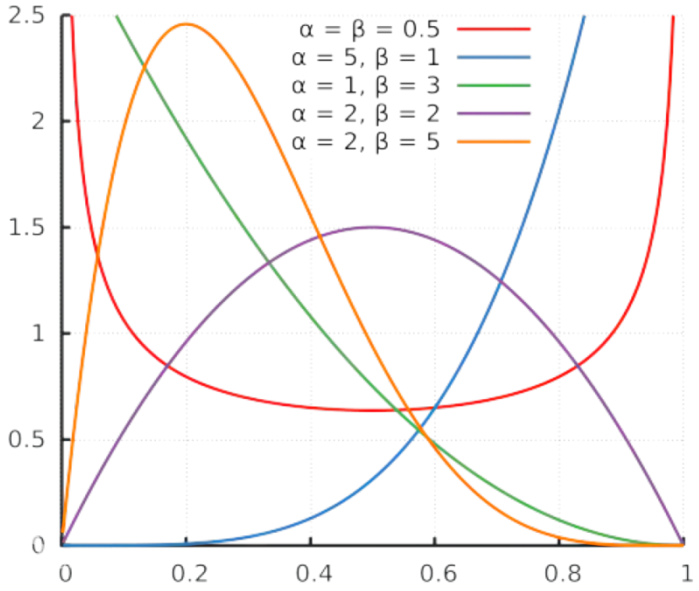

Fig. 11 Plot of probability density function of B distribution

(2) Load and import the data module and set the global parameters

Because tensorflow2.0 or above integrates Keras, we do not need to install Keras separately when we use it, if we use the initial model module, we need to upgrade it to tensorflow2.0 or above, and then add tensorflow in front of keras. Next, according to the experimental computer hardware and software equipment reasonable debugging and use the relevant important global parameters, such as norm_size = 224, EfficientNetB0 default image size is 224 × 224, our experiments taken by the Img-size of 256. datapath = 'data/train ', set the path where the images are stored, here we have to explain that because this experiment requires a large set of images, if the experimental image dataset is large, please do not put it in the project directory, otherwise Pycharm will browse all the images when

loading the project, resulting in a slow computer. epochs = 100, the number of epochs according to the experimental classnum = 2, the number of target categories for the experimental detection, the dataset has 2 categories for the front and back of the car. batch_size = 16, the batchsize, set according to the hardware and the size of the dataset, too small will cause the loss to float too big, too big will cause the float to float too big as the experiment progresses. The final convergence effect is not ideal, according to our previous experimental experience, generally set to the second power of 2 (4, 8, 16, etc.). windows can be viewed through the task manager of the memory occupation. Finally, a loaddata function is defined to pass in the local data, the number of batches to be trained as parameters, then the data is augmented and finally a batch is created to generate the input for the real network, with the following code Figure 12.

```
def loaddata(data_dir, batch_size, set_name, shuffle):
    data_transforms = {
        'train': transforms.Compose([
            transforms.Resize(input_size),
            transforms.CenterCrop(input_size),
            transforms.RandomAffine(degrees=0, translate=(0.05, 0.05)),
            transforms.RandomHorizontalFlip(),
            transforms.ToTensor(),
            transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
        ]),
        'test': transforms.Compose([
            transforms.Resize(input_size),
            transforms.CenterCrop(input_size),
            transforms.ToTensor(),
            transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
        ]),
    }

    image_datasets = {x: datasets.ImageFolder(os.path.join(data_dir, x), data_transforms[x]) for x in [set_name]}
    # num_workers=0 if CPU else =1
    dataset_loaders = {x: torch.utils.data.DataLoader(image_datasets[x],
                                                      batch_size=batch_size,
                                                      shuffle=shuffle, num_workers=1) for x in [set_name]}
    data_set_sizes = len(image_datasets[set_name])
    return dataset_loaders, data_set_sizes
```

Fig. 12 Loading data module

(3) Image enhancement

First read the image and then resize the image with the specified size. The images are converted into arrays by image normalization, and finally the labels are converted into one-hot codes using LabelBinarizer. After preparing the data import process, we need to slice the training set and the test set, and the experiments involved in this paper are slice according to the ratio of 3:2. To slice the dataset, we use the train_test_split() method, which requires importing the from sklearn.model_selection import train_test_split package. imageDataGenerator() is the image generator in the keras.preprocessing.image module. It can also augment the data in batch to expand the size of the dataset and enhance the generalization ability of the model. For example, rotate, deform, normalize, etc.

(4) Training model module

Define the training model train_model, define a parameter best_model_wts to

store the weights and paranoia coefficients to be learned during the training process, then enable batch normalization and drop out, compress the dimensionality of the data, remove the dimensionality of dimension 1 labels = torch. squeeze (labels.type(torch.LongTensor)). Variable can be seen as a wrapper for tensor, which contains not only the content of the tensor, but also information such as gradient, so Variable data structure is often used in neural networks. Receive the returned value and take the value with the highest probability. The gradient is set to zero and the loss is calculated and then the loss is returned, then the parameters are updated and finally the training model is saved, the training results of ResNet34 [51] model are shown in Figure 13 and the training results of EfficientNet-B0 model are shown in Figure 14 below, initially the model training only takes 10 epochs in order to be able to better see the two It is obvious from the figure that the effect of EfficientNet operation and the convergence speed of ResNet34 have been significantly improved with the experiments, and the EfficientNet-B0 model has some advantages under the same experimental equipment and environment.

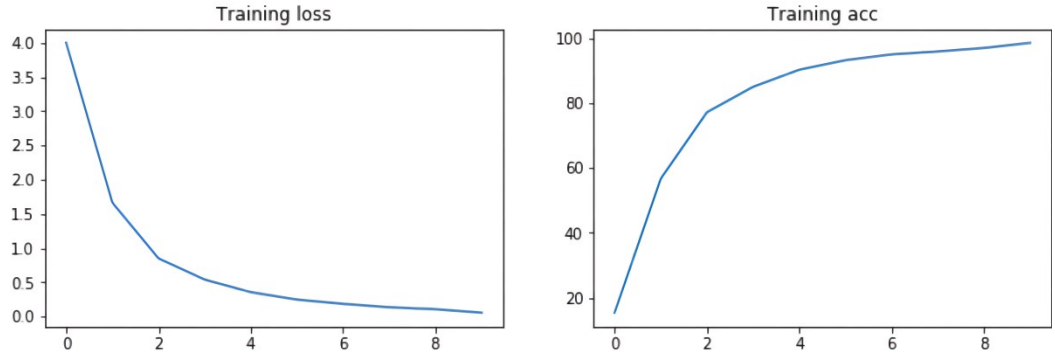

Fig. 13 ResNet34 model training results

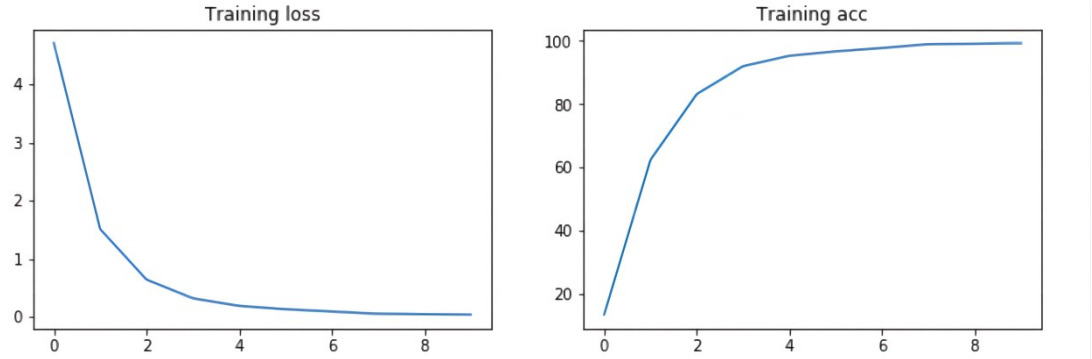

Fig. 14 EfficientNet-B0 model training results

(5) Test model module

Use the data loading module to import the data, keep the best model and set the learning rate dynamically, ModelCheckpoint: used to keep the best model, set the input and output, the callback function will save the model to filepath after each epoch, the filepath can be a formatted string, the placeholder inside will be filled

by the epoch value and the logs keyword passed into on_epoch_end is filled in, for example, if the filepath is weights.{epoch:02d-{val_loss:.2f}.hdf5, multiple files corresponding to epoch and validation set loss will be generated. Also when save_best_only: is set to True, only the model with the best performance on the validation set will be saved (mode: 'auto', 'min', 'max ' one of them). The evaluation criterion for the best performing model is determined when save_best_only=True, e.g. when the monitored value is val_acc, the mode should be max, and when the detected value is val_loss, the mode should be min. In auto mode, the evaluation criterion is automatically inferred from the name of the monitored value. If save_weights_only: is set to True, only the model weights are saved, otherwise the whole model (including model structure, configuration information, etc.) is saved, and the learning rate is reduced when ReduceLROnPlateau: evaluation metrics are not improving; when learning is stagnant, reducing the learning rate by a factor of 2 or 10 often gives better results. This callback function detects the condition of the metrics, and reduces the learning rate if no model performance improvement is seen in patience epochs. After the learning rate is reduced, it will go through cooldown epochs before the normal operation is performed again. We use classification_report to evaluate the validation set, import the package from sklearn.metrics import, keep the training results and generate the images, assign cpu to the input and output, and finally output Loss and Acc. For the accuracy of the experiment, we adjust the original 10 epochs upward to 100 The output results are shown in the following figure, Figure 15 shows the changes of Acc and Loss of EfficientNet test model with the increase of epoch.

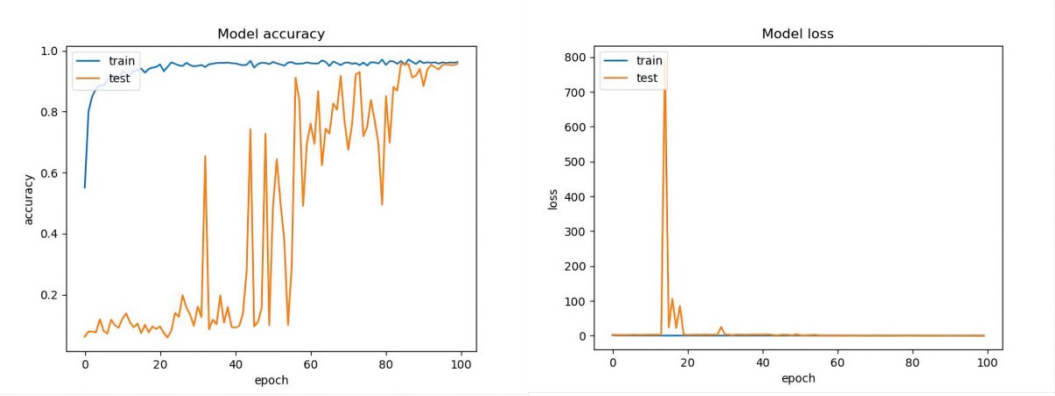

Fig. 15 Change of test model Acc, Loss with increasing epoch

5.4 Analysis of results

Running a query from Google's official website, the recognition accuracy of EfficientNet-B0 to EfficientNet-B7, AmoebaNet-A to AmoebaNet-B, Inception-v2 to SENet [49], ResNet-34 [51] to ResNet-152 [26], with time, the algorithms undergo continuous optimization, and the recognition rates are all significantly improved, and the results are obtained, as represented in Figure 16,

which shows the Top-1 Accuracy of EfficientNet_B0-B7 using MindSpore Vision suite compared to that using TensorFlow, and the recognition accuracy of EfficientNet_B0-B7 using MindSpore Vision Top-5 Accuracy of EfficientNet_B0-B7 using MindSpore Vision.

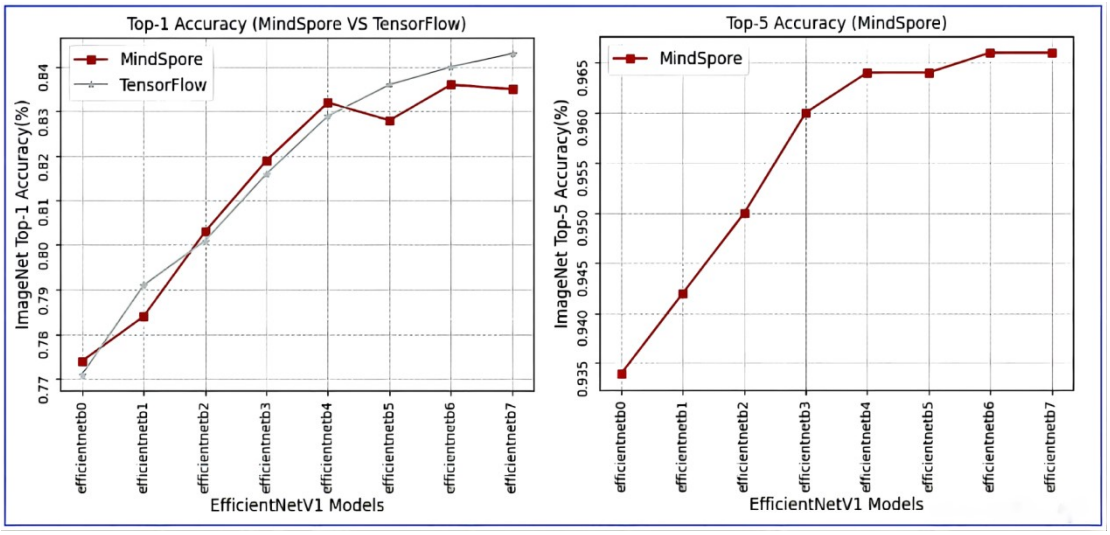

Fig. 16 Top-1 Accuracy of EfficientNet_B0-B7 with MindSpore Vision suite compared with TensorFlow and Top-5 Accuracy of EfficientNet_B0-B7 with MindSpore Vision

It is obvious from the figure that the accuracy of EfficientNet algorithm recognition rises very fast, from 76.3% in EfficientNet-B0 to 84.4% in EfficientNet-B7, EfficientNet compared with other deep learning networks, we can see very clearly that EfficientNet B0-B7 on ImageNet. In addition to this, we bring the algorithm to the head and tail experiments, as shown by the running results, Acc for accuracy, the training test was completed 5 times, after the complete reading of the experimental data, the result given is Training complete in 16m 25s.The results of the five experiments were 96.68%, 99.46%, 99.76%, 99.89%, and 99.97%, which were significantly higher than the previous deep convolutional neural network-based recognition accuracy. The detailed experimental parameters and results are shown in the following table. The use of efficientnet algorithm is realized, and the recognition of the head and tail of the mine car based on efficientnet algorithm is completed. Therefore, in the mineral operation, the efficientnet algorithm can be applied to the identification of the front and rear of the mine car, which is beneficial to observe whether the mine car is entering the car from the entrance and leaving the car from the exit, and when the identification of the entrance and exit of the mine car vehicles are not in accordance with this rule, it can be prompted and the gate will not be opened, and when the identification is the correct entry and exit rule, the gate will be opened and released, which is beneficial to the smooth operation of the mine land traffic and the mine car The rules of entry and exit release.

Table 5. Experimental operation results statistics

Experiment	Loss	Acc	Data Size	Epoch	LR
First	0.0923	0.9668	7448	100	0.01
Second	0.0174	0.9946	7448	100	0.01
Third	0.0090	0.9976	7448	100	0.01
Fourth	0.0046	0.9989	7448	100	0.01
Fifth	0.0015	0.9997	7448	100	0.01

5.5 Experimental comparison - deep convolutional neural network

The traditional deep convolutional neural network-based recognition system for the front end of mining vehicles due to the influence of computer hardware and experimental environment, the source data is first normalized to a size of 50 × 50, and then the obtained images are grayed out, using a 2-dimensional CNN as well as maximum sampling for model training. The experimental process of the backbone network uses 5 layers of convolution plus pooling, and ReLU is used as the activation function of the neurons in the middle hidden layer between each layer. The output layer of the detection network uses Fully_con→Dropout→Fully_con for further processing of the convolutional tail data, and finally the Softmax activation function is experimented to classify the data. The experiment uses 16000 pictures for training, where the pictures are not screened head, tail and other miscellaneous data, using 4000 head and tail for testing, the experimental results show that the accuracy curve of training basically converges to about 0.95, compared with it, our network still has a greater advantage, the specific network training parameters and environment comparison is shown in the following table.

Table 6. Comparison of Efficient Net and deep convolutional neural network parameters

Algorithm	（Lr0）	Img-size	Epoch	Python	Training set	Test set	val_Acc（%）	val_Loss（%）
EfficientNet	0.01	256	100	3.9	7327	4884	99%	3%
DeepCNN[17]	0.001	50	40	3.6	16000	4000	95%	10%

6 Summary and Future work

In the increasingly advanced modern traffic system, traffic management, traffic monitoring and traffic optimization require a lot of human and material resources. If this system can replace some tedious and repetitive management work, it will save a lot of unnecessary human and material resources. The implementation of Efficient Net algorithm based mine vehicle head-end recognition is an important

part of the implementation of intelligent mine traffic management, and is one of the key technologies for implementing intelligent traffic management. This paper describes the task and technical difficulties of mine vehicle refinement identification research, reviews several of the most popular convolutional neural networks and detection algorithms, analyzes the advantages and disadvantages of each algorithm, introduces the relevant data sets and evaluation criteria, and hopes to provide new information and ideas for researchers in the field of method selection. At the current stage, vehicle detection technology has become increasingly mature, but there are several elements that need to be further - researched as follows.

Because of its better feature extraction capability and computational efficiency, the EfficientNet network can effectively reduce the number of model parameters and computational complexity, thus improving the performance of the model. We try to extract advanced features from the input images through the EfficientNet backbone network to obtain a series of feature maps with different resolutions under the condition of limited computational resources. Combined with SOTA technique in ultralytics algorithm framework in one, the adaptive convolution layer is used to adjust the size of the convolution kernel adaptively according to the target size, so as to better handle targets of different sizes. Meanwhile, multi-level feature fusion can fuse feature maps of different scales together and use non-maximum suppression (NMS) algorithm for de-weighting and filtering, which can enhance the model's ability to perceive different regions and thus improve the detection accuracy of the model.

The predecessor of this paper, titled "Truck's Head and Tail Recognition Based on CA Modified Efficient-Net", has been presented at the conference of The Ninth International Conference on Data Science (ICDS) and is scheduled to appear in August 2023. We acknowledge this prior work.

Acknowledgment

This work was supported by the open fund of Information Materials and Intelligent Sensing Laboratory of Anhui Province with Grant No.IMIS202114, by the second batch of industry university cooperation collaborative education project of the Ministry of education in 2021 with No.202102326028, by key scientific research projects of Suzhou University in 2021 with No. 2021yzd01, by an open research topic under the National-Local Joint Engineering Research Center for Agricultural Ecological Big Data Analysis and Application Technology, with No. AE202210.

Reference.

[1] Zhang L, Zhang XL. Status and prospects of research on intelligent mining control technology and equipment [J]. Intelligent Mining, 2022, 3(5):9.

[2] Zhang Baocheng. Prospect of intelligent mine construction in Fushun East Open Pit Mine[J]. Surface Mining Technology, 2022, 37(3):4.
[3] Wang Guofa. The latest technological progress and problems of coal mine intelligence[J]. Coal Science and Technology, 2022, 50(1):27.
[4] Lu Xetong, Li Honglong, Sun Peng. Analysis of construction ideas based on intelligent mine[J]. 2021.
[5] Guo Xian. Construction framework of digital mine construction based on grid-based information[J]. Market Research Information:Comprehensive Edition, 2022(14):3.
[6] Lu Xiaoping. From "digital mine" to "intelligent mine".
[7] Yan Ling, Huang Jiade. Research on unmanned system for mining trucks [J]. Industrial and mining automation, 2021.
[8] Sliege. Discussion on system engineering and key technologies of intelligent mine[J]. World non-ferrous metals, 2019.
[9] Li J, Ma GY, Zhang B, et al. Method and system for identifying irregular operation of mining vehicles:, CN109404045A [P]. 2019.
[10] Deng Kewang, Zhao Huijie, Li Na,et al. An improved sample-driven model compression method for hyperspectral mineral identification[J]. Infrared and Laser Engineering, 2022, 51(3):9.
[11] Du S, Ibrahim M, Shehata M, et al. Automatic license plate recognition (ALPR): A state-of-the-art review[J]. IEEE Transactions on circuits and systems for video technology, 2012, 23(2): 311-325.
[12] Zhang Xinxin. Digital China wins in the future [J]. Outlook, 2022(8):7.
[13] Huo Zhonggang, Wu Xianli. Internet + intelligent mine development direction [J]. 2016.
[14] Lu S, Wang Y. Smart mine construction content and development prospect.
[15] Cheng Baojun, Chen Yuepeng, Zhang Tianshan. Intelligent mine control system application based on GIS one map and three-dimensional spatial data [J]. Shandong Coal Science and Technology, 2022, 40(5):3.
[16] Kaijie Zhang, C. Wang, Xiaoyong Yu, Aihua Zheng, Mingyue Gao, Zhenggao Pan, Guolong Chen,Zhiqi Shen, “Research on mine vehicle tracking and detection technology based on YOLOv5,”Systems Science & Control Engineering, Volume 10, Issue 1 (2022), pp. 347-366, Apr 22,2022
[17] JunQiang Li, C. Wang, GuoLong Chen, “Deep Convolution Neural Network based Research on Recognition of Mine Vehicle Head and Tail,” Lecture Notes in Computer Science, Springer-Verlag, LNCS 12836, pp. 594-605, 2021(International Conference on Intelligent Computing (ICIC 2021), ShenZhen, August 12-15, 2021) (EI， ISTP)
[18] Zhang Q,Cui L,Wang Chao. Research on vehicle refinement recognition based on YOLOv5 and LPRnet[J]. Changjiang Information Communication,2022,35(03):40-43.

[19] Shih F Y. Image processing and pattern recognition: fundamentals and techniques[M]. John Wiley & Sons, 2010.
[20] Recent Trends in Image Processing and Pattern Recognition: Third International Conference, RTIP2R 2020, Aurangabad, India, January 3–4, 2020, Revised Selected Papers, Part I[M]. Springer Nature, 2021.
[21] Lecun Y , Bottou L . Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11):2278-2324.
[22] Krizhevsky A , Sutskever I , Hinton G . ImageNet Classification with Deep Convolutional Neural Networks[J]. Advances in neural information processing systems, 2012, 25(2).
[23] Matthew Zeiler D, Rob F. Visualizing and understanding convolutional neural networks[C]. ECCV, 2014.
[24] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.
[25] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 1-9.
[26] He K , Zhang X , Ren S , et al. Deep Residual Learning for Image Recognition[J]. IEEE, 2016.
[27] Howard A G , Zhu M , Chen B , et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications[J].2017.
[28] Tan M, Pang R, Le Q V. Efficientdet: Scalable and efficient object detection[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020: 10781-10790.
[29] Qin J, Huang Y, Wen W. Multi-scale feature fusion residual network for single image super-resolution[J]. Neurocomputing, 2020, 379: 334-342.
[30] Tan M, Le Q. Efficientnet: Rethinking model scaling for convolutional neural networks[C]//International conference on machine learning. PMLR, 2019: 6105-6114.
[31] Sandler M, Howard A, Zhu M, et al. Mobilenetv2: Inverted residuals and linear bottlenecks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 4510-4520.
[32] Elsken T, Metzen J H, Hutter F. Neural architecture search: A survey[J]. The Journal of Machine Learning Research, 2019, 20(1): 1997-2017.
[33] Yuan Y, Fu R, Huang L, et al. Hrformer: High-resolution vision transformer for dense predict[J]. Advances in Neural Information Processing Systems, 2021, 34: 7281-7293.A
[34] Soltani M, Pourahmadi V, Mirzaei A, et al. Deep learning-based channel estimation[J]. IEEE Communications Letters, 2019, 23(4): 652-655.
[35] Pleiss G, Cunningham J P. The limitations of large width in neural networks: A deep Gaussian process perspective[J]. Advances in Neural Information

Processing Systems, 2021, 34: 3349-3363.
[36] Xu B, Wang N, Chen T, et al. Empirical evaluation of rectified activations in convolutional network[J]. arXiv preprint arXiv:1505.00853, 2015.
[37] Mete B R, Ensari T. Flower classification with deep cnn and machine learning algorithms[C]//2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT). IEEE, 2019: 1-5.
[38] Cordonnier J B, Loukas A, Jaggi M. On the relationship between self-attention and convolutional layers[J]. arXiv preprint arXiv:1911.03584, 2019.
[39] Gang S, Fabrice N, Chung D, et al. Character Recognition of Components Mounted on Printed Circuit Board Using Deep Learning[J]. Sensors, 2021, 21(9): 2921.
[40] Chen Y, Liu L, Phonevilay V, et al. Image super-resolution reconstruction based on feature map attention mechanism[J]. Applied Intelligence, 2021, 51(7): 4367-4380.
[41] Yoon C R, Kim D H. Mobile Convolutional Neural Networks for Facial Expression Recognition[C]//2020 International Conference on Information and Communication Technology Convergence (ICTC). IEEE, 2020: 1315-1317.
[42] Khan Z Y, Niu Z. CNN with depthwise separable convolutions and combined kernels for rating prediction[J]. Expert Systems with Applications, 2021, 170: 114528.
[43] Huang G, Sun Y, Liu Z, et al. Deep networks with stochastic depth[C]//European conference on computer vision. Springer, Cham, 2016: 646-661.
[44] Ketkar N. Introduction to keras[M]//Deep learning with Python. Apress, Berkeley, CA, 2017: 97-111.
[45] Howard A, Sandler M, Chu G, et al. Searching for mobilenetv3[C]//Proceedings of the IEEE/CVF international conference on computer vision. 2019: 1314-1324.
[46] Tian Y, Zhang Z, Yang Z, et al. JMSNAS: joint model split and neural architecture search for learning over mobile edge networks[C]//2022 IEEE International Conference on Communications Workshops (ICC Workshops). IEEE, 2022: 103-108.
[47] Zheng Q, Li X, Wang Z, et al. Mobilattice: A depth-wise dcnn accelerator with hybrid digital/analog nonvolatile processing-in-memory block[C]//2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD). IEEE, 2020: 1-9.
[48] Goetschalckx K, Verhelst M. DepFiN: A 12nm, 3.8 TOPs depth-first CNN processor for high res. image processing[C]//2021 Symposium on VLSI Circuits. IEEE, 2021: 1-2.
[49] Jie H , Li S , Gang S . Squeeze-and-Excitation Networks[J]. IEEE, 2018.
[50] Fu Y, Wang H, Xu K, et al. Mixup based privacy preserving mixed

collaboration learning[C]//2019 IEEE International Conference on Service-Oriented System Engineering (SOSE). IEEE, 2019: 275-2755.
[51] Koonce B. ResNet 34[M]//Convolutional Neural Networks with Swift for Tensorflow. Apress, Berkeley, CA, 2021: 51-61.
[52] Hou Q, Zhou D, Feng J. Coordinate attention for efficient mobile network design[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021: 13713-13722.
[53] Masters D, Labatie A, Eaton-Rosen Z, et al. Making EfficientNet more efficient: Exploring batch-independent normalization, group convolutions and reduced resolution training[J]. arXiv preprint arXiv:2106.03640, 2021.
[54] Labatie A, Masters D, Eaton-Rosen Z, et al. Proxy-Normalizing Activations to Match Batch Normalization while Removing Batch Dependence[J]. Advances in Neural Information Processing Systems, 2021, 34: 16990-17006.
[55] Touvron H, Vedaldi A, Douze M, et al. Fixing the train-test resolution discrepancy[J]. Advances in neural information processing systems, 2019, 32.
[56] Song LY, Liu SH, Wang K, et al. An improved EfficientDet-based method for grid component and defect identification[J]. Journal of Electrical Engineering Technology, 2022, 37(9):11.
[57] Cai Z, Ravichandran A, Maji S, et al. Exponential moving average normalization for self-supervised and semi-supervised learning[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021: 194-203.
[58] Momeny M, Neshat A A, Gholizadeh A, et al. Greedy Autoaugment for classification of mycobacterium tuberculosis image via generalized deep CNN using mixed pooling based on minimum square rough entropy[J]. Computers in Biology and Medicine, 2022, 141: 105175.