



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

**Research on image super-resolution
reconstruction method based on residual
attention network**

by

Yachao Li

Supervisor: Gordon Dickers

Project submitted as part of the requirements for the award of
MSc Software Engineering

November 2021

DECLARATION

I, Yachao Li FULL NAME Yachao Li declare that I am the sole author of this Project; that all references cited have been consulted; that I have conducted all work of which this is a record, and that the finished work lies within the prescribed word limits.

This work has not previously been accepted as part of any other degree submission.

Signed : Yachao Li

Date : 2020/11/20

FORM OF CONSENT

I Yachao Li FULL NAME Yachao Li hereby consent that my Project, submitted in candidature for the M.Sc. Applied Computing degree, if successful, may be made available for inter-library loan or photocopying (subject to the law of copyright), and that the title and abstract may be made available to outside organisations.

Signed : Yachao Li

Date : 2020/11/20

Abstract

The application of image reconstruction technology in various electronic devices affects our daily life, but there is still room for improvement in modern image reconstruction technology. For the problem of detail texture blur in natural image super-resolution reconstruction, a single image super-resolution reconstruction method based on residual attention network is proposed. This method creatively constructs a convolution neural network using the pytorch framework, adds the attention mechanism module to the network to enable the network to learn more high-frequency information in the channel, adds the residual learning module to enable the network to learn more shallow feature information, and uses the sub-pixel convolution method for up sampling. The classical div2k natural image data set is used as the training data set, the parameters are adjusted, and the reconstruction model is obtained after training.

The ablation experiment is carried out on the network model, and then compared with the mainstream image super-resolution network. The experimental results show that the objective evaluation index and visual quality of this method are better than several other representative methods. The reconstruction index is improved by 0.5% ~ 2%, and the edge of the reconstructed image is clearer.

TABLE OF CONTENTS

ABSTRACT.....	3
TABLE OF CONTENTS.....	错误！未定义书签。
1 INTRODUCTION.....	8
1.1 OBJECTIVE 1.....	9
1.2 OBJECTIVE 2.....	9
1.3 STRUCTURE OF THE PAPER.....	10
2 RESEARCH AND LITERATURE REVIEW.....	12
2.1 OVERVIEW OF IN-DEPTH LEARNING.....	12
2.2 CONVOLUTIONAL NEURAL NETWORK.....	14
2.3 SUPER-RESOLUTION RECONSTRUCTION.....	16
2.4 ATTENTION MECHANISM.....	19
2.5 LEARNING RATE.....	22
2.5.1 Discrete Staircase.....	22
2.5.2 Exponential Decay.....	22
2.5.3 Score Reduction.....	22
2.6 ACTIVATION FUNCTION.....	23
2.6.1 What is an Activation Function.....	23
2.6.2 Type of Activation Function.....	23
2.6.2.1 Sigmoid Activation Function.....	23
2.6.2.2 Tanh Activation Function.....	25
2.6.2.3 Linear Rectifier Function (ReLU)	25
2.7 LOSS FUNCTION.....	26
2.7.1 Mean Square Error.....	26
2.7.2 Average Absolute Error.....	27
2.7.3 Mean Bias Error.....	28
2.7.4 Cross Entropy Loss / Negative Log Likelihood:.....	28
2.8 GRADIENT DESCENT ALGORITHM.....	29
2.8.1 Algorithm Principle.....	29
2.8.2 Gradient Descent Optimization Algorithm Adam.....	30

2.9 INTRODUCTION AND COMPARISON OF MAINSTREAM DEEP LEARNING FRAMEWORKS.....	32
2.9.1 Common Software Tools.....	32
2.10 QUALITY EVALUATION STANDARDS.....	34
2.10.1 PSNR value of peak signal-to-noise ratio.....	35
2.10.2 SSIM value of structural similarity.....	35
2.11 CONCLUSION.....	36
3 RESEARCH METHODS.....	37
3.1 RESEARCH STEPS.....	37
3.1.1 Define the Research Theme:.....	37
3.1.2 Define the Research Object:.....	37
3.1.3 Formulate Research Plan and Research Tools.....	37
3.1.4 Analysis and Interpretation of Data.....	38
3.1.5 Writing Papers:.....	38
3.2 RESEARCH METHODS.....	38
3.3 PROJECT RESOURCES.....	39
3.3.1 Experimental Data.....	39
3.3.2 Hardware Resources.....	39
3.3.3 Software Resources.....	39
3.3.4 model training.....	39
4 DESIGN OF NETWORK MODEL BASED ON RESIDUAL ATTENTION.....	40
4.1 NETWORK STRUCTURE.....	40
4.2 SHALLOW RESIDUAL FEATURE INFORMATION EXTRACTION.....	42
4.3 FEATURE EXTRACTION.....	44
4.3.1 Introduction of Attention Mechanism.....	44
4.3.2 Introduction of Dense Connectivity and Residual Mechanisms.....	45
4.4 UP-SAMPLING.....	46
5 EXPERIMENT.....	47
5.1 EXPERIMENTAL PARAMETER SETTING AND TRAINING PROCESS.....	47
5.2 EXPERIMENTAL SIMULATION AND RESULT ANALYSIS:.....	48
5.2.1 Objective Evaluation.....	48
5.2.1.1 Ablation Experiment.....	48

5.2.1.2 Comparison with Other Algorithms.....	50
5.2.2 Subjective Evaluation.....	50
5.2.3 Comprehensive Evaluation.....	52
6 CONCLUSION AND FUTURE WORK.....	53
REFLECTION.....	53
REFERENCE.....	55
APPENDICES.....	57

TABLE OF FIGURES

FIG. 1	STRUCTURE DIAGRAM OF FULL CONNECTION DEPTH NEURAL NETWORK.....	14
FIG. 2	STRUCTURE OF CONVOLUTIONAL NEURAL NETWORK.....	14
FIG. 3	AVERAGE POOLING OPERATION DIAGRAM.....	15
FIG. 4	DROPOUT SCHEMATIC DIAGRAM.....	16
FIG. 5	SRCNN STRUCTURE.....	17
FIG. 6	VDSR STRUCTURE DIAGRAM.....	17
FIG. 7	DRCN STRUCTURE.....	18
FIG. 8	ESPCN STRUCTURE.....	18
FIG. 9	TWO AND THREE LAYERS ResNET RESIDUAL LEARNING MODULE.....	18
FIG. 10	EDSR STRUCTURE DIAGRAM.....	19
FIG. 11	DENSENET STRUCTURE DIAGRAM.....	19
FIG. 12	SE-NET ATTENTION MECHANISM.....	20
FIG. 13	SK-NET ATTENTION MECHANISM.....	21
FIG. 14	CBAM.....	21
FIG. 15	SIGMOID ACTIVATION FUNCTION AND SIGMOID DERIVATIVE.....	24
FIG. 16	TANH ACTIVATION FUNCTION AND TANH DERIVATIVE.....	25
FIG. 17	RELU ACTIVATION FUNCTION AND RELU DERIVATIVE.....	26
FIG. 18	RELATIONSHIP BETWEEN MSE LOSS (Y-AXIS) AND PREDICTION ERROR (X-AXIS).....	27
FIG. 19	RELATIONSHIP BETWEEN MAE LOSS (Y-AXIS) AND PREDICTION ERROR (X-AXIS).....	28
FIG. 20	TRAINING EFFECTS OF DIFFERENT LEARNING RATES.....	31
FIG. 21	OVERALL NETWORK STRUCTURE BASED ON RESIDUAL ATTENTION MECHANISM.....	41
FIG. 22	ERB MODULE STRUCTURE DIAGRAM.....	43
FIG. 23	ECA-NET STRUCTURE.....	44
FIG. 24	DECAB MODULE STRUCTURE DIAGRAM.....	46
FIG. 25	LOSS CURVE DURING NETWORK TRAINING.....	48
FIG. 26	SUPER RESOLUTION IMAGE USING DIFFERENT ALGORITHMS ($\times 4$)	52

1 INTRODUCTION

In recent years, with the rapid development of image resolution of mobile phones, computers, televisions and other display devices, there is an increasing demand for high-resolution images in all walks of life. Accordingly, with the development of artificial intelligence technology, researchers continue to propose and improve different super-resolution reconstruction technologies. Super resolution image reconstruction refers to the reconstruction of existing single or multiple low resolution images to obtain high-resolution images through super-resolution technology. This technology can realize that the same image content contains more pixel information after reconstruction and provide images with better visual effect. At present, super-resolution algorithms have been fully developed. A large number of algorithms based on interpolation[1], reconstruction[2], learning[3] have been proposed to realize super-resolution image reconstruction. This technology has been used to replace expensive image sensors in many fields, such as medical imaging, security monitoring, remote sensing[4-7]. At present, the mainstream research direction is learning based super-resolution. Because deep learning has the ability to generate multi-dimensional channel information and can achieve better reconstruction effect, it has better effect than the other two methods in terms of reconstructed image quality and reconstruction efficiency[8].

However, there are still some problems when using deep learning for super-resolution reconstruction of various images: one is that the existing SR reconstruction methods continue to deepen the number of layers of deep learning network to achieve better performance, which inevitably increases the amount of parameters and the complexity of the model. Another problem is that the previous SR depth network uses a single model to process all the texture information in the image, does not consider the frequency characteristics of the image, and the rich frequency information is not learned and lost; When the network is deeper and

deeper, it will lead to the phenomenon of over fitting of low-frequency information and under fitting of high-frequency information.

In order to solve these problems, this topic uses the innovative network model to improve the fuzzy problem and further improve the reconstruction efficiency and image reconstruction effect. This topic mainly studies the image reconstruction network from three aspects: 1) improving the network to improve the reconstruction effect; 2) selecting the appropriate optimization algorithm and loss function to train the best model; 3) ablation experiment and comparative experiment to evaluate the improved network

1.1 OBJECTIVE 1

The existing super-resolution network is optimized and innovated, and an image super-resolution reconstruction network based on residual attention mechanism is proposed. This method can adaptively learn multi-dimensional channel information, make better use of the high-frequency characteristic information in the convolution layer channel, and maintain high performance while adding a small amount of parameters to the network; The residual learning module is designed and added to the network model. The network with low shallow complexity processes the low-frequency information in the image, and then transmits the high-frequency information to the network with high deep complexity for processing, which effectively avoids the problems of over fitting of low-frequency information and under fitting of high-frequency information; Optimize CNN network with global features, local fusion and dense connection to improve network performance.

1.2 OBJECTIVE 2

Read and summarize the current literature on super-resolution reconstruction based on convolutional neural network. Deepen the understanding of the basic concepts and framework knowledge of super-resolution reconstruction, understand the

research background and significance of natural image super-resolution reconstruction, clarify the research status and existing problems in this field, and then determine the research objectives and contents according to the learned knowledge.

Download the natural image data set DIV2K[9] from GitHub as training data and the Set5 data set as verification data; As the basis of experimental training.

The preprocessing operations such as translation, flip and clipping are carried out on the image to realize the preprocessing of the training data set. The convolution neural network model is built, and the number of network layers in different extraction stages is set. The data set is degraded and added to the network Summary for training. Set the parameters such as Padding, Stripe, convolution kernel size and channel number, adjust the learning rate and Epoch times, use Adam algorithm as the optimization function of gradient descent method, and use L1 norm loss function (MAE) to guide network training. Finally, the reconstruction effect of the test set is improved by 0.5% ~ 2%, and the reconstructed image edge is clearer. The experimental results can verify the effectiveness of the improved network.

Set5, set14, bsd100, urban100 and manga109 data sets are used as test data sets. Save the optimal training model, compare the objective indexes and subjective effects with the mainstream network model, and compare the gap, so as to prove the performance of this method

1.3 STRUCTURE OF THE PAPER

This subject consists of four parts.

The first part is literature review, which combs and summarizes the knowledge of deep learning, convolutional neural network and image super-resolution, including the learning of classical network model, optimization method, loss function, learning rate, learning tools, quality evaluation standards and other main knowledge. These knowledge backgrounds provide theoretical basis and implementation methods for the implementation of this subject.

The second part is an overview of the research methods. Clarify the analysis and design methods and research steps of the subject, prepare the data, software and hardware tools required by the experiment, and clarify the experimental training process.

The third part introduces in detail how to improve and design our own neural network, the improvement work of the whole network, and the improvement idea and effect of the improved modules in the model.

The fourth part is to improve the network model and verify its performance. Firstly, ablation experiments are carried out to verify the effectiveness of our innovation in super-resolution reconstruction network; Then we verify the network performance from two parts: subjective evaluation and objective evaluation. This model is compared with the mainstream bicubic, RDN and Erdn [23] reconstruction methods to compare the texture differences between PSNR, SIM data and the actual reconstructed image.

The first chapter summarizes the subject, and explains the research problems, objectives and structure of the subject in detail.

The second chapter is literature review, which combs and analyzes the knowledge base related to the subject, learns from the previous experience, absorbs the advantages, summarizes the shortcomings of development, thinks of methods to

improve and determines the research objectives. Learned basic knowledge and learning tools, and strengthened the improvement of subject research ability.

The third chapter describes the research process, survey design, research methods and experimental design of this subject.

The fourth chapter describes the network structure of this subject in detail, and introduces the working process of the network, the principle and effect of module design. The overall steps and functions of establishing neural network model are also explained.

The fifth chapter introduces the experimental training process. Set the network layer number, learning rate, optimization algorithm and other parameters, and use the appropriate loss function. Finally, the training model is trained and the graphical loss diagram is established to visually express the training process. Then the effectiveness of the model is tested and compared with other mainstream models to prove the effectiveness of the network.

The sixth chapter summarizes the subject, and explains the aspects that can be improved or other references.

Finally, it reflects on the research of this subject, and explains the difficulties and solutions encountered in the process of this research.

2 RESEARCH AND LITERATURE REVIEW

2.1 OVERVIEW OF IN-DEPTH LEARNING

Artificial intelligence includes machine learning, and deep learning is a new research

direction in the field of machine learning. In recent years, deep learning has developed rapidly and made a breakthrough in many applications such as speech recognition and computer vision. Its motivation is to establish a model to simulate the neural connection structure of human brain. When processing signals such as image, sound and text, the data features are described hierarchically through multiple transformation stages, and then the data interpretation is given. For example, the processing of visual system class signals is as follows: first detect the edge and initial shape, and then gradually form more complex visual shapes. Similarly, deep learning forms a more abstract high-level representation, attribute category or feature by combining low-level features, and gives the hierarchical feature representation of data.

The reason why deep learning is called "depth" is relative support vector machine (swam) and boosting In terms of stack learning methods such as maximum entropy method, among the patterns learned by deep learning, the number of levels P1 of nonlinear operation is more. Shallow learning relies on manual experience to extract sample features. After learning the network model, the single-layer features without hierarchical structure are obtained; while deep learning transforms the feature representation of samples in the original space to a new one by layer feature transformation of the original signal Another theoretical motivation of deep learning theory is that if a function can be expressed in a concise form with a layer structure, it can require an exponential number of parameters (relative to the input signal), and the generalization ability is insufficient.

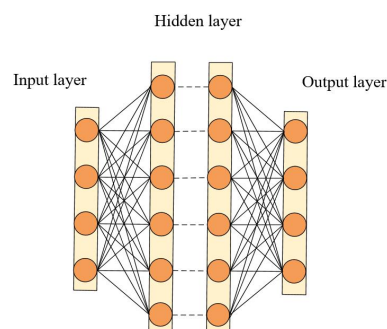


Fig. 1 structure diagram of full connection depth neural network

2.2 CONVOLUTIONAL NEURAL NETWORK

In recent years, convolutional neural network has achieved great success in large-scale image feature representation and classification. The landmark event was in the ImageNet[10], the deep convolutional neural network model implemented by krizhevsky reduced the error rate of image classification by nearly 50%[11]. In the famous go man-machine war in April 2016, he defeated Li Shishi by 4:1. The alphago artificial intelligence Go program adopts CNN + Monte Carlo search tree algorithm. Convolutional neural network was first proposed by Lecun et al. in 1998 for handwritten character image recognition. Its network structure is shown in Figure 2.

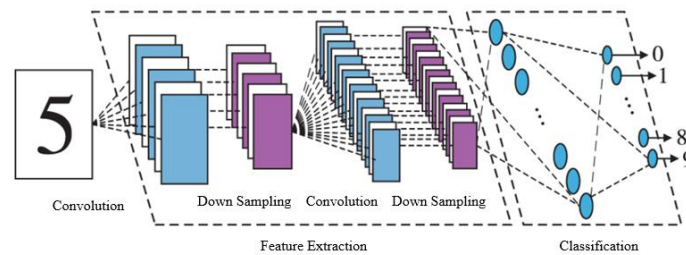


Fig. 2 structure of convolutional neural network

The input of the network is the original two-dimensional image. After several convolution layers and full connection layers, the prediction probability of the output image under each category is obtained. Each convolution layer contains three operations: convolution, nonlinear activation function, and maximum pooling.

In a convolution neural network, you need to learn a set of two-dimensional filter templates=,..., to convolute with the input feature map to get a two-dimensional feature map. The advantages of convolution are as follows: a) Two-dimensional convolution templates can better mine local relationships between adjacent pixels and the two-dimensional structure of the image; b) Compared with the fully connected structure in general neural networks, convolution networks greatly reduce the number of network parameters through weight sharing, making it possible to train large-scale networks; c) Convolution is robust to translation, rotation and scale transformations on images. After the convolution response characteristic graph is

obtained, a nonlinear activation function is usually required to obtain the activation response graph, such as sigmoid, tanh and ReLU functions.

Next, a max pooling or average pooling operation is applied to the activation function response graph. In this operation, the feature map is first divided into several spatial areas with a uniform grid, which can overlap, and then the average or maximum value of each image area is taken as the output. In addition, in maximum pooling, it is often necessary to record the location of the maximum output value. Previous research has proved that maximum pooling is better than average pooling in image feature extraction, so in recent years, researchers have basically adopted maximum pooling. Pooling has two main advantages: a) It enhances the robustness of the network to image transformations such as scaling, translation and rotation; b) Enables the high-level network to learn the higher-level structure of the image at a larger scale, while reducing network parameters, making large-scale network training feasible. Figure 3 below illustrates the difference between the extracted information features after two different processing methods.

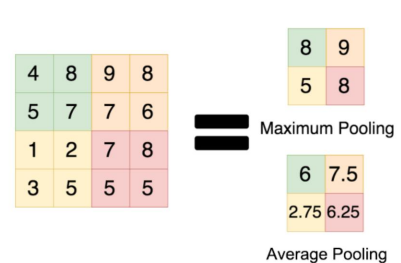


Fig. 3 average pooling operation diagram

Because of the large number of parameters of convolution neural network, it is easy to fit, which affects the final test performance. Researchers have made many improvements to overcome this problem. Hinton[12] proposed an optimization technique called dropout, as shown in Figure 4. It prevents over-fitting by randomly ignoring half of the feature points in each training iteration and achieves certain results. Wan[13] further extended this idea by setting a subset of the network connection weights randomly selected at each iteration to zero in the training of the full connection layer, which makes each network update target a different network

structure and further improves the generalization of the model. There are also simple and effective engineering techniques, such as momentum, weight decay, and data enhancement.

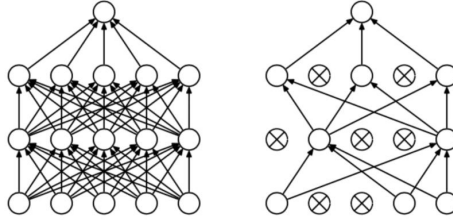


Fig. 4 dropout schematic diagram

2.3 SUPER-RESOLUTION RECONSTRUCTION

Image super-resolution reconstruction is to convert low resolution (LR) images into high resolution (HR) images in the same scene with a specific algorithm. This technology is one of the main technologies of image processing and computer vision. High resolution images are widely used in practical scenes because of their high pixel density and more detailed features. According to the number of input images, super-resolution reconstruction technology can be divided into single LR image generation HR image technology and multiple LR images generation HR image technology. This paper focuses on Single Image Super-Resolution, SISR. With its flexibility, simplicity and high practicability, this kind of technology has been widely used in the fields of image compression, medical imaging, remote sensing imaging, public security and so on.

In the past research work, the traditional single image super-resolution algorithm has achieved great success, but it has some limitations, because with the increase of magnification, the artificially defined prior information and the information provided by the model for high-resolution reconstruction become less and less, so that the algorithm can not achieve the purpose of reconstructing image high-frequency information. Traditional SISR methods are mainly divided into three categories: interpolation based super division algorithm (nearest neighbor interpolation, bicubic interpolation, etc.), although this kind of algorithm is simple, the reconstructed image has artifacts and oscillations; Modeling based super division algorithm (maximum a posteriori estimation method, iterative back projection algorithm, etc.) has better reconstruction effect than interpolation method, but the model operation efficiency is low and is greatly affected by amplification factor; Learning based super division algorithm (local embedding, sparse coding, etc.), which has the best reconstruction quality, is the mainstream research direction at present.

In the aspect of learning based super-resolution reconstruction of natural images, Dong[14] and others first proposed a network model SRCNN (Super-Resolution CNN) based on convolutional neural network, as shown in Figure 5. Firstly, the low

resolution image is preprocessed with bicubic, then the high-resolution image blocks are expressed by three-layer convolution fitting nonlinear mapping, and finally the high-resolution image blocks are aggregated to output the high-resolution image. The three-layer convolution correspondence method consists of three stages: shallow feature extraction, feature nonlinear mapping and final reconstruction. In this way, it can realize end-to-end mapping and implicitly learn the parameter information of convolution kernel, which makes it have better generalization ability and feature expression ability than the traditional methods without learning. However, when the network is too deep, it is difficult to make the network converge during training. In addition, when the network is too deep, the gradient will disappear.

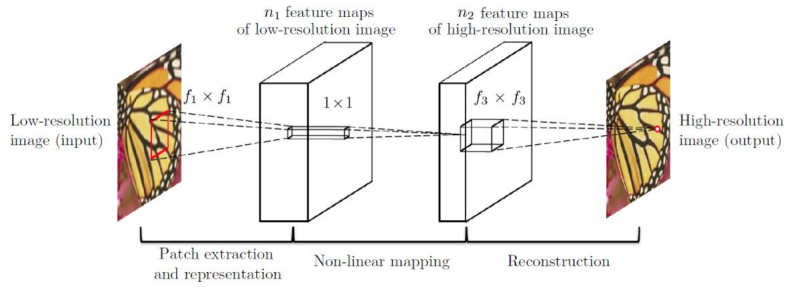


Fig. 5 SRCNN structure

Kim et al. proposed the algorithms of DRCN[15] and VDSR[16], as shown in Fig. 6 and Fig. 7. Based on SRCNN, the number of network layers is increased, the recursive neural network is used, and the idea of residual learning is used to introduce hop structure and adaptive gradient clipping into the network, which improves the problems of gradient disappearance and network expression ability degradation. The first three algorithms need to use bicubic interpolation to enlarge the input low-resolution image, and then convolution, which will lose some detail information in the process of feature extraction.

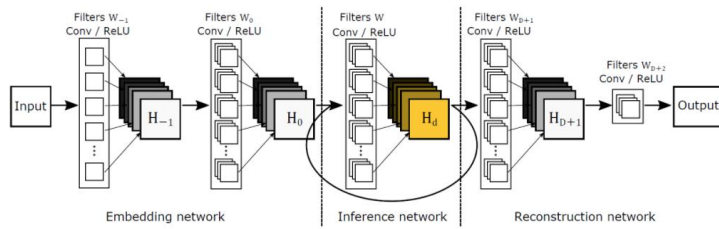


Fig. 6 VDSR structure diagram

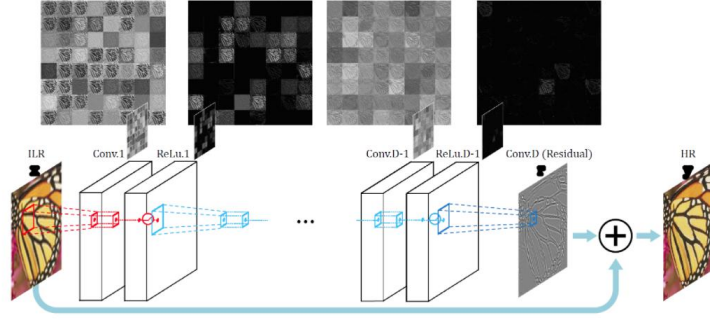


Fig. 7 DRCN structure

The FSRCNN algorithm proposed by Dong[17] and the sub-pixel convolution method ESPCN (as shown in Fig. 8) proposed by Shi[18] directly extract the features of low-resolution images, deconvolute at the end of the network, and reconstruct high-resolution images by recombining the channels of feature extraction. Experiments show that the reconstruction effect of ESPCN is better.

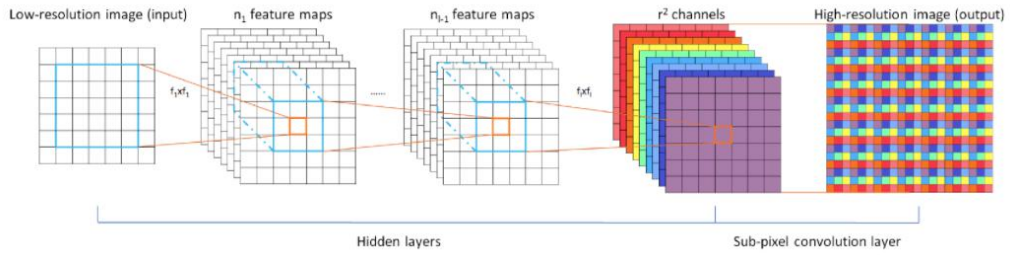


Fig. 8 ESPCN structure

In terms of deep learning network, He[19] and others proposed a residual network ResNet composed of multiple residual blocks (as shown in Fig.9), which effectively improves the problems of gradient disappearance and performance degradation during deep network training.

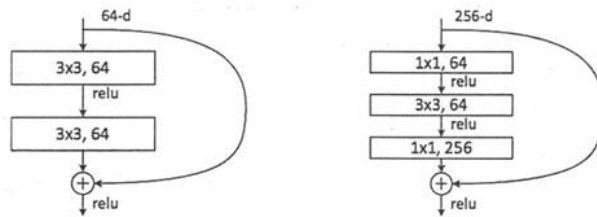


Fig. 9 two and three layers ResNet residual learning module

Ledding[20]. proposed SRResNet on the basis of ResNet. SRResNet introduces a large number of global and local hop structures in the network, which makes the feature information of each layer can be transmitted to the subsequent network layer, so that the features of all layers can be connected in series. This structure brings the advantages of reducing the gradient disappearance problem, strengthening shallow feature propagation, feature reuse and relatively reducing the number of network parameters to the whole network.

Lim[21] proposed EDSR network based on SRResNet, as shown in Fig.10 below. EDSR adopts a deeper number of network layers and removes the batch specification layer, which enables the network to extract more features under the same computing resources, and greatly improves the performance of the deep network.

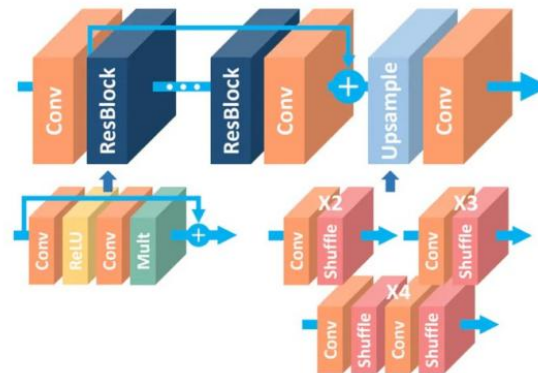


Fig. 10 EDSR structure diagram

In addition, DenseNet proposed by Huang[22] uses a more radical dense connection mechanism to connect all layers to each other, as shown in Fig. 11 below. Each layer accepts all the layers in front of it as additional input. Then the feature maps of different layers are spliced, which can achieve better feature reuse and improve the reconstruction efficiency.

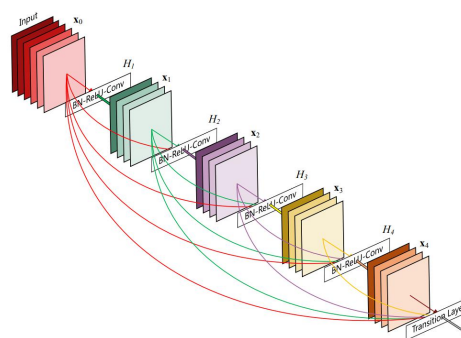


Fig. 11 DenseNet structure diagram

2.4 ATTENTION MECHANISM

The human visual system is essentially an attention mechanism. The brain focuses its limited attention on key information, which is conducive to saving body energy and quickly obtaining the most critical information. For an arbitrary image, human vision will first completely scan the whole image, focusing on some of the most prominent areas, that is, the focus of attention, and then focus on this key area to obtain more needed key information and suppress invalid information. This is a way of survival formed by human beings in the long river of historical evolution. Human visual attention mechanism can help us deal with survival affairs timely and effectively.

SE-Net[23] introduced attention mechanism into convolutional neural network for the first time. The structure diagram of attention mechanism module is shown in Fig.12 below. Its purpose is to directly establish the interdependence between characteristic channels. Specifically, it is to actively learn and retain important features, enhance and transfer important features to the deep network through weighting, and suppress secondary features. The corresponding weight is relatively small, which will gradually disappear in the process of network learning, so as to suppress invalid information.

The first is the compression operation. For a characteristic channel of $H \times W \times C$, it becomes the characteristic channel of $1 \times 1 \times C$ through a global average per pool operation. Each characteristic graph is compressed into a real number to represent the global information of the whole characteristic graph. Here, different channel numbers can be obtained by using different compression rates. Then through an activation function, add some nonlinear factors and filter some feature information, so as to learn a variety of features. In the whole compression stage, the overall number of channels is constant.

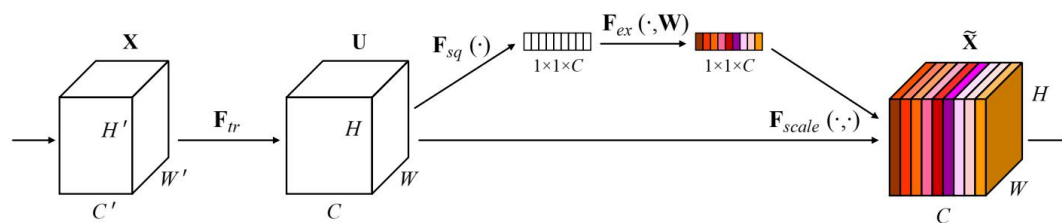


Fig. 12 SE-Net attention mechanism

The second is the expansion operation, through 1×1 number of convolution expansion channels. Parameter W is the corresponding weight of each feature channel generated. Parameter W is obtained through the training network to establish the correlation between feature channels.

The last step is to re label the weight, that is, the weight obtained after compression, which is generally called channel descriptor, and multiply it with the input feature map one by one. In a sense, the adaptive weighting of different feature maps helps to enhance important features, suppress secondary features, facilitate the network to learn better and retain important image information.

SK-Net[24] (shown in Figure 13) proposed a new noticeable mechanism in which each neuron automatically adjusts the size of the receptor field to the input information, using different scales of the convolution nucleus to control the size of the receptor field.

Each SK unit uses convolution kernels of different sizes to extract feature information, and then segments part of the feature. After fusion, it uses softmax for fusion, and finally re scales the weight to the previous feature channel. SK unit includes three

aspects: Split, Fuse and Select.

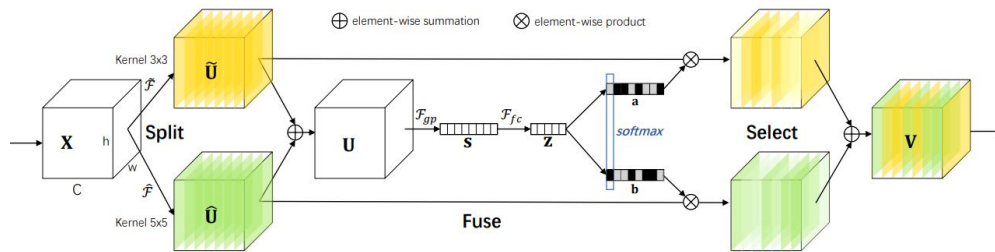


Fig. 13 SK-Net Attention Mechanism

In the segmentation stage, different size filters are used to extract the features of the input image.

In the fusion stage, the feature information from multiple branches is combined to obtain the recalibrated weight, which represents the global representation of the feature information.

In the selection stage, the feature maps of convolution kernels with different sizes are fused according to the recalibration weight.

RCAN[25] introduces attention mechanism into image super-resolution reconstruction for the first time and achieves good results. By considering the interdependence between channels, the network automatically changes the importance of features, selects important features and suppresses unimportant features, which is conducive to learning the main features of images and filtering redundant information.

CBAM[26] (Convolutional Block Attention Module) is a attention mechanism module that combines space and channel, as shown in Figure 14. This enables it to focus on and capture more high frequency information from different regions and dimensions.

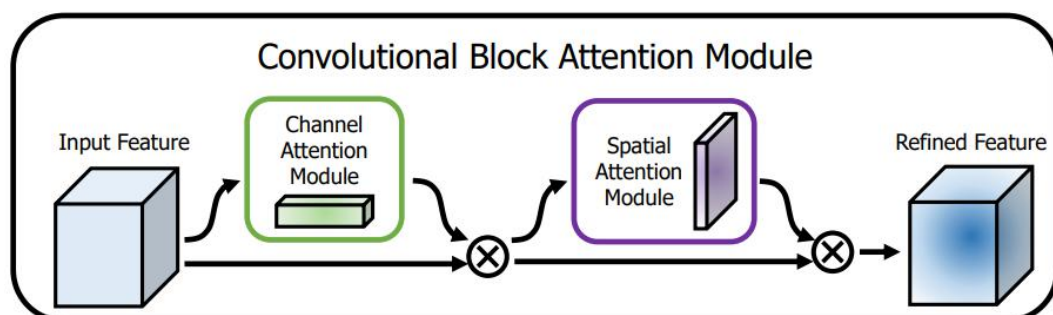


Fig. 14 CBAM

Lu[27] proposed - a multi feature channel network (CAMFFN), which adds an attention mechanism to the recursive module to give priority to important features,

and then recursively extract the image information in these important features. Experiments show that this method has achieved good results. Liu[28] a parallel network is proposed. The overall architecture is divided into two branches, one is the attention network for selecting important features, the other is the dense network for extracting complex features, and then the results of the two are fused to extract important and complex features. Experiments show that this method also achieves good visual effect.

2.5 LEARNING RATE

As an important hyperparameter in supervised learning and deep learning, learning rate determines whether and when the objective function converges to the local minimum. An appropriate learning rate can make the objective function converge to the local minimum in an appropriate time.

2.5.1 Discrete Staircase

For deep learning, the learning rate is halved for each t round of learning. For supervised learning, a large learning rate is initially set, and then the learning rate decreases with the increase of the number of iterations.

2.5.2 Exponential Decay

For deep learning, the learning rate decreases according to the difference of the growth index of the number of training rounds. For example:

$$\alpha = \frac{k}{\text{epoch_num}} \quad (1)$$

It can also be expressed as:

$$\alpha = \frac{k}{\sqrt{\text{epoch_num}}} \quad (2)$$

Where epoch_Num is the number of iteration cycles for the current epoch. However, the second method introduces another hyperparameter k.

2.5.3 Score Reduction

For in-depth learning, the learning rate changes according to the formula, and decay_rate controls the extent of reduction.

$$\alpha = \frac{\alpha}{1 + \text{decay_rate} * \text{epoch_num}} \quad (3)$$

2.6 ACTIVATION FUNCTION

2.6.1 What is an Activation Function

Activation function is a function running on the neurons of artificial neural network. It is equivalent to an intermediate interface and is responsible for the input-output mapping of information. It adds the nonlinear characteristics to the whole network, which can improve the feature expression ability of the network.

2.6.2 Type of Activation Function

Linear activation function: a simple linear function in the form of $f(x)=x$. basically, the input can be transferred to the output without any complex operation.

Nonlinear activation function: the neural network is used to realize complex functions, and the nonlinear activation function is used to separate nonlinear separable data, which determines the mapping from input to output, and can make the neural network approach any complex function. If there is no nonlinearity introduced by the activation function, the multilayer neural network is equivalent to the single-layer neural network.

In recent years, researchers have tried to use a variety of activation functions in experiments, but finding a suitable activation function to make neural network learning more efficient is still a research hotspot. Different types of nonlinear activation functions include Sigmoid, Tanh, ReLU, lReLU, PReLU, Swish and so on.

2.6.2.1 Sigmoid Activation Function

The following figure shows the sigmoid function and its derivative image. Sigmoid is also called logical activation function. It controls the value in the range of 0 to 1. When our ultimate goal is to predict probability, it can be applied to the output

layer. It can be seen from the figure that it can make large negative numbers approach 0 and large positive numbers approach 1. As follows:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (4)$$

The following figure 15 shows the Sigmoid function and its derivative image.

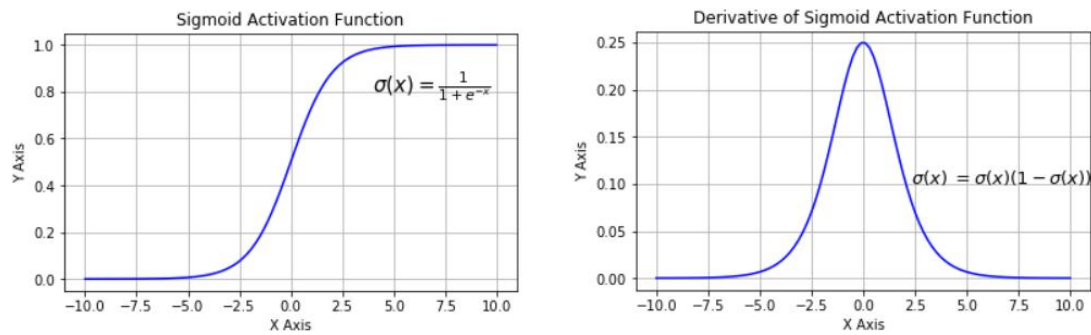


Fig. 15 Sigmoid Activation Function and Sigmoid Derivative

Sigmoid activation function also has four main disadvantages: 1) saturation of neurons leads to the disappearance of gradient. Sigmoid function is flat near 0 and 1, that is, when Sigmoid function performs data back propagation, when the output of neurons is close to 0 and 1, its gradient is close to 0. These neurons are called saturated neurons. In this way, the weights of these neurons cannot be updated, and it is also very difficult to update the weights of neurons adjacent to these neurons, which is called gradient disappearance. Therefore, when Sigmoid activation function is used in deep convolution neural network, a large number of saturated neurons will appear in the network, resulting in the disappearance of gradient, which will lead to the failure of data back propagation in the whole network. 2) Not zero mean. the output of Sigmoid is a function of non-zero bit center, which will lead to very inefficient gradient update. 3) Large amount of calculation. compared with other nonlinear activation functions, the amount of calculation of exponential function increases too fast, which will consume a lot of computing resources in deep network computing training. 4) Gradient explosion. When the x value approaches the left and right sides of 0, it will cause gradient explosion.

2.6.2.2 Tanh Activation Function

Tanh is also called hyperbolic tangent activation function. Like Sigmoid, Tanh also compresses a real value. However, Tanh is zero mean in the output range of - 1 to 1, and the Tanh function is like the fusion of two Sigmoids.

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (5)$$

$$\tanh(x) = 2\text{sigmoid}(2x) - 1 \quad (6)$$

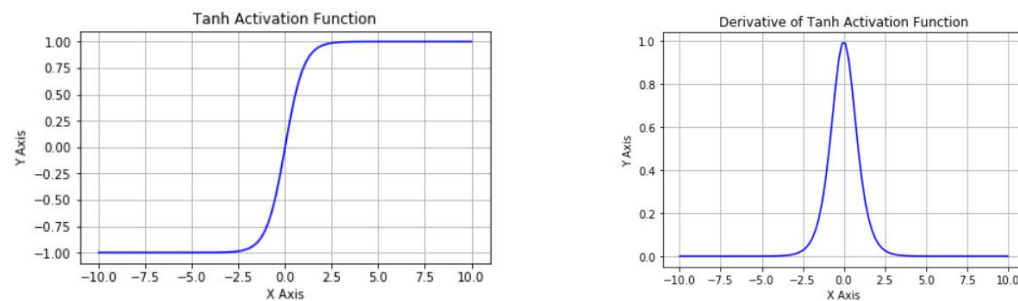


Fig. 16 Tanh Activation Function and Tanh Derivative

According to the comparison between Tanh and Sigmoid, the practical application effect of Tanh is better. This is mainly because the sigmoid function is in $[-1, 1]$ at the input. Once it approaches or exceeds the interval, it will be in a saturated state, which affects the accuracy of neural network prediction. Tanh is 0-means. The output and input of Tanh can maintain the nonlinear monotonic rise and fall relationship, which is in line with the gradient solution of BP network, has good fault tolerance, bounded function and conforms to the law of neural saturation. The only disadvantage of Tanh is that there are still the problems of gradient disappearance and explosion in Sigmoid and the large amount of exponential calculation.

2.6.2.3 Linear Rectifier Function (ReLU)

In order to solve the gradient vanishing problem, another nonlinear activation

function called linear rectifier function (ReLU) appears. It is better than the two activation functions mentioned before, and it is also the most widely used activation function today.

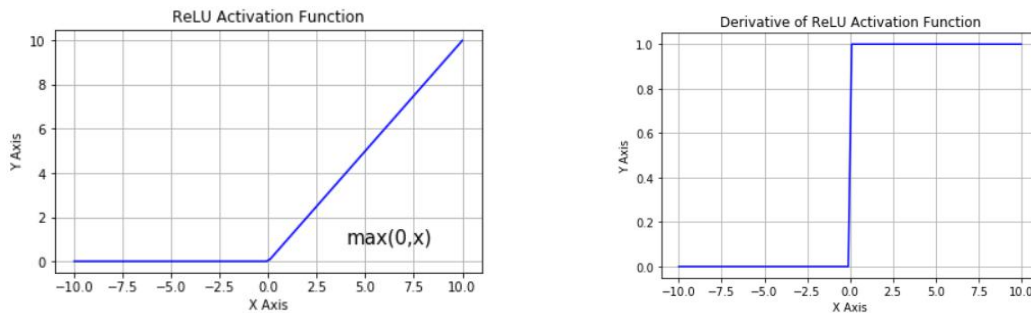


Fig. 17 ReLU Activation Function and ReLU Derivative

The activation function and its derivative are shown in the figure. The ReLU activation function is half corrected from the bottom. When $x \geq 0$, the gradient is 1, while when $x < 0$, the gradient is 0. Mathematically, it can be expressed by this simple expression:

$$f(z) = \max(0, z) \quad (7)$$

This activation function can make the network converge faster and always in the positive region ($x > 0$), which can resist the problem of gradient disappearance. In addition, compared with Sigmoid, ReLU does not involve higher cost exponential operation due to sparsity and lower time and space complexity.

2.7 LOSS FUNCTION

Loss function is used to estimate the inconsistency between the predicted value $f(x)$ of your model and the real value Y . It is a non negative real value function, which is usually expressed by $L(Y, f(x))$. The smaller the loss function, the better the robustness of the model.

2.7.1 Mean Square Error

Mean square error loss is also a common loss function. Mean Square Error (MSE) measures the mean square of the difference between the predicted value and the

actual observed value. The formula is as follows:

$$S_{MSE} = \frac{\sum_{i=1}^n (y_i - \hat{f}(x_i))^2}{n} \quad (8)$$

It only considers the average size of the error, not the direction. However, due to the square, the predicted value with more deviation from the real value will be easier to be eliminated than the predicted value with less deviation. In addition, each point of MSE is continuous and smooth, which is convenient for derivation and has a more stable solution, which makes it easier to calculate the gradient.

The following figure shows the curve distribution of mean square error value. It can be seen that when the input value of the function is far from the central value, the gradient is very large when using the gradient descent method, which may lead to gradient explosion.

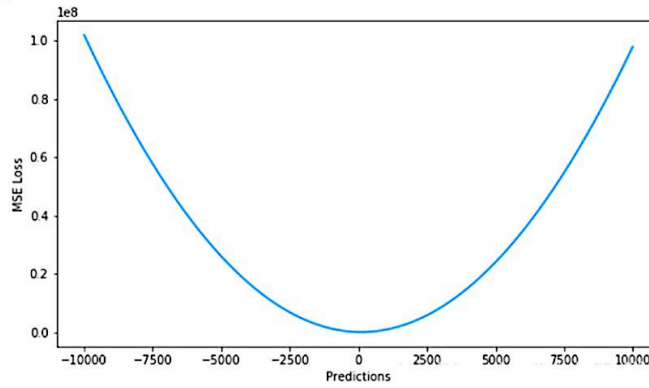


Fig. 18 Relationship between MSE loss (Y-axis) and prediction error (x-axis)

2.7.2 Average Absolute Error

The mean absolute error (MAE) is another commonly used regression loss function. The mean absolute error (MAE) measures the average value of the sum of the absolute differences between the predicted value and the actual observed value, and represents the average error amplitude of the predicted value, ranging from 0 to ∞ . Its formula is as follows:

$$S_{MAE} = \frac{\sum_{i=1}^n |Y_i - f(x_i)|}{n} \quad (9)$$

The average absolute error function curve is as follows:

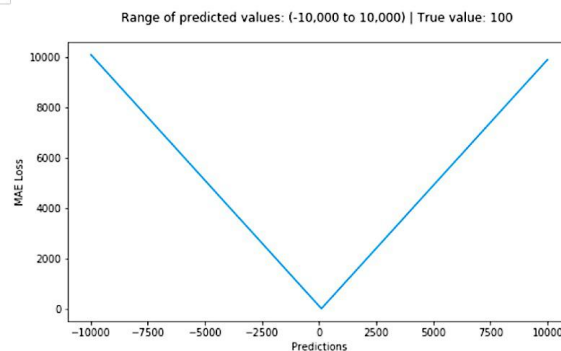


Fig. 19 Relationship between Mae loss (Y-axis) and prediction error (X-axis)

Like MSE, this measurement method does not need to consider the direction of error, but it is different from MSE in that MAE has a stable gradient for any input value, will not lead to gradient explosion, has a more robust solution and better convergence. The disadvantage is that when the loss is very small, L1 loss still adjusts the model with a fixed loss value, which leads to a small difference between the predicted value and the ground truth in the later stage of training, and the absolute value of the derivative of the loss to the predicted value is still 1. If the learning rate remains unchanged, the loss function will fluctuate near the stable value and it is difficult to continue to converge to achieve higher accuracy.

2.7.3 Mean Bias Error

It is very similar to MAE, except that the absolute value of the difference between the predicted value $f(x)$ and the real value Y is not used. The formula is shown in the figure below:

$$S_{MBE} = \frac{\sum_{i=1}^n (y_i - f(x_i))}{n} \quad (10)$$

2.7.4 Cross Entropy Loss / Negative Log Likelihood:

This is the most commonly used loss function in classification problems, and it is also one of the most commonly used models in neural network classification. The mathematical formula is as follows:

$$CrossEntropyLoss = -(y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)) \quad (11)$$

When the actual label is 1 ($y(i) = 1$), the second half of the function disappears, while when the actual label is 0 ($y(i) = 0$), the first half of the function disappears. In short, we simply multiply the logarithm of the actual prediction probability for the real value category.

2.8 GRADIENT DESCENT ALGORITHM

2.8.1 Algorithm Principle

Gradient Descent Optimization is the most famous and commonly used optimization algorithm for neural network models. Each of the latest deep learning libraries contains the implementation of various optimized gradient descent methods. For deep learning models, gradient descent algorithm is basically used for optimization training. The principle behind the gradient descent algorithm: the gradient $\nabla_{\theta} J(\theta)$ of the objective function $J(\theta)$ about the parameter updates the parameter in the opposite direction, which is the fastest rising direction of the objective function. For the minimization optimization problem, the objective function can be reduced only by advancing the parameters one step in the opposite direction of the gradient. This step is also called learning rate, that is, the learning rate η determines the size of the step. The parameter update formula is as follows:

$$\theta \leftarrow \theta - \eta \cdot \nabla_{\theta} J(\theta) \quad (12)$$

According to the amount of data used to calculate the objective function, the gradient descent algorithm can be divided into Batch Gradient Descent, Stochastic Gradient Descent and Mini-batch Gradient Descent. The difference between them is how much data we use when calculating the gradient of the objective function. According to the amount of data, we make a trade-off between the accuracy of parameter update and the time required in the update process.

For Batch Gradient Descent, its $J(\theta)$ is calculated on the whole training set. If the data set is relatively large, it may face the problem of insufficient memory, and its convergence speed is generally slow. Stochastic Gradient Descent is the other extreme. $J(\theta)$ is calculated for a training sample in the training set, also known as online learning, that is, once a sample is obtained, you can perform a parameter update. Therefore, its convergence speed will be faster, but there may be objective function value oscillation, because high-frequency parameter update leads to high square difference.

Mini-batch Gradient Descent is a compromise scheme. Finally, it combines the advantages of the above two methods and uses n (generally a multiple of 2, such as 32, 64, 128, etc.) small batch training samples for each update:

$$\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta; x^{(i:i+n)}; y^{(i:i+n)}) \quad (13)$$

This method can reduce the variance of parameter update, so as to ensure that the training process is more stable, and can quickly solve the small batch gradient by taking advantage of the matrix calculation in the relevant learning API libraries.

2.8.2 Gradient Descent Optimization Algorithm Adam

For the neural network model, the gradient can be calculated efficiently with the help of BP algorithm, which is very conducive to the training of neural network with gradient descent algorithm. An important parameter in the gradient descent algorithm is the learning rate, and the appropriate learning rate is very important: when the learning rate is too small, the convergence speed is slow, and when it is too large, it will lead to training oscillation and may diverge. The ideal gradient descent algorithm should meet two points: the convergence speed should be fast; And it can converge globally. The selection of different learning rates may have a great impact, as shown in Figure 20.

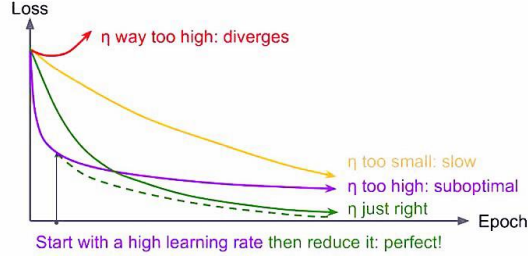


Fig. 20 Training effects of different learning rates

Adam [9] is an adaptive learning rate algorithm proposed by Kingma et al. In 2015. Adam calculates the adaptive learning rate for each parameter and adds an impulse term.

$$m \leftarrow \beta_1 \cdot m + (1 - \beta_1) \cdot \nabla J(\theta) \quad (14)$$

$$s \leftarrow \beta_2 \cdot s + (1 - \beta_2) \cdot \nabla J(\theta) \otimes \nabla J(\theta) \quad (15)$$

$$m \leftarrow \frac{m}{1 - \beta_1^t} \quad (16)$$

$$s \leftarrow \frac{s}{1 - \beta_2^t} \quad (17)$$

$$\theta \leftarrow \theta - \frac{\eta}{\sqrt{s + \varepsilon}} \otimes m \quad (18)$$

m and s are the estimates of the first-order moment (mean) and the second-order moment (uncertain variance) of the gradient, as the name of the algorithm. The initial value is generally set to 0, which may be small at the beginning of training. The third and fourth items are mainly to enlarge them. The last item is parameter update. The deviation is offset by calculating the first-order moment and second-order moment estimates of deviation correction, which can be approximated as an unbiased estimate of the expectation:

$$\hat{m} = \frac{m}{1 - \beta_1} \quad (19)$$

$$\hat{s} = \frac{s}{1 - \beta_2} \quad (20)$$

This generates Adam's update rules:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{s} + \varepsilon}} \hat{m} \quad (21)$$

The researchers suggest that the default value β_1 is 0.9 and β_2 is 0.999, ϵ is 10^{-8} .

Their experience shows that Adam performs well in practice. In pytorch, its implementation is as follows:

```
class torch.optim.Adam(params, lr=0.001, betas=(0.9, 0.999),  
eps=1e-08, weight_decay=0)
```

2.9 INTRODUCTION AND COMPARISON OF MAINSTREAM DEEP LEARNING FRAMEWORKS

2.9.1 Common Software Tools

At present, there are many software tools based on deep learning. Due to the different focus of each software tool, the appropriate deep learning architecture is adopted according to different needs, such as image processing, natural language processing or financial field. The following mainly introduces the current commonly used deep learning software tools.

- a) TensorFlow, a second-generation artificial intelligence system developed by Google based on DistBelief. The platform absorbs the advantages of existing platforms, not only allows users to touch the underlying data, but also has ready-made neural network modules, which can enable users to realize modeling very quickly. It is a very excellent cross-border platform. The software library adopts the data flow diagram mode to realize numerical calculation. The nodes in the flow diagram represent mathematical operations and the edges represent data arrays. The platform architecture developed based on the software library is flexible. The code can be run on single machine, mobile device or server without modification in one development. At the same time, it can support multi GPU / CPU parallel training.
- b) A deep learning abstraction platform based on Keras. It does not have the ability of bottom-level operation coordination, but relies on TensorFlow or Theano for bottom-level operation. Keras provides neural network module abstraction and

process optimization in training, which allows users to have convenient secondary development ability and add their favorite modules while rapid modeling.

- c) A functional platform for in-depth learning based on Caffe, Torch, MXNet and CNTK. This kind of platform provides complete basic modules to support the creation and training of fast neural network model; The disadvantage is that it is difficult for users to access these underlying computing modules.
- d) Theano, the earliest software platform in the field of deep learning, focuses on the underlying basic operations. The platform has the following characteristics: (a): the Python based scientific computing package integrating Numpy can be used in conjunction with the sparse matrix operation package SciPy, which is fully compatible with Numpy library functions; (b): It is easy to use GPU for acceleration and has a relatively large acceleration ratio than CPU; (c): Excellent reliability and speed advantages; (d): It can support dynamic C program generation; (e): With test and self inspection unit, it is convenient to detect and diagnose multiple types of errors.

Table 1 shows several commonly used software tools. It can be seen that there are many software tools based on deep learning and many corresponding programming languages. No programming platform or language can unify the Jianghu. It is believed that newer and more efficient programming languages or platforms may also appear in the future.

Table 1 Comparison of common software tools

Platform	Underlying Language	Operation Language
TensorFlow	C++, Python	C++, Python
Keras	Python	Python
Caffe	C++	C++, MATLAB, Python

Pytorch	C, Lua	Lua, C++
MXNet	C++, Python	C++,Python , Julia,Scala
CNTK	C++	C++ , Python
Theano	Python, C	Python

2.10 QUALITY EVALUATION STANDARDS

In practical application, only low-resolution degraded images can be obtained, and high-resolution images of real scenes can not be obtained, so the reconstruction results of the method can not be accurately evaluated. Therefore, in the research, the advantages and disadvantages of the reconstruction method are generally judged by the results of simulation experiments. In the image super-resolution reconstruction experiment, the truth high-resolution image is used as the label, and then the artificially selected degradation factors and parameters are used to degrade the label to obtain the low-resolution image. The purpose of image super-resolution reconstruction based on learning is to learn the nonlinear mapping relationship between low-resolution image Y and truth high-resolution image X , and reconstruct high-resolution image Z according to the learned model parameters. As shown in equation 2.3, $L(Z,X)$ represents the loss function of Z and X . The goal of super-resolution reconstruction is to make Z and X infinitely close and minimize the value of $L(Z,X)$, Improve the visual expression ability of the reconstructed image. That is to say, super fractional reconstruction is actually the inverse process of image degradation.

$$\theta = \arg \min L(Z, X) \quad (22)$$

In order to measure the effect of image super-resolution reconstruction, it is generally analyzed from two aspects: qualitative evaluation and quantitative evaluation. Qualitative evaluation is mainly to evaluate the reconstruction quality of image based on human visual perception, such as subjective evaluation in image edge details, high-frequency region and smoothness. Quantitative evaluation is an

objective evaluation through the calculation of quality evaluation index formula. The main evaluation indexes of super-resolution reconstruction include peak signal-to-noise ratio PSNR(Peak Signal to Noise Ratio) and structural similarity SSIM(Structural Similarity)[29].

2.10.1 PSNR value of peak signal-to-noise ratio

PSNR is one of the most widely used quality evaluation indexes for image super-resolution reconstruction. As shown in equation 2.4, MSE (mean square error) represents the mean square error of reconstructed image X and true value image Y , where H and W represent the height and width of the image. In the PSNR calculation formula of equation 2.5, n is the number of bits per pixel. When n is 8, the gray scale number of pixels is 256. PSNR calculates the difference between pixels. The greater the PSNR value, the less the distortion of the reconstructed image compared with the true value image, and the better the reconstruction effect.

$$MSE = \frac{1}{H \times W} \sum_{i=1}^H \sum_{j=1}^W (X(i, j) - Y(i, j))^2 \quad (23)$$

$$PSNR = 10 \log_{10} \left(\frac{(2^n - 1)^2}{MSE} \right) \quad (24)$$

2.10.2 SSIM value of structural similarity

Structural similarity SSIM value is also one of the most widely used quality evaluation indicators for super-resolution reconstruction. As shown in equation 2.6, SSIM measures image similarity from three perspectives: brightness, contrast and structure.

$$l(X, Y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \quad (25)$$

$$c(X, Y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \quad (26)$$

$$s(X, Y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \quad (27)$$

Among them, μ_x 、 μ_y represents the mean value of image X and Y respectively, σ_x 、 σ_y represents the variance of images X and Y respectively, σ_{xy} represents the covariance of images X and Y, C_1 , C_2 , and C_3 are constants, usually $C_1=(K_1*L)^2$, $C_2=(K_2*L)^2$, $C_3=C_2/2$, $C_3=C_2/2$, generally $K_1 = 0.01$, $K_2 = 0.03$, $L= 255$, that is:

$$\mu_x = \frac{1}{H \times W} \sum_{i=1}^H \sum_{j=1}^W X(i, j) \quad (28)$$

$$\sigma_x^2 = \frac{1}{H \times W - 1} \sum_{i=1}^H \sum_{j=1}^W (X(i, j) - \mu_x)^2 \quad (29)$$

$$\sigma_{xy} = \frac{1}{H \times W - 1} \sum_{i=1}^H \sum_{j=1}^W (X(i, j) - \mu_x)(Y(i, j) - \mu_y) \quad (30)$$

Then, the SSIM value is calculated as shown in equation 2.10. The larger the SSIM value, the higher the structural similarity between the reconstructed image and the true value image.210 conclusion.

$$SSIM(X, Y) = l(X, Y) \cdot c(X, Y) \cdot s(X, Y) \quad (31)$$

2.11 CONCLUSION

In this chapter, the literature research and review expounds the theoretical support and research results in the field of super-resolution from three aspects: network structure, optimization methods and tools. By summarizing the knowledge of deep learning and convolution neural network, this paper expounds the significance and position of super-resolution reconstruction based on deep learning. Then compare the types of deep learning performance optimization methods, analyze the advantages and disadvantages, and find the network optimization method suitable for the network. Finally, the learning tools for building deep learning are introduced, and the underlying programming language and operating system of each tool are listed and analyzed in detail.

Scientific research is a process of continuous new exploration on the basis of

inheriting and learning from the research and discovery of others. Therefore, scientific research must consult literature. Reading literature can help us understand the historical accumulation and current development of domain knowledge, clarify the research direction and scope, and avoid meaningless duplication and waste. We can also learn from others' research ideas and research tools, which will be of great help to us.

3 RESEARCH METHODS

The research process is divided into three parts: research steps, research methods and project resources.

3.1 RESEARCH STEPS

Research questions need to be carried out step by step. The research process can be divided into five steps.

3.1.1 Define the Research Theme:

Generally, it is necessary to collect relevant information about the problem, have an overall understanding of the problem, and determine the research topic according to the problem to be solved. Determine the experimental significance of the research topic.

3.1.2 Define the Research Object:

Collect research contents related to the subject, including relevant high-level academic literature, materials and objects. Sort out the data, clarify the research objects and objectives, and put forward specific assumptions on the basis of existing literature.

3.1.3 Formulate Research Plan and Research Tools

Verify the proposed assumptions. Identify the information needed and the tools and platforms for data collection.

3.1.4 Analysis and Interpretation of Data

Determine the comparative evaluation criteria, quantitative and variables. Determine whether to use charts; How to count and analyze the research results, combined with relevant theories. Analyze how the research results are generated, whether there is generalization ability and reliability, and further determine the research significance and role of the research results.

3.1.5 Writing Papers:

The paper includes abstract (research background, purpose, process, results and conclusion), introduction, research design, experimental process, result analysis and discussion, references and annexes.

3.2 RESEARCH METHODS

The research methods used in this subject include:

- 1) Literature research methods: in order to have a clearer understanding of the research principle, development background, current situation and improvement direction of this network, it is necessary to investigate and sort out the relevant literature in this research direction.
- 2) Network structure analysis method: after analyzing the data of the research direction, it is necessary to determine the problems to be improved, analyze the overall network, improve the network structure according to the needs, realize method innovation, and improve the efficiency and performance of the network model.
- 3) Experimental methods: optimize the physical conditions, control the model training variables, use a reasonable network optimization algorithm to prevent network over fitting and gradient explosion, and strive to reduce the loss to the lowest stable value.
- 4) Comparative analysis method: test the trained optimal model. Firstly, ablation experiments are carried out to prove the effectiveness of the improved network method and the role of innovation. Then compare the experimental model with

other mainstream super-resolution methods for objective evaluation and subjective evaluation, and compare its accuracy and actual viewing effect, so as to prove the superiority of this network architecture.

3.3 PROJECT RESOURCES

3.3.1 Experimental Data

DIV2K data set and Set5 data set are required as training set and verification set. Test the generalization ability of Set5, Set14, BSD100, Urban100 and Manga109 data sets.

3.3.2 Hardware Resources

The hardware environment of the experiment is shown in Table 1.

Table 1 Experimental environment parameters

Hardwareconfiguration	Parameter
CPU	Intel(R) Core(TM)i7-4710MQ CPU @2.50GHz
RAM	8G
GPU	RTX3080
GPU Memory	10G
Development Framework	Pytorch 1.7.6

3.3.3 Software Resources

1. Install Anaconda and PyCharm development tools.
2. Use Python to program the network model and related training and test files in the PyTorch framework.
3. Install OpenCV, Numpy and other common API libraries.

3.3.4 model training

The neural network experiment was carried out by using PyTorch and PyCharm[30].

1. Data Preparation

Download DIV2K data set, cut, rotate and translate the data set with preprocessed

files, and generate .h5 files containing 3000 high-definition pictures.

2. Establish Model

Write model code and corresponding initialization file, training and generalization ability test file code in model file. Using TensorBoard to visually view the model structure diagram and training process. You can also save the training data locally, and use the Matplotlib provided in PyTorch for drawing and data analysis.

3. Training Model

To use CUDA for model training, you need to install the corresponding CUDA and cuDNN to support the graphics card for data calculation, and download the corresponding GPU version of PyTorch framework. Training parameters and optimization methods also need to be set.

4 Save Model

The experiment needs 800 epoch cycles for each training, and the model code needs to be modified. By comparing the loss, the "best model" is selected and saved to the local folder of the computer in .PTH type.

4 DESIGN OF NETWORK MODEL BASED ON RESIDUAL ATTENTION

4.1 NETWORK STRUCTURE

The proposed superresolution reconstruction network based on residual attention mechanism is shown in Fig 21. The whole network consists of shallow feature extraction, ERB residual learning, non-linear mapping, and up-sampling. The shallow feature extraction phase consists of two 3×3 Convolution layer, which is used to extract the information of residual features that are easily lost; There are eight cascaded DECB (Dense Enhanced Channel Attention Block) modules in the non-linear mapping phase. Global jump structures are added between and within each module to improve model complexity and training difficulty convergence. *Concat* combines channel characteristics of each module to obtain detailed feature

information. The ERB residual module consists of two Up-Down structures to obtain high frequency information that is easily lost during shallow feature extraction. The up-mapping module uses subpixel convolution to sample the feature information extracted from the network layer and rebuild the output high-resolution image.

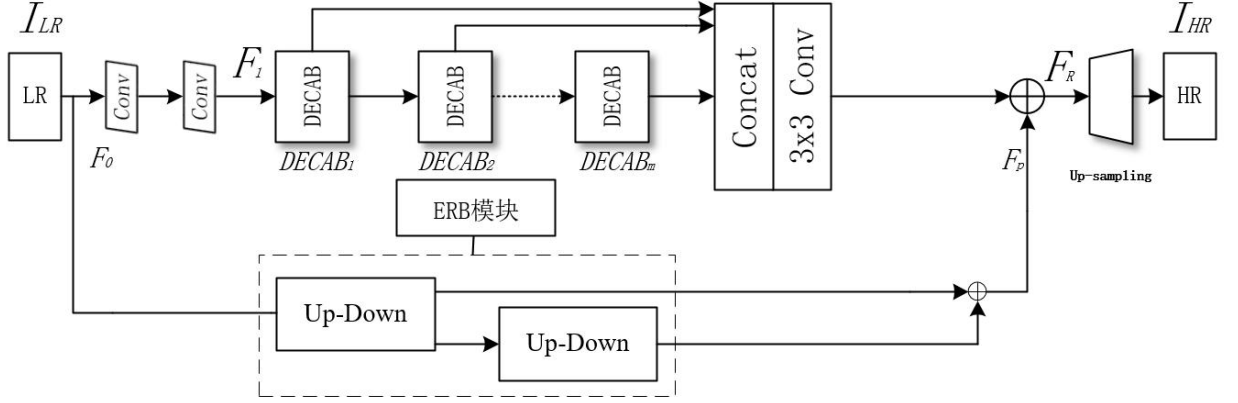


Fig. 21 overall network structure based on residual attention mechanism

First, input the low resolution image I_{LR} , and then perform the convolution operation to extract the shallow features of the image, that is:

$$F_1 = H_{SF}(I_{LR}) \quad (32)$$

HSF represents the convolution operation of shallow feature extraction to get a deeper feature representation, with the convolution layer parameter set to $3 \times 3 \times 2 \times 64$, two sizes 3×3 convolution kernels, 64 channels per convolution kernel, 1 step convolution operation, and 1 padding operation at the edge to keep the input and output image sizes consistent. The convoluted feature map is then processed through a cascade of channel attention mechanism modules to extract deeper, multidimensional features. In order to extract network features better, a designed DECAB module mechanism is added to the network layer. The output signal of F_1 is used as the input of DECAB. Suppose we have n DECABs, The signal F_n of the n th DECAB output can be expressed as:

$$F_n = H_{DECAB,n}(F_{D-1}) = H_{DECAB,n}(H_{DECAB,n-1}(\dots(H_{DECAB,1}(F_1)))) \quad (33)$$

Where, $H_{DECAB,n}$ represents the feature extraction operation of the n th DECAB.

After extracting local layer features using 8 cascaded DECAB, the feature information

obtained by them can be spliced together. 3×3 convolution can adaptively connect a series of features from different DECA-B layers. The finally extracted multi-dimensional image features F_{GF} are global features and local features, which are obtained by *Concat* splicing. This operation can be expressed as:

$$F_{GF} = f^{3 \times 3}(\text{Concat}[F_1, F_2, \dots, F_n]) = f^{3 \times 3}(\sum_{i=1}^c X_i * K_i + \sum_{i=1}^c Y_i * K_{i+c}) \quad (34)$$

Among them, $f^{3 \times 3}$ is convolution operation, *Concat* is signal connection operation, X_i and Y_i represent different characteristic channels, and K represents characteristic information of different layers.

F_p is the feature information extracted by F_0 through ERB residual learning module. The feature F_R is obtained after fusion. Finally, using the subpixel convolution method, the collected information feature F_R is up-sampled, and the low-resolution feature map is upgraded to the high-resolution output image I_{HR} .

$$I_{HR} = H_{UP}(F_R) = H_{all}(I_{LR}) \quad (35)$$

In Formula: H_{UP} represents an up-sampling operation, and H_{all} represents the entire operation on the input low-resolution image.

4.2 SHALLOW RESIDUAL FEATURE INFORMATION EXTRACTION

Although deep convolution neural network has brought good results on super-resolution performance, shallow network learning is limited, high-frequency information is under-learned, and deep network is under-fit for low-frequency information. The ERB (Embedded Residual Block) module proposed in this paper can extract more high frequency information of shallow features and improve the reconstruction accuracy.

The ERB module is composed of two sets of Up-Down convolution blocks designed, and the structure is shown in Figure 22. When feature extraction is performed, the input low-resolution feature map information A is sampled up to get the feature map

information B. At this time, B is divided into two parts, one part extracts the basic feature information from B through three convolution blocks to get C. The other part uses downsampling to map B to feature information D of the same size as the initial feature graph, and the difference Q between D and feature information A is the residual feature information that cannot be recovered at this time. The residual information Q is processed by the local residual learning stage and A+1 is used as the input data for the next set of Up-Down convolution blocks. These restored feature information of different frequencies are then fused. This enables lower-frequency information to be learned at the shallow network layer, and higher-frequency information to be learned at the deeper network layer by downsampling to the next module.

$$I'_x = H_{up}(I_x) \quad (36)$$

$$O_x = H_{up}(I'_x) \quad (37)$$

$$I_D = H_{down}(I'_x) \quad (38)$$

$$I_{x+1} = H_{RL}(I_x - H_{down}(I'_x)) \quad (39)$$

$$F_P = O_x + O_{x+1} + I_{x+2} \quad (40)$$

Where: H_{UP} represents up sampling, H_{down} represents down sampling, and H_{RL} represents the processing of residual learning stage. F_P represents the feature information extracted in the whole ERB residual learning stage.

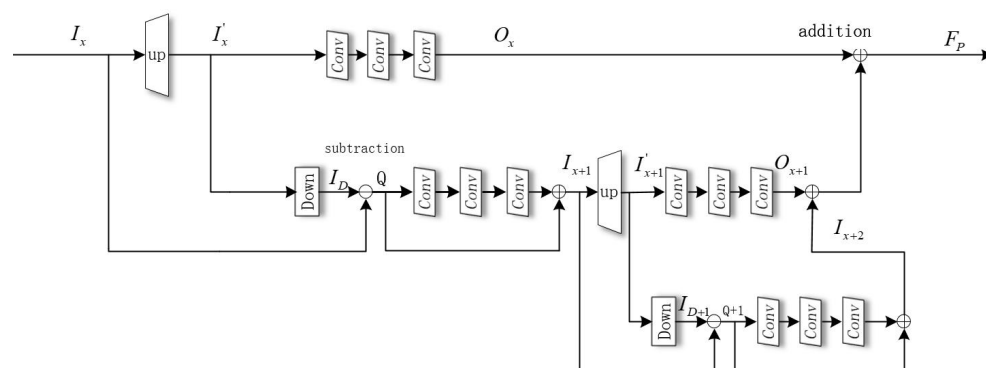


Fig. 22 ERB module structure diagram

4.3 FEATURE EXTRACTION

4.3.1 Introduction of Attention Mechanism

Nowadays, the reconstruction methods based on CNN all process channel feature information equally, ignoring the high frequency details outside the convolution area in ILR space. In each DECAB module, channel attention mechanism ECA-Net[31](shown in Fig. 23) can be added to adaptively determine the scope of local cross-channel information interaction without reducing dimensionality, and to re-scale the characteristics of each channel. This will help to use more computing resources to learn high performance and efficiency channel attention and suppress other useless information. Thus, the learning ability of feature discrimination of the network is enhanced.

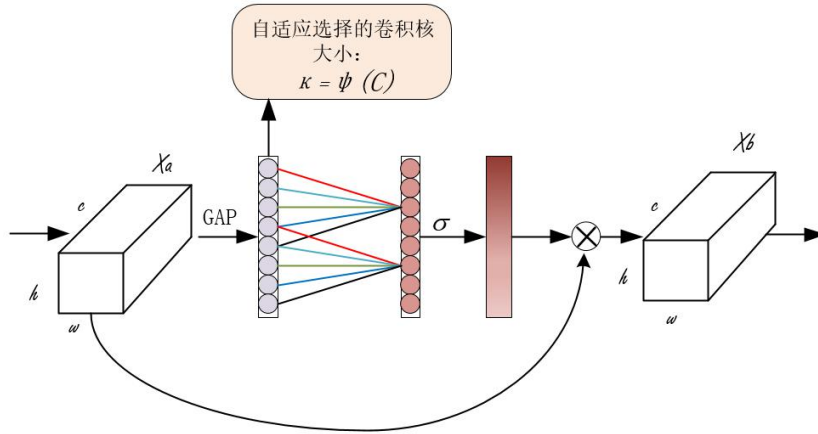


Fig. 23 ECA-Net structure

The dimensions of input and output characteristic graphs are $H \times W \times C$. H is the height of the characteristic graph, W is the width of the characteristic graph, and C is the number of channels of the characteristic graph; GAP is a global pooling operation. After using the global average pooling operation without dimensionality reduction, the ECA module first adaptively determines the core size k , and then executes 1×1 convolution, and then execute the Sigmoid function to learn the attention information of channels with different dimensions, as shown in equation (5).

$$\omega = \sigma(C1D_k(y)) \quad (41)$$

Used here ω To show the channel attention learned. δ Activate the function for ReLU. Where $C1d$ represents 1×1 convolution. Then, given the channel dimension C , the kernel size k can be adaptively determined by the following equation (6):

$$k = \psi(C) = \left\lfloor \frac{\log_2(C)}{\gamma} + \frac{b}{\gamma} \right\rfloor_{\text{odd}} \quad (42)$$

Where $|t|_{\text{odd}}$ represents the adjacent odd number of t . In this paper, all experiments γ and b are set to 2 and 1, respectively. When $k = 3$, ECA module can achieve the same effect as se-var3 [19], but it has lower model complexity. Therefore, this information interaction mode of adaptive sharing multi-dimensional channels ensures the model performance and reconstruction effect.

4.3.2 Introduction of Dense Connectivity and Residual

Mechanisms

In addition to introducing attention mechanisms in the DECAB module, jump structures and residual learning mechanisms are also added to the module. As shown in Figure 24. A large number of jumping structures connect the features input to each basic unit (two convolution layers, two ReLUs, and one ECA-Net), which enhances the transmission of feature information in the network and alleviates the problem of gradient disappearance. The *Concat* splicing operation splices each part of the channel feature information to get F_s , which achieves local feature fusion and guarantees the flow of information. Let F_d and F_{d+1} represent the input and output of DECAB, respectively, and the output of the m th convolution layer is as follows (7):

$$D_m = H_E(\delta(f^{3*3}(\text{Concat}[F_d, D_1, D_2, \dots, D_{m-1}])))) \quad (42)$$

f^{3*3} means Convolution layer representing 3×3 cores, δ Represents the ReLU activation function, and he represents the operation of ECA-Net mechanism.

After all mapped feature information is extracted, f^{1*1} convolution is used to

adaptively output feature information F_s , as shown in formula (8) below. The purpose of this is to reduce the number of feature maps, which can not only reduce the dimension and reduce the amount of calculation, but also integrate the features of each channel.

$$F_s = f^{1 \times 1}(\text{Concat}[F_d, D_1, D_2, \dots, D_m]) \quad (43)$$

In addition, in order to improve the network expression ability, the residual learning mechanism fuses the features of F_{d-1} and F_s to obtain F_d . As shown in equation (9):

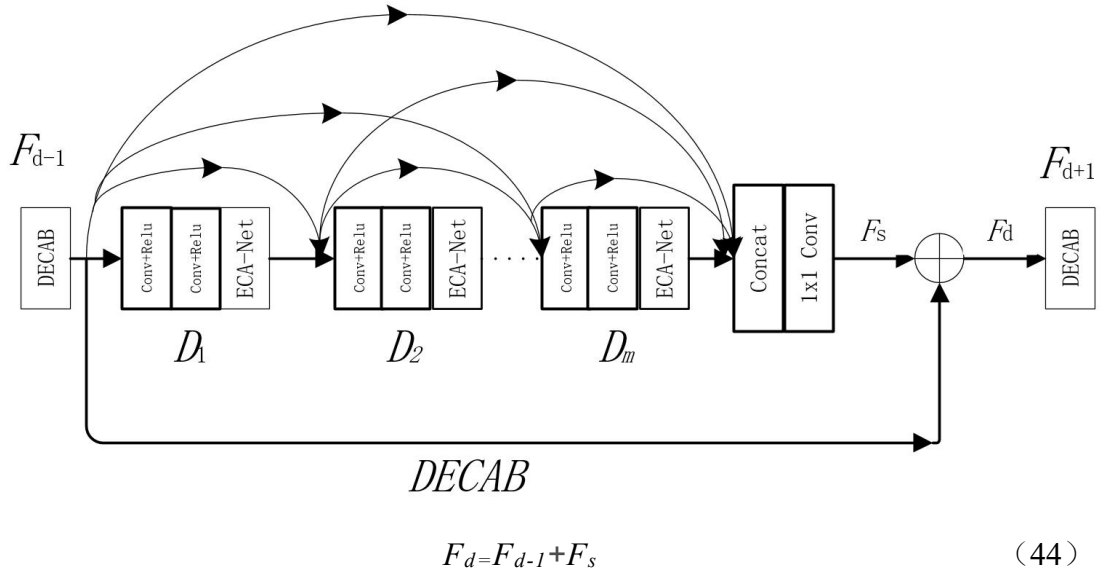


Fig. 24 DECAB module structure diagram

4.4 UP-SAMPLING

The sub-pixel convolution method is used for up sampling to promote the final low-resolution feature map to high-resolution output. As shown in formula (4-5).

In this way, instead of using artificial lifting filters such as bilinear or bicubic sampler, the sub-pixel convolution layer can learn more complex lifting operations through training, and the overall calculation time is reduced. Suppose we want to enlarge a color image $r \times r$ times, and then through three convolution layers, a characteristic diagram of $3r^2$ channel is obtained. Then, the characteristic diagram of $3r^2$ channel is rearranged into $r \times r \times 3$ area. In this way, The characteristic diagram of $3r^2$ size is

rearranged into $rH \times rW \times 3$ size high resolution image.

5 EXPERIMENT

5.1 EXPERIMENTAL PARAMETER SETTING AND TRAINING PROCESS

In order to train the network, low resolution and corresponding high-resolution images are required. The data set of DIV2K [25] contains 900 high-resolution natural images for training. The bicubic method is used to down sample the high-resolution image.

$$I_{LR} = (I_{HR}) \downarrow b_i \quad (45)$$

I_{LR} represents low resolution image, I_{HR} represents high resolution image, and $\downarrow$ represents bicubic down sampling operation. The obtained low resolution image is divided into 48×48 size block and its corresponding $64r \times 64r$ high resolution block, where r is the magnification factor. Finally, a '.h5' type file with picture information of DIV2K dataset corresponding to HR and LR is generated.

Batch-size learning is adopted in the training process. Due to the limitation of hardware resources, the number of batch size is set to 16, and the number of DECAB modules is set to 8, The number of 3×3 convolution cores and ECA-Net modules in each DECAB module are set to 16., and the ERB module uses two Up-Down convolution modules. Adam algorithm is used as the network optimization method, $\beta^1 = 0.9$, $\beta^2 = 0.99$, $\epsilon = 10^{-8}$. The initial learning rate is set to $10e^{-4}$. The training epoch is 800, and the L1 loss is set as the loss function. In the first 80% period, the learning rate does not change. In order to avoid over fitting, in the last 20% period, the learning rate becomes one tenth of its original value to ensure the training to the optimal state.

The hardware environment of the experiment is shown in Table 1.

Table 1 Experimental environment parameters

Hardwareconfiguration	Parameter
CPU	Intel(R) Core(TM)i7-4710MQ CPU @2.50GHz
RAM	8G
GPU	RTX3080
GPU Memory	10G
Development Framework	Pytorch 1.7.6

5.2 EXPERIMENTAL SIMULATION AND RESULT ANALYSIS:

After 800 iterative training of convolutional neural network, the loss curve of the network is shown in Fig. 5. In the initial stage of training, because the learning rate is large, the convergence speed is fast, the loss is quickly reduced, and the super-resolution reconstruction accuracy is rapidly improved. With the progress of the training cycle, in the last 20% period, the learning rate becomes 1 / 10 of its original value, and the network loss gradually tends to be stable and reaches the global minimum.

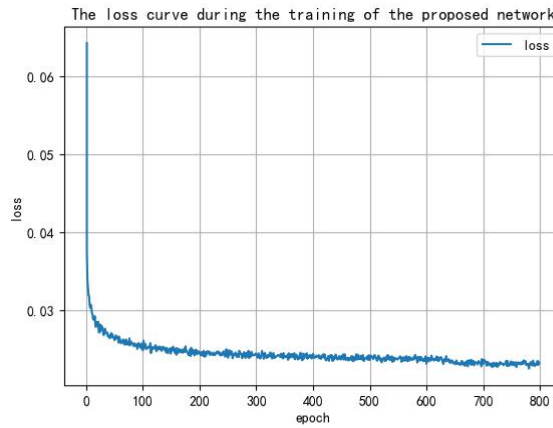


Fig. 25 Loss curve during network training

5.2.1 Objective Evaluation

5.2.1.1 Ablation Experiment

PSNR and SSIM are widely used objective indicators to evaluate image quality [26].

Generally, the higher the PSNR value, the closer the SSIM is to 1, indicating the better the image quality. For the test image, the bicubic method is used to down sample it. Then, these low resolution images are put into the super-resolution model to obtain the enlarged image. The value of PSNR is the difference between the original test image and the corresponding enlarged image.

In Table 2, Set5 dataset is used as the test set and PSNR is used as the criterion to evaluate the impact of different modules on the performance of the network model. The first row, represented by R1, shows that when RDN networks are used only, the average PSNR value for super-resolution images in Set5 is only 32.40dB. The second row, represented by R2, shows that the average PSNR of the super-resolution image in Set5 is 32.46 dB when using RDN networks and CBAM. The third row, represented by R3, shows that the average PSNR of super-resolution images in Set5 is 32.44dB when using RDN networks and ERBMs. In the fourth row represented by R4, the average PSNR of the super-resolution image in Set5 is 32.51dB when using RDN networks, ERBM, and DECAB. In the fifth row represented by R5, when using RDN networks and DECABs, the average PSNR of super-resolution images in Set5 is 32.46-32.47 dB.

Table 2 The Comparisons of PSNR(dB) for Different Network Structure in the Set5 Training Set
($\times 4$)

	Basic RDN	ERB	DECAB	Set5
R1	√	×	×	32.40
R2	√	√	×	32.44
R3	√	×	√	32.46
R4	√	√	√	32.51

By comparing the data of R1 and R2 and the data between R1 and R3, it can be seen that the corresponding PSNR values are increased by 0.07dB and 0.06db respectively, which shows that the performance of the model can be improved by using ERB module or DECAB module alone in RDN network. Comparing the data of R1 and R4, it can be seen that using ERB module or DECAB module at the same time in RDN network can improve the performance the most, and the corresponding PSNR value

is increased by 0.11dB.

5.2.1.2 Comparison with Other Algorithms

To prove the effectiveness of our proposed algorithm, we compare the proposed algorithm with the typical bicubic interpolation algorithm and the popular SR algorithm based on convolution neural network:RDN and ERDN. The magnification factor is $\times 4$, This means that the image is enlarged four times horizontally and vertically. The experimental results are shown in Table 3, where the test images are from classical super-resolution natural image data sets: Set5, Set14, BSD100, Urban100, and Manga109. Table 3 shows the PSNR/SSIM indicators for different images of different algorithm networks. In our proposed algorithm, both ERB module and DECAR module are added to the network model.

From the data in Table 3, we can see that our algorithm performs best among the PSNR/SSIM average indices of different test sets, which also indicates that our algorithm is superior to others.

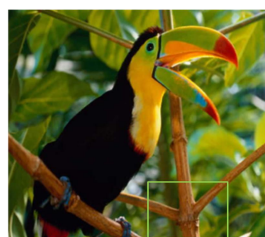
Table 3 Mainstream methods test results with Average PSNR / SSIM($\times 4$)

Dataset	Bicubic		RDN		ERDN		Ours	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Set5	26.69	0.790	32.39	0.879	32.47	0.879	32.51	0.880
Set14	22.61	0.605	28.88	0.765	28.85	0.763	28.93	0.770
BSD100	24.65	0.661	26.33	0.737	26.30	0.736	26.35	0.738
Urban100	21.70	0.652	24.86	0.790	24.70	0.786	24.95	0.793
Manga109	9.065	0.312	29.10	0.892	28.88	0.889	29.18	0.893

5.2.2 Subjective Evaluation

In addition, the subjective comparison of different algorithms is also carried out, and the results are shown in Figure 26. The five images on the left are from the above five different natural image test sets: Set5, Set14, BSD100, Urban100 and Manga109 data sets. Original intercepts the sub image for the details of the original image, and other images refer to the super-resolution reconstructed image of the original image,

which are generated by bicubic algorithm, RDN algorithm, ERDN algorithm and our proposed algorithm respectively. From these figures, it can be seen that the super-resolution image using bicubic algorithm is the most blurred image, followed by the super-resolution image using RDN. ERDN algorithm and our proposed algorithm have the best performance. However, if we observe carefully, the image edge using ERDN algorithm is relatively smooth and fuzzy, and there is not enough detail information. Our algorithm recovers more high-frequency information in texture and edge areas, less noise and closer to the original image. Therefore, the super-resolution image reconstructed by our algorithm performs best, and our algorithm is the most advanced algorithm.



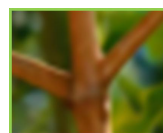
Set5



Original
PSNR/SSIM



Bicubic
32.48/0.860



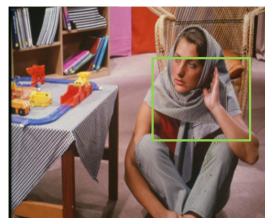
RDN
35.46/0.938



ERDN
36.11/0.935



Ours
35.57/0.937



Set14



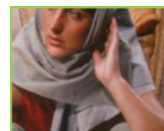
Original
PSNR/SSIM



Bicubic
22.61/0.605



RDN
26.24/0.745



ERDN
26.06/0.741



Ours
26.41/0.746



BSD100



Original
PSNR/SSIM



Bicubic
30.68/0.836



RDN
33.25/0.882



ERDN
33.25/0.882



Ours
33.32/0.884

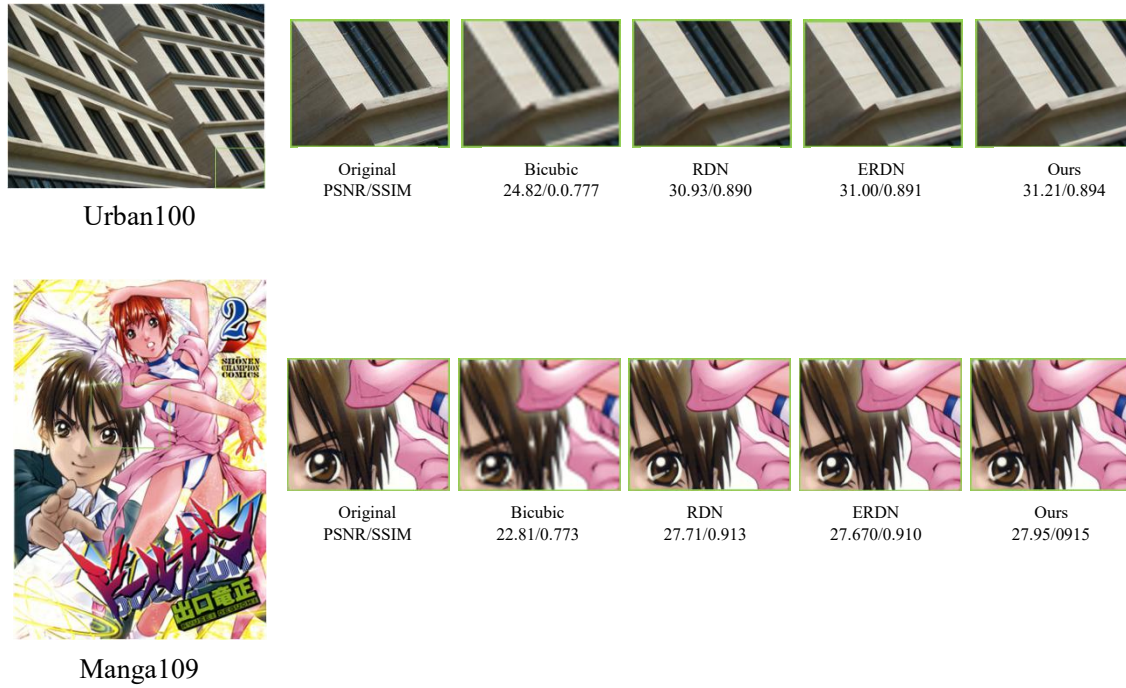


Fig. 26 super resolution image using different algorithms (× 4)

5.2.3 Comprehensive Evaluation

From the experimental data table 3 and the effect comparison of each super-resolution method Fig. 26, it can be seen that the single frame natural image super-resolution method based on residual channel attention network has higher objective indicators, more obvious detail features and better visual effect. The objective evaluation indexes (PSNR, SSIM) of this method on all test sets are higher than the traditional interpolation based methods and the mainstream methods based on deep learning RDN and ERDN; It can be seen from the effect comparison diagram of each super-resolution method in Fig. 6 that this method can reconstruct natural images with higher quality, more texture detail features and more natural and delicate image edges, with good visual effect, which is closest to the real high-resolution image.

In conclusion, the single frame natural image super-resolution method based on residual channel attention network is better than the traditional interpolation method and the current mainstream depth learning method.

6 CONCLUSION AND FUTURE WORK

In order to solve the problem of unclear image detail texture in natural image super-resolution reconstruction, a method based on residual attention network is proposed in this paper. In this method, the attention mechanism is used to adaptively determine the information interaction dimension and extract the multi-dimensional high-frequency feature information in the deep convolution network; In the shallow feature processing stage, the residual learning mechanism is used to extract the easily lost high-frequency residual information, which improves the performance of the model and further improves the problems of gradient disappearance and network feature loss. The experimental results show that among the mainstream super-resolution methods of deep convolution network, the model we constructed can achieve high reconstruction effect, the image edge is clear, the performance is improved by 0.5 ~ 2%, and has strong generalization ability. In our future work, we will adjust the number of convolution layers, dense spatial attention blocks and other structural optimization models to further improve the accuracy of the algorithm in practical application scenarios.

REFLECTION

After several months of research and experiment, the subject was finally completed. In the subject, the optimization and innovation of super-resolution reconstruction network is realized, and the model performance is improved. Through ablation experiments and subjective and objective comparisons with other mainstream models, it is proved that our network model is advanced and effective.

In the process of research, I gained a lot. From the initial access to a large number of literature to determine the research direction and experimental object, the support of experimental basic data, to the use of deep learning platform, the construction and performance test of network structure for countless times, and finally achieve

the research goal and write the article. Along the way, I made countless attempts. A lot of sweat and effort. Encountered many setbacks, but also improved my experimental ability.

Under the guidance of Professor Terry Walcott, I have a lot of knowledge about deep learning and super-resolution. From a little knowledge at the beginning to the later teacher recommended articles and materials to me, the work gradually got on track. I also consulted a large number of excellent papers in the industry on the academic website. All these are of great help to my project. Through the continuous communication with teachers and the improvement of the network, all the work is finally completed.

At first, I had no direction and only wanted to do in-depth learning, but after my teacher's guidance and consulting a large number of literature, I chose the super-resolution reconstruction I was familiar with and very interested in for research. After the related concepts of online learning and deep learning, a simple deep learning training is carried out based on MNIST data set, which is expressed as the simulation training of this subject.

I made a rigorous analysis on the selection of data sets. Div2k natural data set with high persuasion is selected as the training data set, set5 is selected as the verification data set, and finally five different types and mainstream data sets are selected as the comparison basis of different algorithms. During the experiment, I encountered some problems, such as program errors caused by inconsistent channel numbers, memory leakage caused by program problems, poor test picture effect caused by incorrect network structure, etc. After I continue to learn relevant knowledge and improve the model for experimental testing, I finally realize the improvement of the performance of the network model.

Finally, through this experiment, I learned the relevant knowledge of in-depth

learning. At the same time, I also had an in-depth understanding of the problems in the field of image super-resolution reconstruction, realized the network innovation, and successfully completed this experiment.

REFERENCE

- [1] F. Zhou, W. Yang and Q. Liao, "Interpolation-Based Image Super-Resolution Using Multisurface Fitting," in IEEE Transactions on Image Processing, vol. 21, no.7, pp. 3312-3318, July 2012, doi: 10.1109/TIP.2012.2189576.
- [2] BOYD S, PARIKH N, CHU E. Distributed optimization and statistical learning via the alternating direction method of multipliers[M]. Now Publishers Inc, 2011:11-22.
- [3] CHANG H, YEUNG D Y, XIONG Y. Super-resolution through neighbor embedding[C]// Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition:volume 1. IEEE, 2004: I-L.
- [4] HOEKS CM, BARENTSZ O, HAMBROCK T. Prostate cancer:multiparametric MR imaging for detection, localization, and staging[J]. Radiology, 2011,261(1): 46-66.
- [5] 任逢春,杨汉丰,杜勇.多参数磁共振成像在前列腺癌诊断中的应用[J]. 中华腔镜泌尿外科杂志（电子版）,2013,7(2):58-61.
- [6] UNLU E,TEMIZOZ O, ALBAYRAM S, et al. Contrast-enhanced MR 3D angiography in the assessment of brain AVMs [J]. European journal of radiology, 2006,60(3):367-378.
- [7] GRAICHEN H, BONEL H, STAMMBERGER T, et al. Subacromial space width changes during abduction and rotation-a 3D MR imaging study[J]. Surgical and Radiologic Anatomy,1999,21(1):59-64.
- [8] 苏衡, 周杰, 张志浩. 超分辨率图像重建方法综述[J]. 自动化学报, 2013, 39(008):1202-1213.
- [9] Agustsson E , Timofte R . NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study[C]// 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). IEEE, 2017.
- [10] Russakovsky O , Deng J , Su H , et al. ImageNet Large Scale Visual Recognition Challenge[J]. International Journal of Computer Vision, 2014:1-42.

- [11] MIT Technology Review. The 10 breakthrough technologies of 2013 [EB/OL]. (2013-04-23) .
[https:// www. technologyreview. com/s/513981/the-10-breakthrough-technologies-of-2013 /](https://www.technologyreview.com/s/513981/the-10-breakthrough-technologies-of-2013/).
- [12] Hinton G E , Srivastava N , Krizhevsky A , et al. Improving neural networks by preventing co-adaptation of feature detectors[J]. Computer Science, 2012, 3(4):págs. 212-223.
- [13] Wan L , Zeiler M D , Zhang S , et al. Regularization of Neural Networks using DropConnect[C]// International Conference on Machine Learning. 2013.
- [14] DONG C, LOY CC, HE K, et al. Learning a deep convolutional network for image super resolution[C]//European Conference on Computer Vision 2014. Springer International Publishing, 2014:184-199.
- [15] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
- [16] E. Agustsson, and R. N. Timofte, "2017 challenge on single image super-resolution: Dataset and study," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops., vol. 2017, pp. 126–135.
- [17] E. Agustsson, and R. N. Timofte, "2017 challenge on single image super-resolution: Dataset and study," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops., vol. 2017, pp. 126–135.
- [18] Shi W , Caballero J , F Huszár, et al. Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network[J]. IEEE, 2016.
- [19] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
- [20] Ledig C , Theis L , F Huszar, et al. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network[C]// IEEE Computer Society. IEEE Computer Society, 2016.
- [21] Lim B , Son S , Kim H , et al. Enhanced Deep Residual Networks for Single Image Super-Resolution[C]// 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). IEEE, 2017.
- [22] Huang G , Liu Z , Laurens V , et al. Densely Connected Convolutional Networks[C]// IEEE Computer Society. IEEE Computer Society, 2016.

- [23] Jie H , Li S , Gang S , et al. Squeeze-and-Excitation Networks[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017, PP(99).
- [24] Li X , Wang W , Hu X , et al. Selective Kernel Networks[C]// 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2020.
- [25] Zhang Y , Li K , Li K , et al. Image Super-Resolution Using Very Deep Residual Channel Attention Networks[J]. 2018.
- [26] Woo S , Park J , Lee J Y , et al. CBAM: Convolutional Block Attention Module[J]. Springer, Cham, 2018.
- [27] Lu Y , Zhou Y , Jiang Z , et al. Channel Attention and Multi-level Features Fusion for Single Image Super-Resolution[C]// 2018 IEEE Visual Communications and Image Processing (VCIP). IEEE, 2018.
- [28] Li X , Wang W , Hu X , et al. Selective Kernel Networks[C]// 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2020.
- [29] A. Horé and D. Ziou, "Image Quality Metrics: PSNR vs. SSIM," 2010 20th International Conference on Pattern Recognition, 2010, pp. 2366-2369, doi: 10.1109/ICPR.2010.579.
- [30] Ketkar N . Introduction to PyTorch[M]. 2017.
- [31] Wang Q , Wu B , Zhu P , et al. ECA-Net: Efficient Channel Attention for Deep Convolutional Neural Networks[J]. 2019.

APPENDICES

1.Main function

```
import torch
from torch import nn
import torch.nn as nn
import torchvision

class DenseLayer(nn.Module):
    def __init__(self, in_channels, out_channels, expansion =
4):
        super(DenseLayer, self).__init__()
        #self.expansion = expansion
```

```

        self.conv = nn.Conv2d(in_channels, out_channels,
kernel_size=3, padding=3 // 2)
        self.conv1 = nn.Conv2d(out_channels, out_channels,
kernel_size=3, padding=3 // 2)
        self.relu = nn.ReLU(inplace=True)
        self.eca = eca_layer(out_channels,k_size=3)
        self.cbam = CBAM(channel=out_channels )
    def forward(self, x):

        f = self.conv(x)
        g = self.relu(f)
        y = self.conv1(g)
        h = self.relu(y)
        #t = self.conv1(h)
        #r = self.relu(t)

        c = self.eca(h)
        #c = self.cbam(h)
        return torch.cat([x, c], 1)

```

```

class RDB(nn.Module):
    def __init__(self, in_channels, growth_rate, num_layers):
        super(RDB, self).__init__()
        self.layers = nn.Sequential(*[DenseLayer(in_channel
s + growth_rate * i, growth_rate) for i in range(num_layers)]
)

        # local feature fusion
        self.lff = nn.Conv2d(in_channels + growth_rate *
num_layers, growth_rate, kernel_size=1)

    def forward(self, x):
        return x + self.lff(self.layers(x)) # local r
esidual learning

```

```

class RDN(nn.Module):
    def __init__(self, scale_factor, num_channels, num_featur
es, growth_rate, num_blocks, num_layers,in_channels=64,out_channel
s=64):
        super(RDN, self).__init__()
        self.G0 = num_features
        self.G = growth_rate

```

```

        self.D = num_blocks
        self.C = num_layers

        # shallow feature extraction
        self.sfe1 = nn.Conv2d(num_channels, num_features,
kernel_size=3, padding=3 // 2)
        self.sfe2 = nn.Conv2d(num_features, num_features,
kernel_size=3, padding=3 // 2)

        # residual dense blocks
        self.rdb = nn.ModuleList([RDB(self.G0, self.G, s
elf.C)])

        for _ in range(self.D - 1):
            self.rdb.append(RDB(self.G, self.G, self.C)
)

        # global feature fusion
        self.gff = nn.Sequential(
            nn.Conv2d(self.G * self.D, self.G0, kernel_size=3,
padding=3 // 2),
            nn.Conv2d(self.G0, self.G0, kernel_size=3,
padding=3 // 2)
        )
        # 先进行上采样
        self.up = nn.ConvTranspose2d(in_channels=64, out_channels=64, kernel_size=8, stride=4, padding=2)
        self.dfg = nn.Conv2d(in_channels=64, out_channels=64, kernel_size=3, stride=4, padding=1)
        # 上采样特征进行重建
        self.sr_flow = nn.Sequential(*[
            nn.Conv2d(in_channels=64, out_channels=64, kernel_size=3, padding=1),
            nn.PReLU(out_channels),
            nn.Conv2d(in_channels=64, out_channels=64, kernel_size=3, padding=1),
            nn.PReLU(out_channels),
            nn.Conv2d(in_channels=64, out_channels=64, kernel_size=3, padding=1),
            nn.PReLU(out_channels)
        ])
        # 再进行下采样
        self.down = nn.Conv2d(in_channels=64, out_channels=64, kernel_size=8, stride=4, padding=2)
        # 残差进行重建

```

```

        self.bp_flow = nn.Sequential(*[
            nn.Conv2d(in_channels=64, out_channels=64,
kernel_size=3, padding=1),
            nn.PReLU(in_channels),
            nn.Conv2d(in_channels=64, out_channels=64,
kernel_size=3, padding=1),
            nn.PReLU(out_channels),
            nn.Conv2d(in_channels=64, out_channels=64,
kernel_size=3, padding=1),
            nn.PReLU(out_channels)
        ])
        self.eca = eca_layer(out_channels, k_size=3)

        # up-sampling
        assert 2 <= scale_factor <= 4
        if scale_factor == 2 or scale_factor == 4:
            self.upscale = []
            for _ in range(scale_factor // 2):
                self.upscale.extend([nn.Conv2d(self.G
0, self.G0 * (2 ** 2), kernel_size=3, padding=3 // 2),

                nn.PixelShuffle(2)])
            self.upscale = nn.Sequential(*self.upscale)
        else:
            self.upscale = nn.Sequential(
                nn.Conv2d(self.G0, self.G0 * (scal
e_factor ** 2), kernel_size=3, padding=3 // 2),
                nn.PixelShuffle(scale_factor)
            )

        self.output = nn.Conv2d(self.G0, num_channels, ke
rnel_size=3, padding=3 // 2)
        def forward(self, x):
            sfe1 = self.sfe1(x)
            sfe2 = self.sfe2(sfe1)

            x = sfe2
            local_features = []
            for i in range(self.D):
                x = self.rdfs[i](x)
                local_features.append(x)
            #bb = self.gff(torch.cat(local_features, 1)) +
sfe1
            x11 = self.gff(torch.cat(local_features, 1)) +

```

```

    sfel    # global residual learning

    up = self.up(sfel)    # torch.Size([16, 64, 128,
128])
    # ox = self.sr_flow(up)
    # jx = self.eca(ox)    # torch.Size([16, 64, 12
8, 128])
    # gt = self.dfg(jx)
    down = self.down(up)    # torch.Size([16, 64, 32,
32])

    sub = sfel - down
    ix = self.bp_flow(sub)
    ux = self.eca(ix)
    ux += sub
    up2 = self.up(ux)
    ox2 = self.sr_flow(up2)
    x2 = self.eca(ox2)    # torch.Size([16, 64, 128,
128])

    gt2 = self.dfg(x2)    #####
    #####
    down2 = self.down(up2)
    sub2 = ux - down2
    ix2 = self.bp_flow(sub2)
    ux2 = self.eca(ix2)
    ux2 += sub2    #####
    x = ux2 + gt2 + x11
    xx2 = self.upscale(x)
    x = self.output(xx2)
    return x

```

```

import torch
from torch import nn
from torch.nn.parameter import Parameter

```

```

class eca_layer(nn.Module):
    """Constructs a ECA module.

    Args:
        channel: Number of channels of the input featu
re map
        k_size: Adaptive selection of kernel size
    """

```

```

def __init__(self, out_channels, k_size=3):
    super(eca_layer, self).__init__()
    self.avg_pool = nn.AdaptiveAvgPool2d(1)
    self.conv = nn.Conv1d(1, 1, kernel_size=k_size,
padding=(k_size - 1) // 2, bias=False)

    self.sigmoid = nn.Sigmoid()

def forward(self, x):#torch.Size([16, 64, 128, 128])
    # x: input features with shape [b, c, h, w]
    b, c, h, w = x.size()

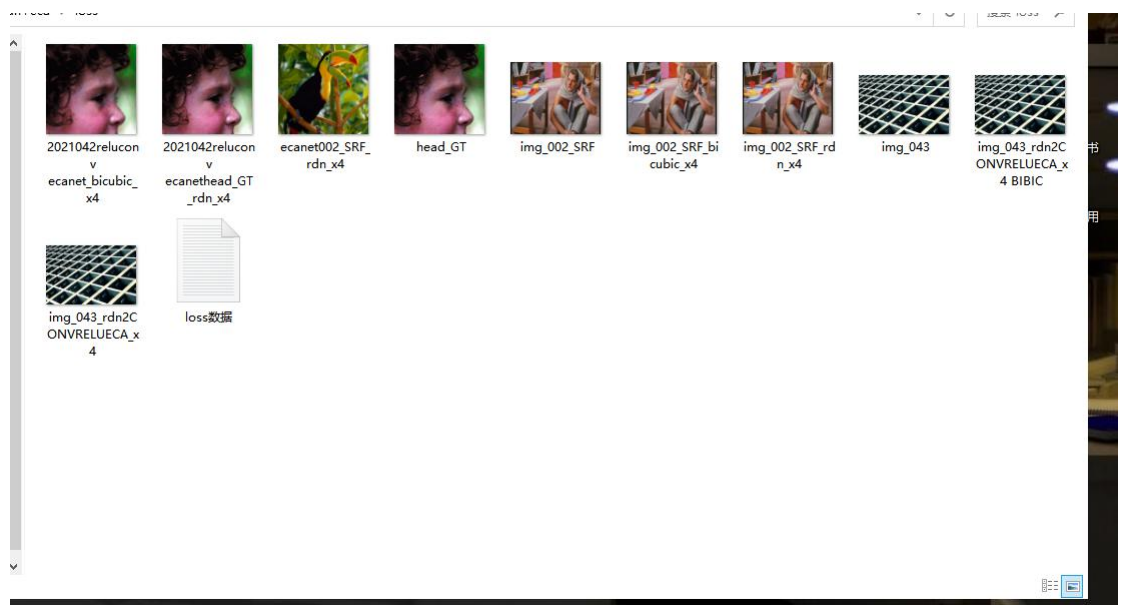
    # feature descriptor on the global spatial inf
ormation
    y = self.avg_pool(x)

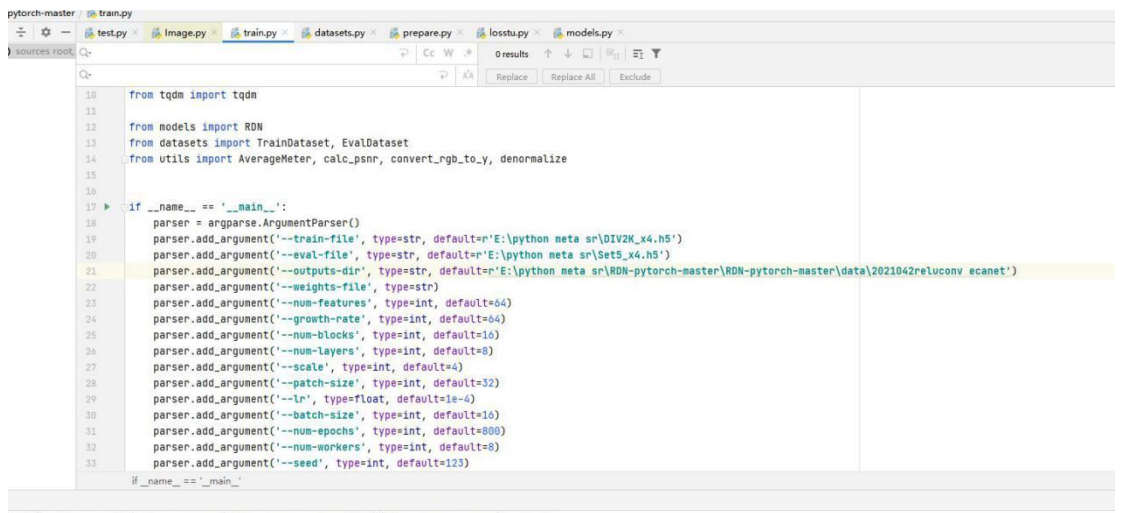
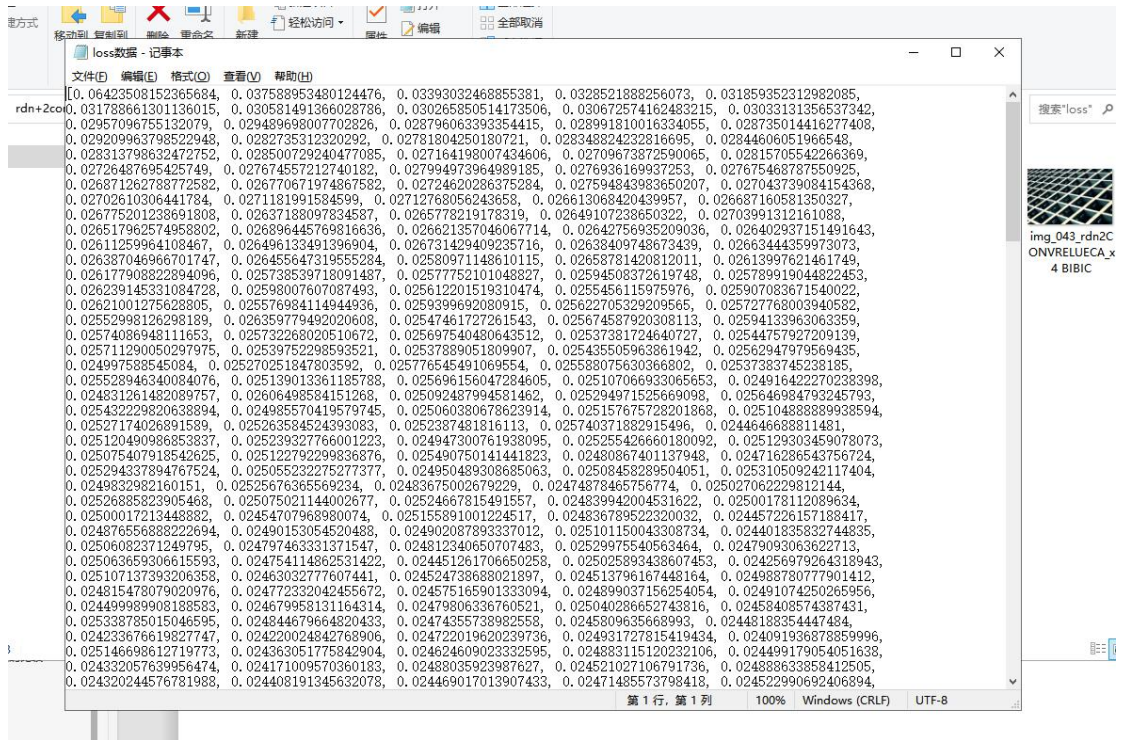
    # Two different branches of ECA module
    y = self.conv(y.squeeze(-1).transpose(-1, -2)).tra
nspose(-1, -2).unsqueeze(-1)

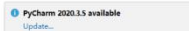
    # Multi-scale information fusion
    y = self.sigmoid(y)

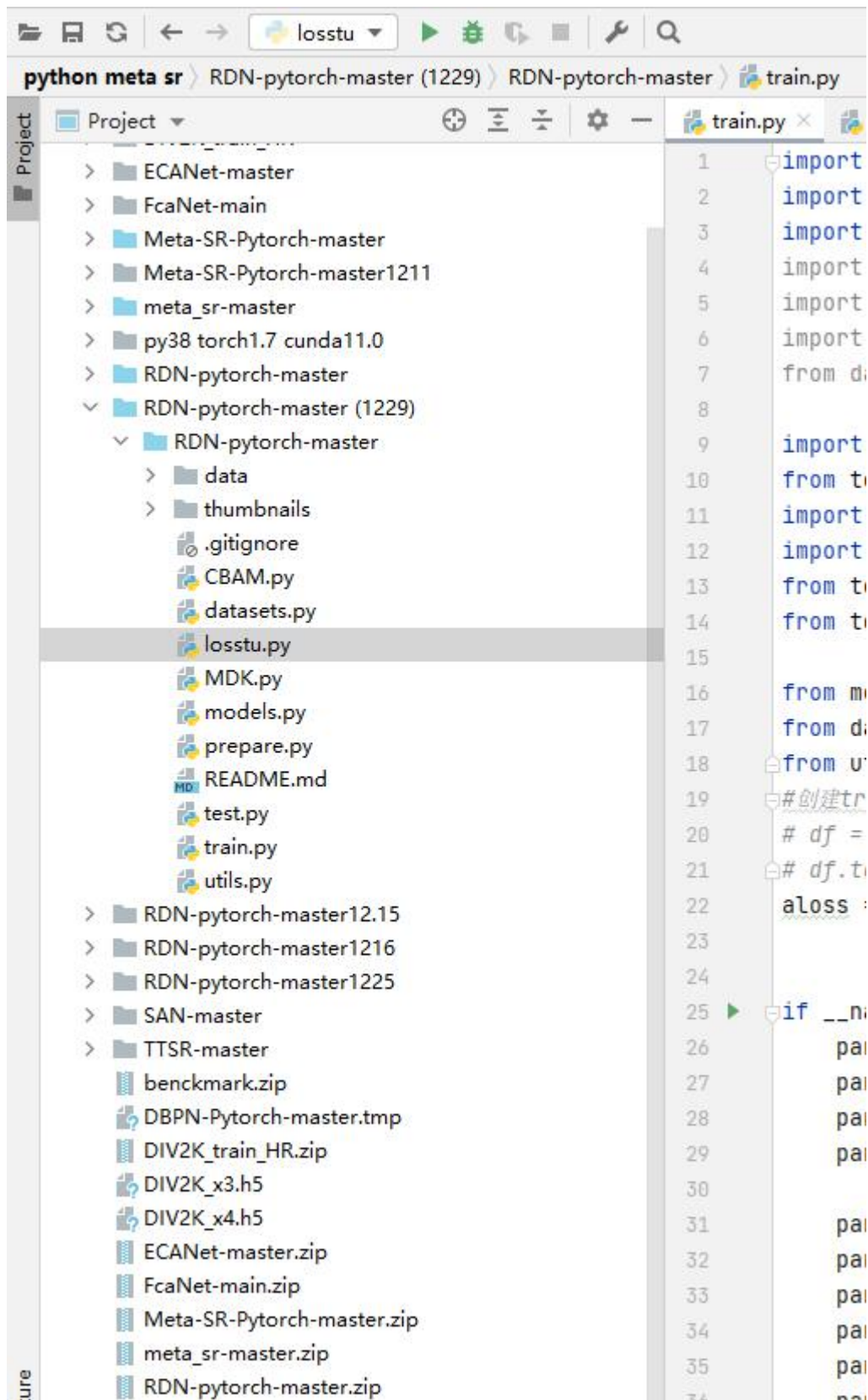
    return x * y.expand_as(x)

```









Project

ECANet-master

FraNet-main

Meta-SR-Pytorch-master

Meta-SR-Pytorch-master1211

meta_sr-master

py38_torch1.7_cudat11.0

RDN-pytorch-master

RDN-pytorch-master (1229)

RDN-pytorch-master

data

thumbnails

gignore

CDAK.py

datasets.py

loss.py

MDK.py

models.py

prepare.py

README.md

test.py

train.py

utils.py

RDN-pytorch-master12.15

RDN-pytorch-master1216

train (2)

models.py

MDK.py

loss.py

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 list1=[0.0580440082937479, 0.03708293368667364, 0.03316802141442895, 0.0318961488455534,
5         0.031173488385975362, 0.03032975560426712, 0.02981215112656355, 0.0290468265786767,
6         0.029388614624738692, 0.029205142438411713, 0.028658450834453106, 0.02808561236783862,
7         0.028687881093472242, 0.028332868069410325, 0.028092717383050878, 0.028375017419457438,
8         0.026329548943787813, 0.026237717397511084, 0.02655457566305995, 0.026795902312481404]
9
10 #list2=[2,3,4,5,8,9,2,1,3,4,5,2,4]
11 plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
12 plt.title("Loss曲线图")
13 plt.xlabel("epoch")
14 plt.ylabel("loss")
15 x=np.arange(0,len(list1))+1
16 x[0]=1
17 my_x_ticks = np.arange(1, 22, 1)
18 plt.xticks(my_x_ticks)
19 plt.plot(x, list1, label="rdn")#添加label设置图形名称
20 #plt.plot(x, list2, label="list2")#添加label设置图形名称
21 plt.legend()
22 plt.grid()
```

Run

train (2)

epoch: 796/799: 100% |██████████| 4000/4000 [01:14<00:00, 53.391t/s, loss=0.023479]
[0.06423508152365684, 0.037588953480124476, 0.03393032468855381, 0.0328521888256073, 0.031859352312982085, 0.031788661301136015, 0.030581491366028786, 0.030265850514173506, 0.030672574162483215, 0.030331313561
eval psnr: 32.42
epoch: 797/799: 100% |██████████| 4000/4000 [01:14<00:00, 53.421t/s, loss=0.022938]
[0.06423508152365684, 0.037588953480124476, 0.03393032468855381, 0.0328521888256073, 0.031859352312982085, 0.031788661301136015, 0.030581491366028786, 0.030265850514173506, 0.030672574162483215, 0.030331313561
eval psnr: 32.44
epoch: 798/799: 100% |██████████| 4000/4000 [01:15<00:00, 53.271t/s, loss=0.023364]
[0.06423508152365684, 0.037588953480124476, 0.03393032468855381, 0.0328521888256073, 0.031859352312982085, 0.031788661301136015, 0.030581491366028786, 0.030265850514173506, 0.030672574162483215, 0.030331313561
eval psnr: 32.43
epoch: 799/799: 100% |██████████| 4000/4000 [01:15<00:00, 52.831t/s, loss=0.023146]
[0.06423508152365684, 0.037588953480124476, 0.03393032468855381, 0.0328521888256073, 0.031859352312982085, 0.031788661301136015, 0.030581491366028786, 0.030265850514173506, 0.030672574162483215, 0.030331313561
eval psnr: 32.43
[0.06423508152365684, 0.037588953480124476, 0.03393032468855381, 0.0328521888256073, 0.031859352312982085, 0.031788661301136015, 0.030581491366028786, 0.030265850514173506, 0.030672574162483215, 0.030331313561
best epoch: 788, psnr: 32.46
Process finished with exit code 0

PyCharm 2020.3.5 availa