Image Classification and Its Applications on Insect Pest Recognition

劉 文傑

A Thesis submitted to Tokushima University in partial fulfillment of the requirements for the degree of Doctor of Philosophy

March, 2021

Tokushima University
Graduate School of Advanced Technology and Science
Information Science and Intelligent Systems

Table of Contents

List of Tables

List of Figures

Acknowledgement

During the COVID-19 pandemic, my PhD career will come to an end, and I have mixed feeling. I sincerely hope that the people around the world could stick together to achieve the victory over the COVID-19 at an early stage. Looking back on the past time, I am very much missing the time during the two year of study in Tokushima. Upon the completion of this thesis, I have to thanks those who have offered me encouragement and support for my study in Tokushima university.

First of all, the profound gratitude should go to my supervisor Professor Fuji Ren, who is the director of the Tokushima University Ren Laboratory, for his guidance. He helps me to select the research topic and gives a lot pertinent suggestions with his patience. With his great support, I could dedicate to complete the research on image classification tasks. He also inspires me to broaden my research interests, which will provide more choices for my research work in the future. Meanwhile, with his help, I was awarded the scholarship to pursue my study in Tokushima.

Secondly, my sincere gratitude also goes to Dr. Xin Kang and Asada. They helped me a lot in my daily life during the two years in Tokushima. Additionally, I need to give thanks to my friends at Ren Laboratory. Thanks to Duo Feng, Mengjia He, Siyuan Xue, Jiawen Deng, Yangyang Zhou, and other friends. We debated the problems together, which brought lots of benefits to me. Besides, we went to climb the mountain and enjoyed the time together, which gave me a deeply impression.

Finally, I need to thank my wife, my parents, and parents in law, for their continuous encouragement and support. With their endless love, I can successfully finish my PhD degree.

Abstract

In computer vision community, many excellent convolutional neural networks are proposed in recent years. Kinds of approaches are applied to improve the model performance, such as adding model depth, feature fusion, attention mechanism etc. However, how to effective extract and utilize the feature in the model is a critical problem for convolutional neural networks. In this thesis, we focused on constructing more effective feature extracting unit based on residual network for image classification, and we proposed three variant residual networks, including feature reuse residual network (FR-ResNet), deep feature fusion residual network (DFF-ResNet), and deep multi-branch fusion residual network (DMF-ResNet). Meanwhile, insect pests are regarded as the main thread to the commercially important corps. An effective classification method can avoid economic losses significantly. Earlier detection will help decrease agricultural losses. For the traditional classification method, it needs more experts to distinguish the categories of insect pests, which is expensive and low efficiency. As the deep learning method attracting more attention, this approach is applied in this domain. In our thesis, we applied our models to recognize the insect pest, which can achieve considerable improvement compared other convolutional neural networks.

Feature reuse is an effective method to improve the capability of model performance, and we also adopted this method in our model. Based on the original residual block, we combined feature from the input signal of a residual block and the residual signal together, which reuse the feature from the previous layer in a new and simple mode. Therefore, we named it a feature reuse residual block. In each block, it enhances the capacity of representation by learning half and reuse half feature. By stacking the feature reuse residual block, we obtained the feature reuse residual network (FR-ResNet) and evaluated the performance on several benchmark dataset, including CIFAR, SVHN, and IP102. The experimental results showed that FR-ResNet could achieve significant performance improvement in terms of image classification. Moreover, to demonstrate the adaptive of our approach, we applied it to various kinds of residual networks, including ResNet, Pre-ResNet, and WRN. The experimental results also showed that the performance could be improved obviously than original networks. Based on these experiments on several benchmark datasets, it demonstrates the effectiveness of our approach.

In FR-ResNet, it has a drawback that adding the model width will bring more parameters. Therefore, to obtain a good tradeoff between model performance and parameters, we modified the architecture of feature reuse residual block and proposed the feature fusion residual block. In each feature fusion residual block, we fused the feature from the previous layer between two 1×1 convolution layer in residual signal branch to extract more feature for our task. Meanwhile, we explored the contribution of each residual group to the entire model. We found that adding the number of residual blocks in earlier residual groups can promote the model performance significantly, which makes the model having a better capacity of generalization. Following the architecture of FR-ResNet, we construct the Deep Feature Fusion Residual Network (DFF-ResNet). Furthermore, we combined our approach with two common residual networks (Pre-ResNet and WRN) to prove the validness and adaptiveness of our method. Then, we validated these models on several benchmark datasets, including CIFAR and SVHN. The empirical experimental results indicate that our models have a better model performance than FR-ResNet and other state-of-the-art methods. Then, we apply our models in the field of recognizing insect pests, and we evaluate our models on IP102 dataset. Under the similar total number of model parameters, the DFF-ResNet surpasses FR-ResNet on several benchmark datasets with fewer parameters, respectively. Meanwhile, DFF-ResNet had better test accuracy performance than other state-of-the-art methods.

Activated by the proceding works, to learn the multi-scale representation to improve the model performance, we fused the extracted feature from three branches in each residual block. Specifically, in the new residual block, it contains three branches, including the basic branch, the bottleneck branch, and the branch from the input with linear conversion. Based on this structure, to further improve the model performance, we proposed the SFR module to recalibrate channel-wise feature responses and to model the relationship between these branches. The experimental results verified the effectiveness of our approach on CIFAR-10 and CIFAR-100 datasets. Even for extremely deep DMF-ResNet, our model can achieve compelling results. Compared to the baseline models and other state-of-the-art methods, our model can obtain the best model performance on IP102 dataset, which had proved the validness of our approach for the high-resolution image classification task. Through visualizing the highlighted regions on images, we can further explain the effect of our approach for the image classification task.

In this thesis, we proposed the FR-ResNet, DFF-ResNet, and DMF-ResNet end evaluated these models on several benchmark datasets. The experimental results demonstrated that our approaches can effectively improve the model performance. Besides, it also verified that our proposed models could extract more useful features for image classification tasks.

Keywords: Deep Learning, Residual Network, Feature Reuse Feature Fusion, Multi-branch Fusion, Insect Recognition, Image Classification

Chapter 1

Introduction

The emergence of deep learning technology has made breakthroughs in various fields, including natural language processing [1, 2, 3], emotion computing [4, 5, 6], especially for computer vision tasks, such as image classification [7, 8, 9, 10, 11, 12, 12, 13, 14, 15], object detection [16, 17, 18, 19], image segmentation [20, 21, 22, 23], facial expression recognition [24, 25]. Since LeNet [26] introduced the use of deep neural network architectures for computer vision tasks, the advanced architecture AlexNet [10] acquired ground-breaking victory at the ImageNet competition in 2012 by a large margin over traditional methods. Subsequently, many excellent neural networks, such as ZF-net [27], VGG [11], GoogleNet [12], Residual Networks [13, 14], and Inception Residual Networks [15], have been proposed and achieved better performance on ImageNet and other benchmark datasets. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2015, Residual Networks (ResNets) [13] win the 1st places on: ImageNet classification, detection, localization and COCO detection as well as segmentation tasks. The concept of shortcut connections inside a proposed residual unit for residual learning makes it possible to train much deeper network architectures. Then, an increasing number of deep residual network variants are emerged, which construct a family of deep residual networks. Our works in this thesis are also based on ResNet. Meanwhile, the development of deep learning is inseparable from convolutional neural networks.

The development of the convolutional neural network can be traced back to 1962, as Hubel and Wiesel [28] studied the visual system in the cat brain. In 1998, Yann LeCun proposed LeNet-5 [26], which adopted gradient-based learning applied to document recognition. Meanwhile, Back-Propagating (BP) algorithm [29] is applied to the training of neural network structure, forming the prototype of contemporary convolutional neural network. Until 2012, with the emergence of AlexNet [7], the neural networks attracted more attention. In the ImageNet Large Scale Visual Recognition (ILSVRC) 2012, the AlexNet drop the test error on ImageNet dataset from over 25% to 15%, which introduced a new deep structure and a dropout neural network method and overturned the field of image recognition. Since then, the convolutional neural network has gained great fame and developed rapidly. It is widely used in various fields, includiing autonomous driving, intelligent robots, and agriculture, and achieves the best performance in many problems. Thus, to evaluate the effectiveness of our approaches on real world problems, we choose to apply our proposed models in agricultural field to recognize insect pests.

1.1 Main Research Contents

The main research contents can be divided into following points:

1) To enrich the extracted feature for the classification task, we proposed a feature reuse residual block. By stacking the feature reuse residual block, we obtained the FR-ResNet. To verify the effectiveness of our approach, we evaluated our model on CIFAR-10/100, SVHN benchmark dataset and explored the properties of the model. Meanwhile, we applied our approach on Pre-ResNet and WRN to demonstrate the generalization of our method. The experimental results indicate that our approach can not only enhance model performance effectively but also be used in other networks.

2) To further enhance the capacity of FR-ResNet and address the drawback of FR-ResNet, we proposed the Deep Feature Fusion Residual Network (DFF-ResNet).

Then, we followed the experiment strategies in FR-ResNet to demonstrate the effectiveness the method. As the experimental results showed that DFF-ResNet can achieve better performance than FR-ResNet.

3) To fuse the feature from multi-scale branches, we proposed a new residual block containing three branches to extract more features for image classification task. Furthermore, we proposed a module and embedded it into the new residual block to recalibrate the channel-wise feature response and to model the relationship of the three branches. By stacking this kind of block, we constructed the Deep Multi-branch Fusion Residual Network (DMF-ResNet).

4) We applied the proposed models to recognize the insect pest and evaluated the model performance on IP102 datasets. The experimental results showed that our proposed approaches can achieve considerable improvement than the original ResNet with a similar total number of parameters or fewer parameters.

1.2 Thesis Organization

This thesis is divided into 7 chapters, which is organized as follows.

Chapter 2: Gives some background works including the structure of CNN, CNN learning algorithm, CNN training and inference process, image classification models, and the related datasets.

Chapter 3: Reviews some related works including deep convolutional neural networks, residual networks, feature fusion networks, attention mechanism and insect pest recognition.

Chapter 4: Introduces the feature reuse residual block, model optimization, the model properties, and the experimental results on several benchmark datasets.

Chapter 5: Introduces the feature fusion residual block, model optimization, the model properties, and the experimental results on several benchmark datasets.

Chapter 6: Introduces the multi-branch fusion residual block, model optimization, the model properties, and the experimental results on several benchmark datasets.

Chapter 7: Presents the conclusion and future works.

Chapter 2

Background

2.1 The Architecture of Convolutional Neural Network

Convolutional neural networks [30] are widely adopted in the computer vision tasks, including image classification, object detection, and segmentation. In this part, the components of a typical convolutional neural network are introduced, including the input layer, the convolutional layer, the pooling layer, the fully connection layer, the activation function, and the output layer. Fig. 2.1.1 shows an architecture of convolutional neural network.

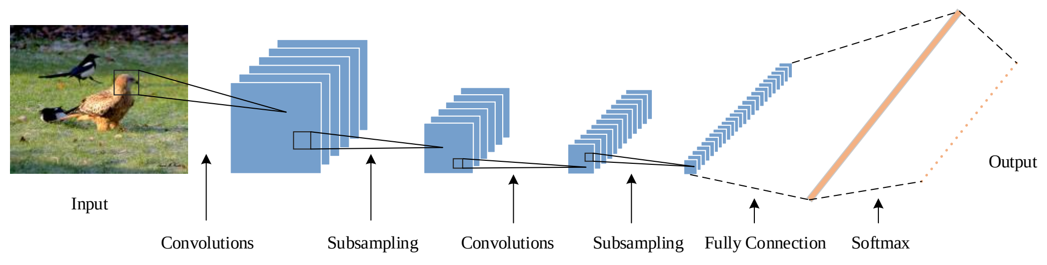


Fig. 2.1.1. The architecture of convolutional neural network.

Input Layer. The convolutional neural network is familiar to deal with euclidean spatial data, including images. For image classification tasks, the images are feed into the network. If you input the black and white image, it only has one channel. For color image, it has three channels for each pixels. In computer vision community, most of the open datasets contain the color images, which is same as the real world. In this thesis, we also focus on classifying color images.

Convolution Layer. Convolutional neural network is the first success learning method applied to deep neural network training. It utilizes the relationship between features to reduce the parameters to be learned, which improves the back-propagation training performance and meets the requirements of minimizing data preprocessing. In the convolutional layer, there are several critical properties, including local connectivity, spatial arrangement, and parameter sharing. Due to the sparse interactions in convolution layer, each node connects part of nodes in previous layer. The local connection is called receptive field. The Fig. 2.1.1 shows local connectivity between convolution layers. In each convolutional layer, the input of each neuron is connected to the local receptive field of the previous layer to extract the local features. When the feature is extracted, the corresponding positional relationship between other features is determined. Spatial arrangement determine the number of neurons in each convolutional layer, which is controlled by three hyperparameters: the depth, stride, and zero-padding. The depth of the output volume corresponds to the number of filters we would like to use, each learning to look for something different in the input. The stride corresponds to the step of filters moving distance on the feature map. The zero-padding controls the number of pixel to pad the input volume with zeros around the border. The nice feature of zero padding is that it will allow us to control the spatial size of the output volumes. Meanwhile, in each depth slice, the filters are constrained to use the same weights and bias, which is called parameter sharing scheme. This scheme is used to control the number of parameters in the convolutional layers.

Pooling Layer. The advantages of adopting pooling layer in ConvNet can be divided into several points. The first is used to reduce the spatial size of the representation. The second is used to decrease the number of parameters and computation. Meanwhile, it also can control overfitting. Therefore, it is common to periodically insert a pooling layer between successive convolutional layers in ConvNet architecture. Besides, the pooling layer reduce the model's dependency on location information. Therefore, the model will not be sensitive to object location and rotating for convolutional network. The other character is that pooling layer can increase the receptive field obviously. Convolutional layer has finite receptive field, which is hard to obtain the global input image information. Pooling layer can provide wider receptive field for the following convolutional layer. There are two common pooling operations in ConvNet architecture: max pooling and average pooling. Fig. 2.1.2 shows the max and average pooling. As the Fig. 2.1.2 shown, in max pooling layer, the max value in the sliding window is selected as the output. In average pooling layer, the values in the sliding window are averaged as the output. In each pooling layer, there are two hyperparameters: filter size and stride. As Fig. 2.1.2 illustrated, they determine the size of sliding window and the output volume.

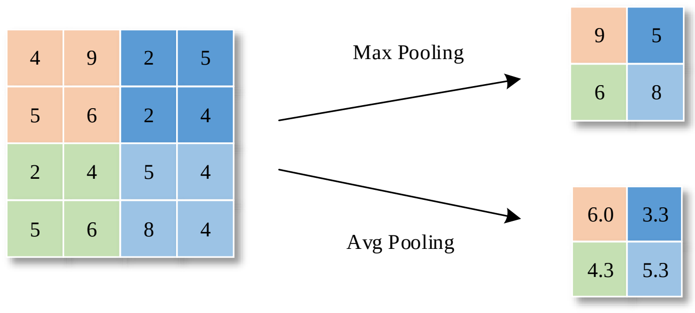
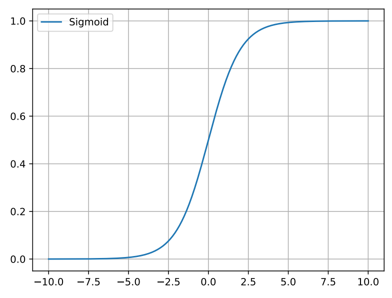
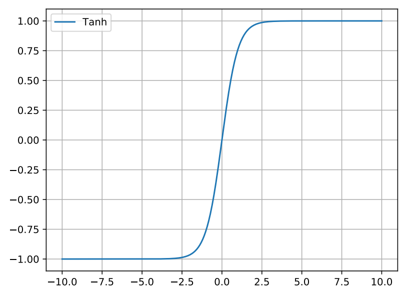


Fig. 2.1.2. Max pooling and average pooling.

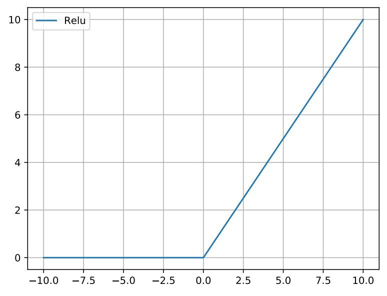
Fully Connected Layer. The fully connected layer is used to select the extracted feature from previous convolutional layers and output the optimal result. Before the fully connection layer, flatten or global pooling operation is performed, and all the information is converted into a one-dimension vector. This operation drops the spatial information, but it still can acquire the global input information as the increased receptive field in the previous layers and is convenient to output the result. In recent years, global pooling with one fully connection layer is commonly used in ConvNet designation.



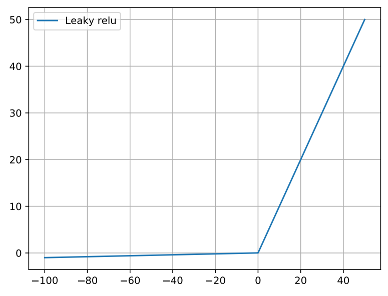
(a) Sigmoid



(b) Tanh



(c) ReLu



(d) Leaky ReLu

Fig. 2.1.3. The activation functions

Activation Function. The activation function plays a critical role in the convolutional network, which can improve the model non-linear representation ability. When the information pass through the convolutional layer, the output will be affected by the activation function. In the earlier period of CNN architecture design, the sigmoid and tanh functions are adopted. Fig. 2.1.3 (a) and (b) shows the activation functions. The formulations of these functions are represented as follows:

$$sigmoid(x) = \frac{1}{1+e^{-x}} \tag{2.1.1}$$

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.1.2}$$

The sigmoid and tanh activation functions have two advantages. One is that the input can be constraint between 0 to 1 or -1 to 1, which is convenient to calculate between the CNN layers. The other is that these two functions are differentiable in the whole interval, which is a necessary condition for back-propagation algorithm. However, they also have drawbacks. When the output value is too large or small, the gradient of the function is close to zero. This phenomenon is called gradient saturation, and it will lead to the update value closing to zero in the back-propagation process. Then, the network parameters are difficult to be updated. Secondly, they need more calculation leading to more time for model training, especially for the deep convolutional neural network. Therefore, they are gradually replaced by Rectified Linear Unit (ReLu) function, as shown in Fig. 2.1.3(c). The formula of the function is very simple.

$$ReLu(x) = max(0, x) \tag{2.1.3}$$

Meanwhile, there are also some works trying to improve the ReLu activation function, and the Leaky ReLu [31] and PReLu [32] are proposed by changing the slope. Fig. 2.1.3 (d) shows the Leaky ReLu activation function. The advantages of the improved activation functions can be summed up in the following two points. The first is that the gradient will not saturated, which can alleviate the problem of gradient diffusion and accelerate the parameter update rate. The second is that these activation functions do not need much accumulation compared with exponent arithmetic, which will save the training time for large neural networks.

Output Layer. Most of works adopt softmax function to output the predicted result for image classification model. The softmax layer will output the probability corresponding to the real category. The formula of softmax function is presented as follows:

$$softmax(x_i) = \frac{exp(x_i)}{\sum_j exp(x_j)} \tag{2.1.4}$$

In this section, we introduce the components in the ConvNet. Based on the specific tasks, we can modify these components to improve the model performance. In the next section, we will introduce the ConvNet learning algorithm.

2.2 Convolutional Neural Network Learning Algorithm

The learning algorithm of convolutional neural network comes from back-propagation algorithm. The significant difference of the two algorithm is that the neural nodes only connect part of nodes in the previous layer in ConvNet compared with fully connection in the back-propagation algorithm.

Convolution layers. At a convolution layer, the feature maps from previous layer are convolved with learnable kernels and pass through the activation function to form the output feature map. Therefore, the formula of convolutional layer in the l-th can be represented as follows:

$$x_j^l = f(\sum_{i \in M_j} x_i^{l-1} * k_{i,j}^l + b_j^l) \tag{2.2.1}$$

Here, x^l refers to the output of feature map in the l-th. M_j represents a selection of input maps. k^l refers to the convolutional kernels. f represents the activation function. $*$ represents the convolution operation, and b^l refers to the bias.

The gradient of convolution layer. In the back-propagation process, it updates the weight and bias by minimizing the residual error. To acquire the residual error, we should compute the sensitivity map δ corresponding to a block of pixels in the convolutional layer's output map. We assume that each convolution layer l is followed by a subsampling layer $l+1$, and δ_{l+1} represents the associated sensitivity map in the subsampling layer $l+1$. Thus, the sensitivity map in the l-th layer can be represented as follows:

$$\delta_j^l = \beta_j^{l+1}(f'(u_j^l) \circ up(\delta_j^{l+1})) \tag{2.2.2}$$

Here, $f'(u_j^l)$ refers to the gradient of the j-th gradient in the l-th layer. β represents the sampling coefficient, and $\circ$ denotes element-wise multiplication. The $up(\cdot)$ represents the upsampling operation, and the subsampling factor is n. Then, one possible way to upsampling operation is to use the Kronecker product:

$$up(x) \equiv x \otimes 1_{n\times n} \tag{2.2.3}$$

Thus, the bias gradient can be represented as follows:

$$\frac{\partial E}{\partial b_j} = \sum_{u,v}(\delta_j^l)_{u,v} \tag{2.2.4}$$

The gradient for the kernel weights are computed by using backpropagation. We sum the gradients for a given weight for all the connections:

$$\frac{\partial E}{\partial k_{ij}^l} = \sum_{u,v}(\delta_j^l)_{u,v}(p_i^{l-1})_{u,v} \tag{2.2.5}$$

Here, the $(p_i^{l-1})_{u,v}$ is the patch in x_i^{l-1}. The equation 2.2.5 can be implemented in a single MATLAB line:

$$\frac{\partial E}{\partial k_{ij}^l} = rot180(conv2(x_i^{l-1}, rot180(\delta_j^l), 'valid')) \tag{2.2.6}$$

Where $'valid'$ refers to the valid convolution operation.

Subsampling layers. For the subsampling layer, the formulation can be represented as follows:

$$x_j^l = f(\beta_j^l down(x_j^{l-1}) + b_j^l) \tag{2.2.7}$$

Where $down(\cdot)$ refers the downsampling operation.

The gradient of subsampling layers. We assume that the subsampling layers is followed by convolution layers, and the sensitivity maps can be computed in MATLAB:

$$\delta_j^l = f'(u_j^l) \circ conv2(\delta_j^{l+1}, rot180(k_j^{l+1}), 'full') \tag{2.2.8}$$

Here, $'full'$ refers the fully convolution operation. Then, we need to compute the gradients for b and β. The additive bias b is the sum over the elements of the sensitivity map:

$$\frac{\partial E}{\partial b_j} = \sum_{u,v} (\delta_j^l)_{u,v} \tag{2.2.9}$$

To acquire the gradient for β, we need to recompute the maps during backpropagation.

$$d_j^l = down(x_j^{l-1}) \tag{2.2.10}$$

Then, the gradient for β can be represented as follows:

$$\frac{\partial E}{\partial \beta_j} = \sum_{u,v} (\delta_j^l \circ d_j^l)_{u,v} \tag{2.2.11}$$

Updating the weight and bias. After computing the value of residual error, we can update the weight and bias. Following the back-propagation algorithm, the updating formula can be represented as follows:

$$W(t+1) = W(t) + \eta \delta(t) x(t) \tag{2.2.12}$$

Here η refers to the learning rate, and $x(t)$ refers to the input. The $\delta(t)$ denotes to the error term.

2.3 The Convolutional Neural Network Training and Inference Process

The type of training process for pattern recognition can be divided into two categories, including the supervised and unsupervised learning. In this thesis, we focus on

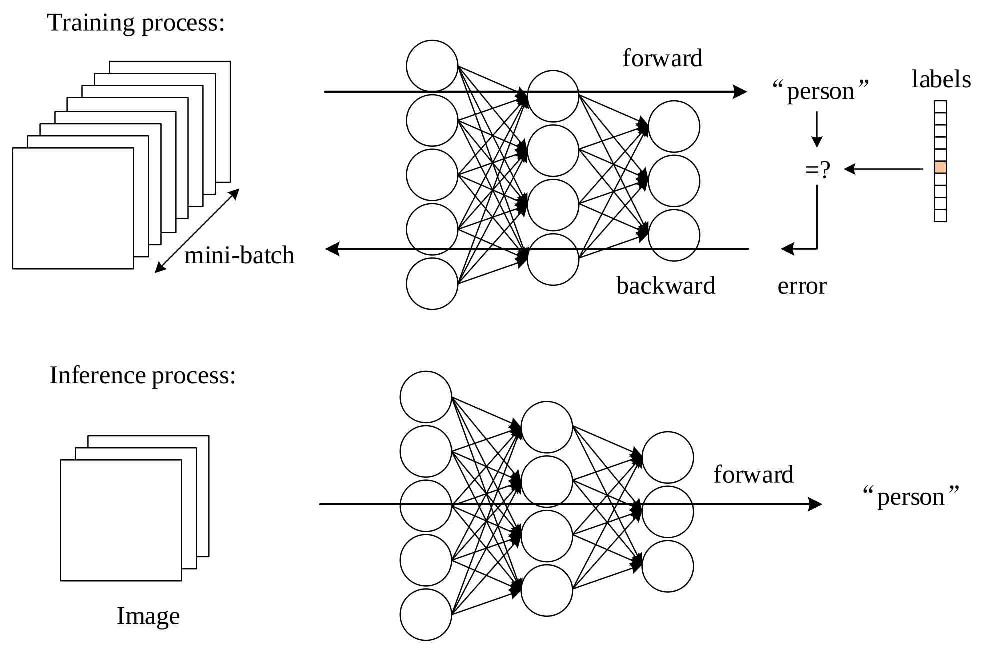


Fig. 2.2.1. Convolutional neural network training process.

supervised learning. The convolutional neural network training process consists of two stages: forward propagation and backward propagation. As shown in Fig. 2.2.1, in the forward propagation, a minibatch images are sampled from the dataset, which contains the images and truth labels. Then, the images are feed into the network and pass through the input layer to the output layer. In the backward propagation, the output obtained from the forward propagation process will be used to compare with the truth labels to compute the training loss. Then, based on the loss value, the model parameters are updated by minimizing the error function. As the loop of training progress, the training loss will be convergenced. Then, we get the optimal model. In the inference period, it only needs to perform the forward process. The images are feed into the trained model to predict and output the category directly.

2.4 Image Classification Model

For traditional image classification task, the primary solutions are based on the handcrafted feature, such as SIFT [33], HOG [34]. These methods could obtain a good result on the low-level feature representations, including color, edge, and texture. However, because of lacking high-level semantic information representation ability, the handcrafted methods can not reach satisfactory results. In recent years, deep learning methods attracted more attention in the computer vision community. Many great convolution neural networks, such as AlexNet [10], VGG [35], and ResNet [35], are proposed and achieve state-of-the-art results on large-scale benchmark datasets, which exceed the handcrafted methods significantly. With the development of convolutional neural network, more and more excellent networks are proposed. There are three models' structure having important impact on the following content, including GoogleNet [12], ResNet [35], DenseNet [9], and SENet [36]. Therefore, in the following section, these models will be shortly introduced.

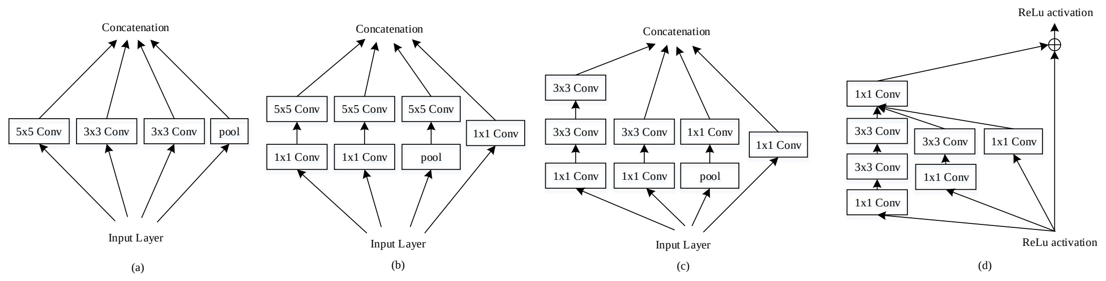


Fig. 2.4.1. The inceptions used in GoogleNet.

GoogleNet. The GoogleNet is proposed in 2014, which explored to improve the model capacity by fusing the feature from multi-scales branches. The authors found that fusing the feature from different scales can effectively alleviate the problem of overfitting, gradient disappearance, and gradient explosion. The model has several variants, including Inception-v1 [12], Inception-v2 [37], Inception-v3 [38], and Inception-v4, Inception-

ResNet [15]. In these models, they construct the model by stacking the inception unit and modify the unit to improve the model performance. Fig. 2.4.1(a) shows the inception structure used in Inception-v1, which utilizes the kernels with different size to extract feature. Based on this observation, it demonstrated that adding the model width can improve the model performance effectively. For GoogleNet, it explores to design a wider model to obtain higher model performance. In Fig. 2.4.1(b), the unit is inserted a 1×1 convolutional layer to decrease the dimension, and it is used in Inception-v2 and Inception-v3 to reduce the number of calculations. In Fig. 2.4.1(c), the authors replaced the 5×5 convolutional layer by two successive 3×3 convolutional layer to not only improve the computational speed but also reduce the model parameters. In Fig. 2.4.1(d), activated by the architecture of ResNet, the authors added the identity mapping in the module, and it is used in Inception-ResNet to accelerate model convergence.

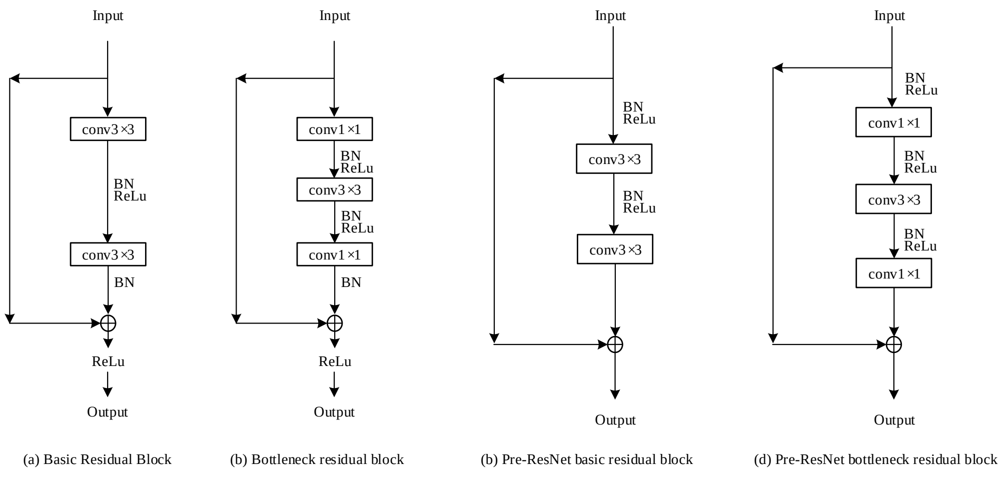


Fig. 2.4.2. Residual block and pre-activation residual block.

ResNet. The residual network (ResNet) is proposed in 2015, which win the 1st in several computer vision tasks, including image classification, object detection, and segmentation. It achieved tremendous improvement than the previous convolutional neural network on ImageNet dataset. In ResNet, it introduces a shortcut conception to propagate information smoother in the entire network, as shown in Fig. 2.4.2 (a) and (b). For deep convolutional neural networks, it is difficult to train due to the problem of degradation. Adding the shortcut conception can alleviate the problem effectively. Even in extremely deep exceeding 1000+ layers, ResNets still achieve considerable accuracy performance on benchmark datasets. For high-resolution image classification task, the authors proposed the bottleneck structure, as shown in Fig. 2.4.2(b). The bottleneck architecture uses a stack of 1×1, 3×3, and 1×1 convolution, where 1×1 convolutional layers are used to reduce and then increase dimension. The bottleneck design can significantly reduce the model parameters with the same depth model. Activated by the excellent performance, more and more deep neural network are proposed based on ResNet. However, for the extremely deep ResNet, the degradation problem appeared. Therefore, to address this problem, the authors proposed the pre-activation residual block, as shown in Fig. 2.4.2 (c) and (d). They regarded the identity mapping as the skip connection and after-addition activation. Meanwhile, they converted the order of Conv-Bn-ReLu to Bn-ReLu-Conv.

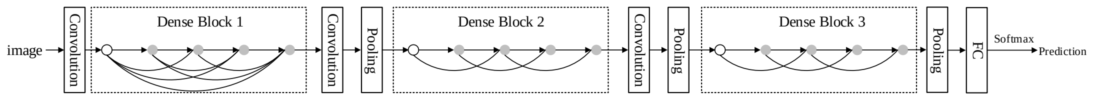


Fig. 2.4.3. Densely connection network.

DenseNet. Activated by the characteristic of creating short paths from early layers to later layers, the author proposed the densely connected convolutional neural network. To ensure maximum information flow between layers in the network, they design to connect all layers directly with each other. Fig. 2.4.3 shows the architecture of DenseNet. From the Fig. 2.4.3, we can see the l-th layer has l inputs, which consist of the feature maps from all preceding convolutional blocks. Unlike in ResNet using element-wise summation to fuse the feature, DenseNet adopt the concatenation operation to fuse the feature from preceding layers. Therefore, in the L-layer network, it has $(L(L+1)/2$ connections, instead of just L, as in the traditional architecture. Due to the densely connections, the previous feature maps can be accessed by all subsequent layers, which encourages feature reuse throughout the network.

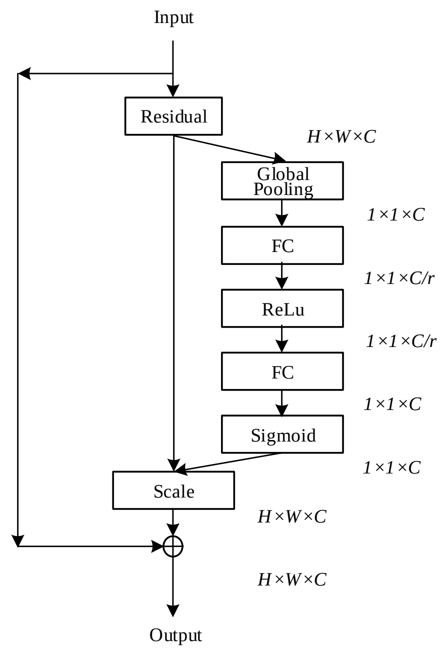


Fig. 2.4.4. Squeeze and excitation residual block.

Squeeze-and-Excitation Networks. Squeeze-and-Excitation Networks (SENet) [36] was proposed in 2017 and achieved the champion on ImageNet 2017 competition. To boost the representation power for the existing network, the work focus on modeling the channel relationship and adaptively recalibrates channel-wise feature responses. The architecture of the squeeze-and-excitation block contians three operations, inculding squeeze, excitation, and reweight. Fig. 2.4.4 illustrates the architecture of the squeeze and excitation module with residual block. For squeeze operation, the authors adopt the global average pooling to compress the information into a one-dimension vector. Then, in excitation operation, to make use of the information aggregated in the squeeze operation, the authors employed a gating mechanism with a sigmoid activation. To realize this purpose and limit the model complexity, two fully connection with bottleneck architecture is applied into the architecture, and a relu function is used to enhance the capacity of the model non-linearity. The final output of the SE module performs the channel-wise multiplication to realize the purpose of feature recalibration. In this way, the model can learn to selectively emphasise information features and suppress less useful ones.

2.5 Datasets

In this thesis, the proposed models are evaluated on several benchmark datasets, including CIFAR-10, CIFAR-100, SVHN, and IP102. In the following paragraphs, we introduce the information of these datasets.

CIFAR-10 [39] is a dataset comprising a collection of 50k training images and 10k testing 32×32pixel RGB images in 10 classes of natural scene objects. Fig. 2.5.1 shows some example images from CIFAR-10 dataset, which includes airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.

Similar to CIFAR-10, CIFAR-100 [39] is a dataset comprising a collection of 50k training images and 10k testing 32×32pixels RGB images, but the number of classes is extended to 100. Due to each class only consists of 600 images, it is more challenging for classification on CIFAR-100 dataset.

The Street View House Number (SVHN) [40] dataset is also a well-known benchmark dataset in computer vision. The dataset contains 73,257 digits in the training set, 26,032 in the test set and 531,131 additional training images respectively. All the images are obtained from house numbers in Google Street View images. The label of dataset are the house number from 0 to 9.

The resolution of the images from CIFAR-10, CIFAR-100, and SVHN datasets is

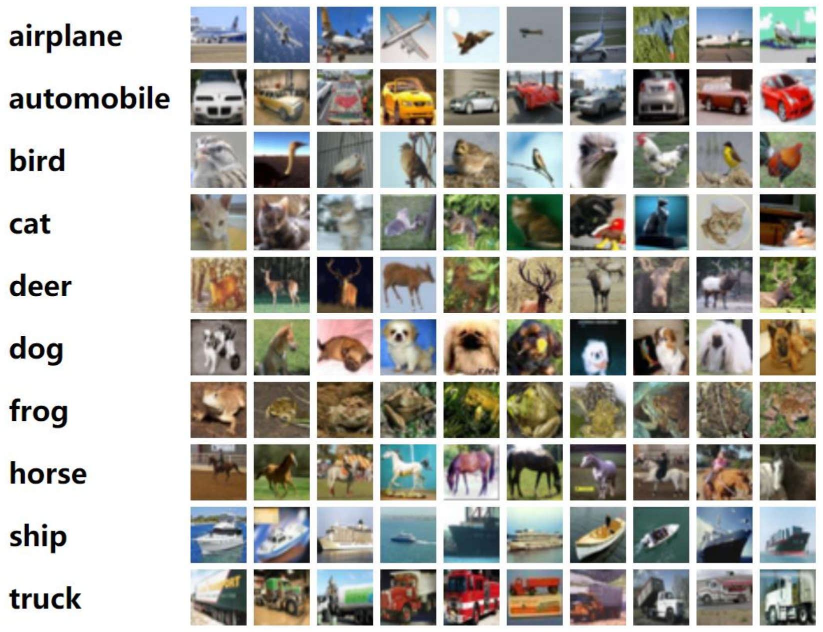


Fig. 2.5.1. Example images from Cifar-10 dataset

low. Due to this reason, it do not need much time to train a CNN model on these datasets. Therefore, most of works first demonstrate the effectiveness their method on the these datasets. Then, it is applied to high-resolution image classification tasks. In this thesis, we also demonstrate the validness of our method on CIFAR-10, CIFAR-100, and SVHN datasets, then it is applied to recognize insect pests.

IP102 [41] is a large-scaled insect pest dataset covered 102 species of common crop insect pests. The dataset contains 45,095 images in the training set, 7,508 images in the validation set and 22,619 images in the testing set for classification task. Fig. 2.5.2 shows some example images from IP102. As illustrated in [41], there are several factors affecting the classification performance. First, the pests are difficult to be distinguished, because the colors are similar between object and background. Second, IP102 contains the image throughtout pests life cycle, and it is hard to classify especially in the larval period. Third, the pests between classes are often similar. Due to these factors, it is more challenging for classification on IP102 dataset.

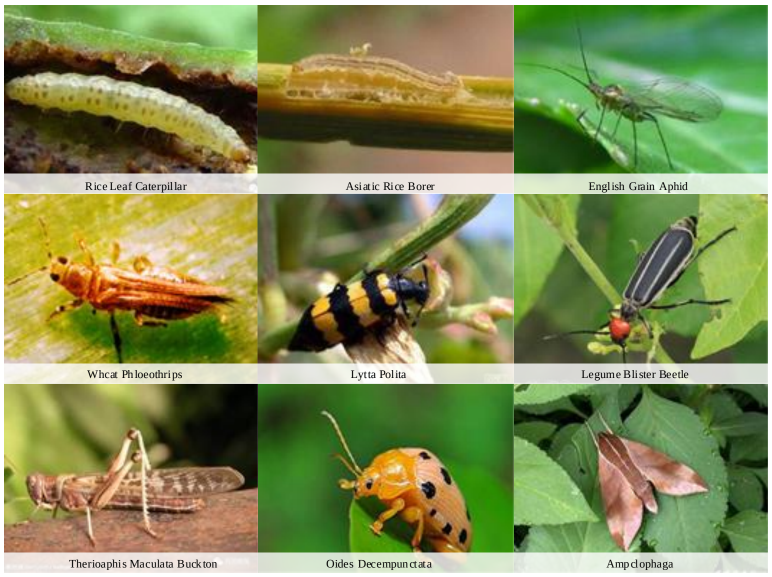


Fig. 2.5.2. Example Images from the IP102 dataset.

Chapter 3

Related Work

In this section, we will review some related work, including deep convolutional neural networks, residual networks, feature fusion networks, attention mechanism in CNNs, and application in insect pest recognition.

3.1 Deep Convolutional Neural Networks

The development of convolutional neural networks undergo for a long time. In 1998, LeNet-5 [26] was trained with back-propagation algorithm, which forms the embryonic of contemporary convolutional neural network. Until the AlexNet [10] proposed in 2012, it consists of five convolutional layers, some of which are followed by max-pooling layer. In the output layer, it contains three fully connected layer with a final 1000-way softmax. Meanwhile, the authors employed the dropout method to reduce overfitting in the fully connected layers. Then, the convolutional neural networks attract more attention, and more and more convolutional neural networks emerged, such as SegNet [42], NiN [43], GoogLeNet [12], and XNOR-Net [44]. In VGGNet [35], it proposed an architecture with small (3×3) convolution filters, and the depth of the network can be pusshed to 16-19 weight layers. With depth going deep, the accuracy has continued to increase. However, very deep CNNs have to face the crucial problem of vanishing gradients. Initialization methods and layer-wise training were adopted to reduce this problem in earlier works. Moreover, the ReLU [31] activation function and its variants were also used to prevent vanishing gradients, such as ELU [45], PReLU[32], and PELU [46]. Batch normalization (BN) [37] could also largely address this problem through standardizing the mean and variance of hidden layers for each mini-batch, and MSR initialized the weights with a more reasonable variance. Meanwhile, a degradation problem has been emerged, and several methods were proposed to resolve this problem. Inspired by Long Short-Term Memory recurrent networks [47] and by using adaptive gating units to regulate the information flow, Highway Networks [48] can be trained directly through simple gradient descent. ResNets [35] introduced a shortcut conception to propagate information to deeper layers of networks, which are simpler and more effective than Highway Networks. Consequently, ResNets construct deep residual network with layers exceeding 1000+ and still have compelling accuracy and nice convergence behaviors on many computer vision tasks. Therefore, it attracts many researchers, and more and more residual network variants have been proposed as a family of extremely deep architectures. The models proposed in this thesis are also based on ResNet.

3.2 Residual Networks

ResNets achieved significant success in computer vision. However, ResNets [8] become difficult to converge when the depth goes very deep. Therefore, Pre-ResNets [14] proposed a new residual block with a BN-ReLU-Conv order to reducing training difficulties with identity mappings as the skip connections and after-addition activation. Weighted Residual Networks [49] found the original residual networks have the incompatibility between ReLU and element-wise addition and deep network initialization problem. Therefore, they proposed the weighted residual networks, which enjoy a consistent improvement over accuracy when depths increase from 100+ layers to 1000+ layers. More residual network variants try to improve performance by constructing deeper residual networks, while the problem of diminishing feature reuse for very deep residual networks makes these networks very slow to train. To tackle these problems, WRNs [50] generated residual networks by decreasing depth and increasing width of residual networks. WRNs improved accuracy and reduced the training time compared with thin and very deep counterparts. Huang et al. [51] proposed a stochastic depth drop-path method which randomly drops a subset of layers and bypasses them with identity function. Their experiments showed that their method shortened training time substantially and reduced the test errors. RoR [52] further dug the optimization ability of residual networks by adding shortcut connections upon original residual networks. Pyramidal Residual Network [53] enhanced the generalization ability by increasing the feature map dimension gradually instead of sharply increasing the feature map dimension at down-sampling location. More and more residual variants networks are proposed and form a fammily of ResNet [8, 14, 49, 50, 51, 52, 53, 54, 55, 56, 57].

3.3 Feature Fusion Networks

Many convolutional neural networks adopted feature fusion method to gain model performance improvement in several computer vision tasks, inculding image classification [58, 59, 60, 61], object detection [62, 63, 64, 65], segmentation [66, 67, 68, 69], and other domains [70, 71, 72]. In DenseNets [9], the features from preceding layers are input into subsequent layers directly. In this way, DenseNet can build very deep networks without the problem of training difficulty; thus, it had achieved state-of-the-art results with reusing features method. CondenseNet [73] proposed a learned group convolution to intensify the capacity of the network by removing superfluous feature reuse connections. In ShuffleNet V2 [74], half of the feature channels directly go through the block and are input into the next block, which is deemed to a kind of feature reuse. DSOD [16] trained an object detector model from scratch by learning half of the feature and reusing half from the contiguous high-resolution feature maps. Therefore, the feature fusion method is an effective method to enhance the model performance. In this thesis, our proposed works also adopt the feature fusion method to construct more validness ConvNet models for image classification tasks.

3.4 Attention Mechanism in CNNs

Attention mechanism has been approved their effectiveness in many tasks, such as sequence learning [75, 76, 77, 78], localization [79, 80, 81, 82], and image captioning [83, 84, 85, 86] etc. Meanwhile, soft attention can be trained end-to-end for convolutional neural networks. Therefore, some works combined the soft attention method with the existing models in an innovative way to construct new models. Wang et al. [87] proposed the attention residual learning, which helped to train very deep residual attention networks. They used the mixed attention to capture different types of attention guiding feature learning and encoded top-down attention mechanism into trunk-and-mask architecture based on hourglass modules [88]. To strengthen the model representational power, J. Hu et al. [36] proposed a light-weight squeeze-and-excitation block to adaptively recalibrate channel-wise feature responses. It can significantly promote the model performance for existing state-of-the-art CNN models. M. Luo et al. [89] proposed a stochastic region pooling module to improve the capacity of channel-wise attention network. This module made the channel descriptors more diversity and representative through generating more or wider important feature response. To realize the adaptive receptive field sizes of neurons, X. Li et al. [90] proposed a selective kernel convolution to aggregate information from multiple kernels. Woo et al. [91] proposed a simple yet effective attention module for feed-forward convolutional neural network, which sequentially infers attention maps in two dimensions, channel and spatial.

3.5 Application in Insect Pest Recognition

For the traditional insect pest recognition task, the primary solutions are based on the handcrafted feature, such as SIFT [33], HOG [34]. These methods obtain a good result on the low-level feature representations, including color, edge, and texture. However, because of lacking high-level semantic information representation ability, the handcrafted methods can not reach satisfactory results. In recent years, deep learning methods attracted more attention in the research community. Many great convolution neural networks, including VGG [11], ResNet [8], and GoogleNet [12], are proposed and achieve state-of-the-art results on large-scale benchmark datasets, which exceed the handcrafted methods significantly. The deep learning technology is alsp applied in agriculture in recent years, and most of applications focused on identification of weed [92, 93], plant recognition [94, 95, 96], fruits counting [97] and crop type classification [98]. To solve the problem of pests' different scales and attitudes, R. Li et al. [99] proposed an valid data augmentation strategy for CNN-based models. To detect and classify eight insects, K. Dimililer and S. Zarrouk [100] proposed a two-stages method depended on neural networks. Liu et al. [101] released their dataset consisting of about 5,000 training images in 12 categories of paddy field pests and trained a deep CNN model on this dataset. Meanwhile, a large-scale dataset will promote the development of insect pest recognition. X. Wu et al. [41] collected a large-scale dataset called IP102 for insect pest recognition, which consists of more than 75,000 images in 102 classes. In this thesis, we also evaluated our models on IP102 benchmark dataset to demonstrate the effectiveness of our methods.

Chapter 4

Feature Reuse Residual Network

In Densenet, it reuse the feature from previous layer throughout the network to improve the model performance. Besides, GoogleNet fuse the feature from multi-scale branches to enhance the capacity of the network. Activated by these networks, the feature reuse residual network is proposed by reusing the feature from previous layer and adding a branch to wide the model in the residual block. In the following section, the methodology of the feature reuse residual network will be introduced.

4.1 Methodology

The original residual block with identity mapping can be expressed by the following computation:

$$y_l = h(x_l) + F(x_l, w_l) \tag{4.1.1}$$

$$x_{l+1} = f(y_l) \tag{4.1.2}$$

Where x_{l+1} and x_l are output and input of the l-th residual block in the network, F is a residual function and w_l are parameters of the l-th residual block. The function $h(x_l)$ is an identity mapping: $h(x_l) = x_l$, and f is a ReLU function. Fig 4.1.1(a) shows the original residual basic block with branched residual signal consisting two successive 3×3 conv layers. Residual network consists of sequentially stacked residual blocks.

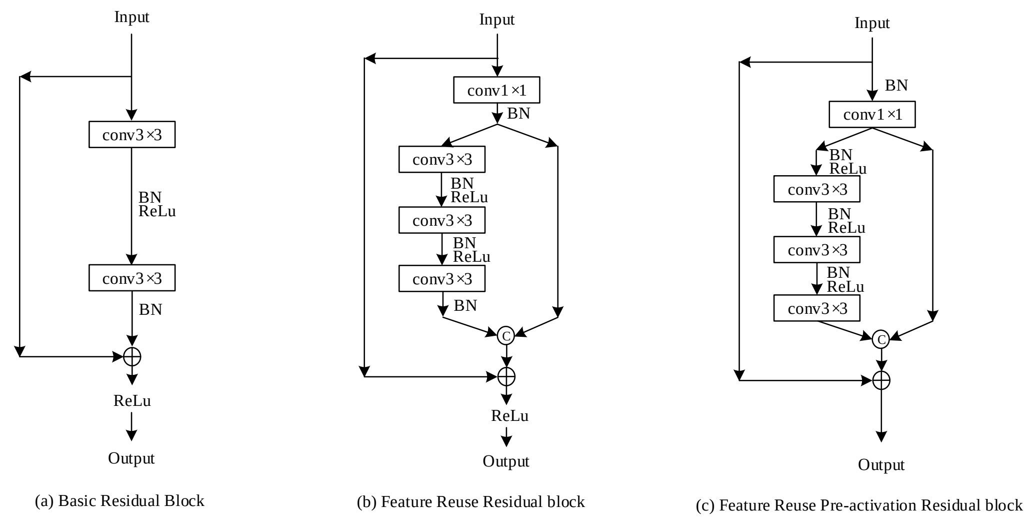


Fig. 4.1.1. The original basic residual block and the feature reuse residual blocks.

We try to explore the effective of feature reuse for the original residual block by adding extra connection from the input signal of residual block. In order to match feature size and dimension, the input feature maps pass through a 1×1 conv layer without ReLU function shown in Fig. 4.1.1(b). To maximize the performance of network, we experiment kinds of residual block size and analysis the result in the following section. Therefore, the Feature Reuse Residual block can be expressed by the following formulations:

$$y_l = h(x_l) + F\left(g(x_l), w_l\right)\ o\ g(x_l) \tag{4.1.3}$$

$$x_{l+1} = f(y_l) \tag{4.1.4}$$

Where g is a function to transform feature map size and dimensions, which is realized by a 1×1 conv layer without ReLU. Meanwhile, the location of reducing feature size also have significant impact on test error, and our experiments empirically show that using average pooling layer before 1×1 convolutional layer in down-sampling block can achieve better performance than other options. The comparison of this matter is continued in the following section.

4.2 Model Optimization

In order to optimize FR-ResNet, we must determine some important principles, such as residual block size, and location of reducing feature map size. We tested these principles on CIFAR-10 benchmark dataset.

In the case of original ResNets, the basic residual unit consists of a stack of two 3×3 convolutional layers in [8] as shown in Fig. 4.1.1(a). In order to analyze the effects of different residual block sizes, we explored several types of convolutions in every residual block with a similar total number of parameters. We use $B(M)$ to denote residual block structure, which is used by WRN [50], and M is a list with the kernel sizes of the convolutional layers in a block. For example, $B(1,3,3,3)$ denotes a residual block with one 1×1 and three 3×3 convolutional layers as shown in Fig. 4.1.1(b). We experiment with these types of convolutions on CIFAR-10 dataset, and the results are reported in Table 4.2.1. The experimental results show that $B(1,3,3,3)$ achieved the best performance when the epoch number was 164 and 500.

Tab. 4.2.1. Test error (%) on CIFAR-10 with different types of convolutions for FR-ResNet.

block type	depth	paprams	164 epoch	500 epoch
B(1,3,3)	98	1.7M	6.12	5.23
B(1,3,3,3)	135	1.7M	5.71	4.76
B(1,3,3,3,3)	122	1.7M	6.77	5.52

The experiments show that the performance can vary depending on the location of reducing feature map size in down-sampling block. We consider four variants in this paper: (A) adding a 2×2 average pooling layer before 1×1 conv; (B) adding a 3×3 max pooling layer with the stride of 2 before 1×1 conv; (C) 1×1 conv with the stride of 2; (D) first 3×3 conv with the stride of 2. The test result are shown in Fig. 4.2.1 which shows A or B can achieve similar and lower test error than other variants on CIFAR-10 dataset. Therefore, in this work, we choose A in our structures.

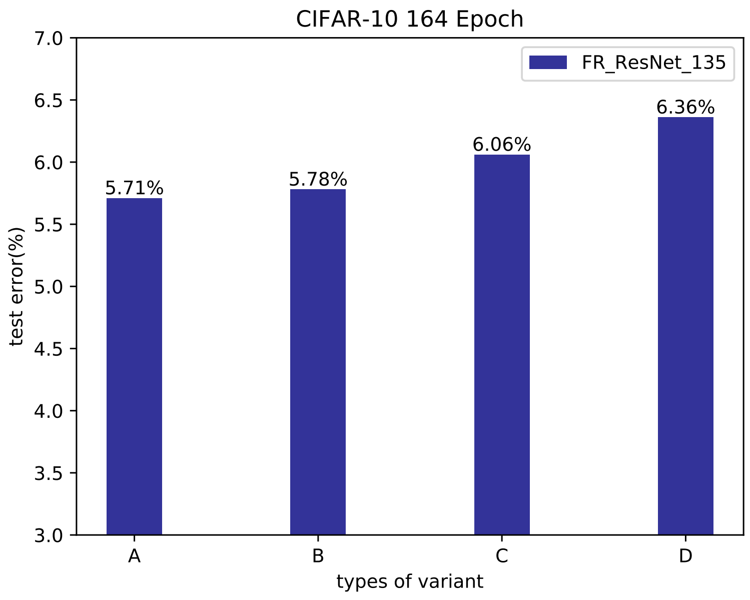


Fig. 4.2.1. Comparison of FR-ResNet with different location of reducing feature map size on CIFAR-10. Using type A can achieve best performance than others.

4.3 Experiments and Analysis

In order to investigate the effectiveness of our approach generalize, we combined our method with various residual networks: ResNet, Pre-ResNet, and WRN and evaluate the performance on a series of benchmark datasets: CIFAR-10, CIFAR-100, and SVHN. Then, we demonstrated the effectiveness of our approach on IP102 dataset for insect pest recognition.

4.3.1 Implementation on CIFAR-10, CIFAR-100 and SVHN datasets

We combined our method with various residual networks: ResNet, Pre-ResNet, and WRN and evaluate the performance on CIFAR-10, CIFAR-100, and SVHN datasets to demonstrate the effectiveness of our approach. We compared the results of FR-ResNets and the original ResNets baseline with a similar total number of parameters. In the case of CIFAR, we used the 135-layer and 194-layer FR-ResNets compared with 110-layer and 164-layer ResNets, respectively. The original ResNets contained three groups of 16 filters, 32 filters and 64 filters of residual blocks, and the feature map sizes respectively are 32, 16 and 8. As shown in Fig. 4.1.1(b), we adopted each convolution in residual mapping following batch normalization and activation (ReLU). In FR-Pre-ResNet and FR-WRN experiments, we adopted BN-ReLU-Conv order. For CIFAR datasets, we initialize the weights with Kaiming Xavier algorithm [32] and use SGD with a mini-batch size of 128 for 500 epochs as in [52]. The learning rate is initialized by 0.1 and is divided by 10 at the 250th and 375th. For SVHN dataset, we adopted SGD with a mini-batch size of 128 for 50 epochs. The learning rate is initialized by 0.1 and is divided by 10 at the 30th and 35th as in [52]. The weight decay is 0.0001, and momentum is 0.9 on all datasets. According to [51], the stochastic drop-path method can alleviate overfitting and enhance the test performance, so we also adopted this method in this paper. We set p_l with the linear decay rule of $p_0 = 1$ and $p_L = 0.5$ when depth exceeds 100 layers, and we set $p_0 = 1$ and $p_L = 0.8$ with the linear decay when the depth is less than 100 layers. In the following sections, we use "SD" to denote adopting the stochastic drop-path method in the experiments.

4.3.2 CIFAR-10 Classification by FR-ResNet

The standard data augmentation strategies were adopted in our experiments for training: 4 pixels are padded on each side, then a random 32×32 crop is sampled from the padded image; mean and standard deviation normalization is also applied or its horizontal flip.

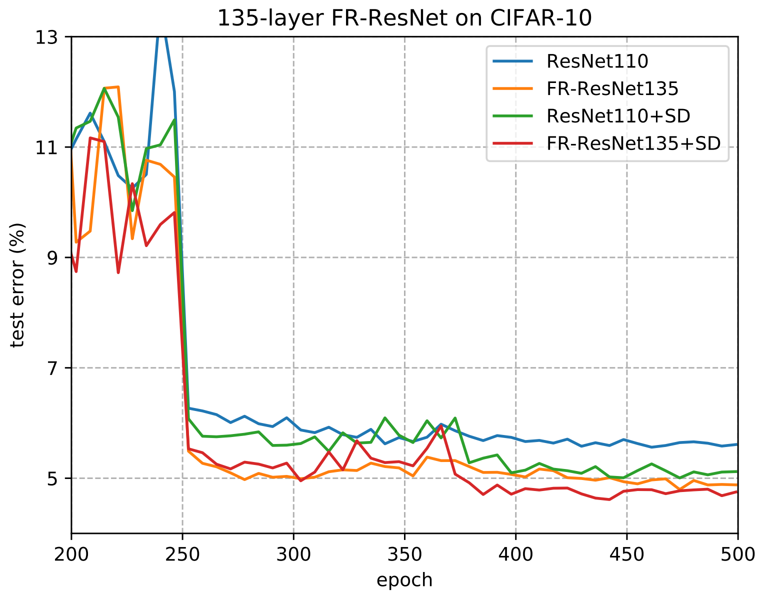


Fig. 4.3.1. Smoothed test errors on CIFAR-10 by ResNets, FR-ResNet, ResNets+SD and FR-ResNet+SD during training, corresponding to results in Table 4.3.1. Either FR-ResNet without SD (the orange curve) or FR-ResNet+SD (the red curve) is shown yielding a lower test error than ResNets.

Tab. 4.3.1. Test error(%) on CIFAR-10 by ResNets and FR-ResNets.

CIFAR-10 500 Epoch	depth	# params	error(%)	error(%)+SD
ResNets	110	1.7M	5.79	4.84
	164	2.6M	5.59	4.70
FR-ResNets	135	1.7M	**4.76**	**4.57**
	194	2.5M	**4.96**	**4.15**

In Table 4.3.1 and Fig. 4.3.1, we construct 135-layer and 194-layer FR-ResNets compared with 110-layer and 164-layer ResNets with a similar total number of parameters, respectively. The 110-layer ResNets without SD achieved a competitive 5.79% error on the test set. The 135-layer FR-ResNets without SD had a 4.76% error on the test set and outperformed the 110-layer ResNets without SD by 17.8% on CIFAR-10.

As can be observed, the 135-layer FR-ResNets without SD can also slightly outperform the 4.92% error of the 1001-layer Pre-ResNets with the same mini-batch size []. The 194-layer FR-ResNets without SD achieved a 4.96% error on the test set, and it outperforms the 164-layer ResNets without SD by 11.3%. Consequently, we found that the 194-layer FR-ResNets performance is worse than 135-layer FR-ResNet while the result changed when we added SD. We conjectured that the incompatibility between ReLU and element-wise addition degraded the accuracy and feature reuse method made the situation worse, and the problem was resolved in FR-Pre-ResNets which treat both $h(x_l)$ and f serve as identity mappings.

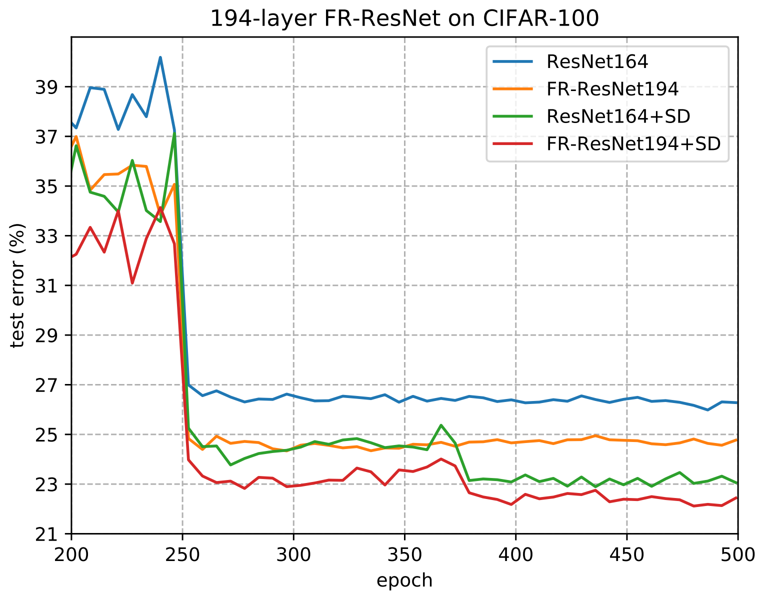


Fig. 4.3.2. Smoothed test error on CIFAR-100 by ResNets, FR-ResNet, ResNets+SD and FR-ResNet+SD during training, corresponding to results in Table 4.3.2. FR-ResNet+SD (the red curve) yields lower test errors than other curves.

Tab. 4.3.2. Test error (%) on CIFAR-100 by ResNets and FR-ResNets.

CIFAR-100 500 Epoch	depth	# params	error(%)	error(%)+SD
ResNets	110	1.7M	26.21	23.45
	164	2.6M	25.96	22.78
FR-ResNets	135	1.7M	**24.88**	**22.19**
	194	2.5M	**24.26**	**21.83**

4.3.3 CIFAR-100 Classification by FR-ResNet

We adopt the same augmentation and preprocessing techniques as on CIFAR-10. In Table 4.3.2 and Fig. 4.3.2, the 110-layer and 164-layer ResNets without SD achieved a competitive 26.21% and 25.96% error on the test set, and the results of the 135-layer FR-ResNets and 194-layer FR-ResNets without SD had 24.88% and 24.26% error on the test set. Unlike on CIFAR-10, FR-ResNets without SD outperformed their counterparts obviously on CIFAR-100 even when depth become deep. FR-ResNets perform better performance on more challenging dataset. It is gratifying that the 135-layer FR-ResNets+SD and 194-layer FR-ResNets+SD achieved a 22.19% and 21.83% error on the test set, and they outperformed the 110-layer ResNets, 110-layer ResNets+SD, 164-layer ResNets and 164-layer ResNets+SD by 15.3%, 5.3%, 15.9% and 4.2%, respectively on CIFAR-100.

Tab. 4.3.3. Test error(%) on CIFAR-10 and CIFAR-100 by Pre-ResNets and FR-Pre-ResNets.

500 Epoch	depth	# params	CIFAR-10		CIFAR-100	
			error(%)	error(%)+SD	error(%)	error(%)+SD
Pre-ResNets	110	1.7M	5.22	4.71	25.93	23.99
	164	2.6M	4.75	4.69	25.06	22.98
FR-Pre-ResNets	135	1.7M	**4.41**	**4.35**	**23.38**	**21.53**
	194	2.5M	**4.36**	**3.90**	**22.39**	**20.73**

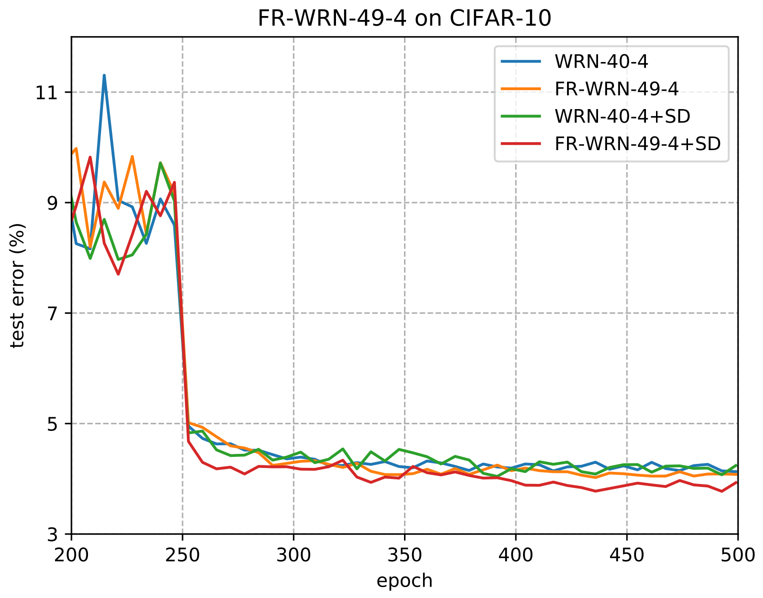


Fig. 4.3.3. Smoothed test error on CIFAR-10 by WRN40-4, WRN40-4+SD, FR-WRN49-4 and FR-WRN49-4+SD during training, corresponding to results in Table 4.3.4. FR-WRN49-4+SD (the red curve) yields lower test errors than the other curves.

4.3.4 Feature Reuse for Pre-ResNet and WRN

Pre-ResNets [14] changed the order of Conv-BN-ReLU to BN-ReLU-Conv to reduce vanishing gradients, and WRN [50] can achieve a dramatic performance improvement by decreasing depth and increasing width of residual networks. First, we changed the residual blocks of the original FR-ResNet with a BN-ReLU-Conv order as shown in Fig. 4.1.1(c). We did the same experiment by FR-Pre-ResNet on CIFAR-10 and CIFAR-100, and the results are reported in Table 4.3.3 where FR-Pre-ResNet is compared with Pre-ResNet. As can be observed, the 135-layer and 194-layer FR-Pre-ResNets with SD achieved 4.35% and 3.9% test error, and they outperformed the 110-layer Pre-ResNet, 110-layer Pre-ResNet+SD, 164-layer Pre-ResNet and 164-layer Pre-ResNet+SD by 16.7%, 7.6%, 17.9% and 16.8%, respectively on CIFAR-10. Consequently, a similar phenomenon appeared on CIFAR-100.

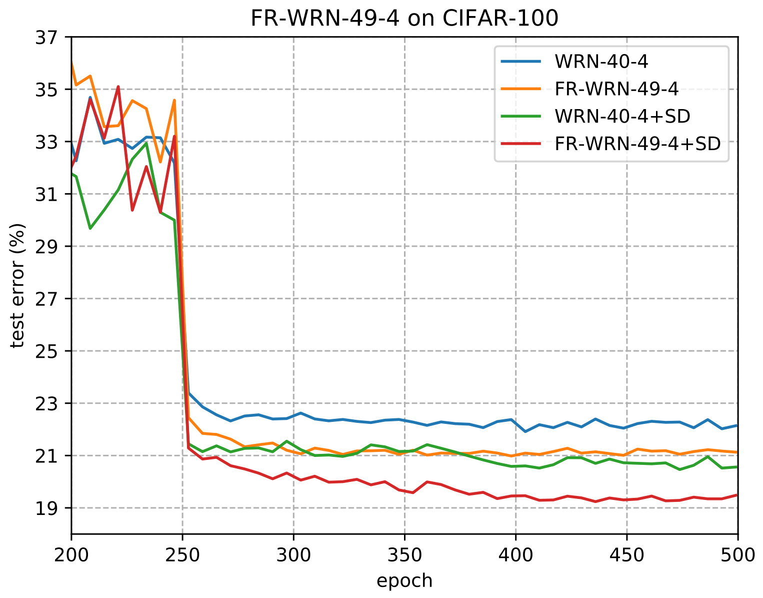


Fig. 4.3.4. Smoothed test error on CIFAR-100 by WRN40-4, WRN40-4+SD, FR-WRN49-4 and FR-WRN49-4+SD during training, corresponding to results in Table 4.3.4. FR-WRN49-4+SD (the red curve) yields lower test errors than the other curves.

In the case of WRN, we found $B(1,3,3)$ can achieve better performance. Therefore, $B(1,3,3)$ was adopted in FR-WRN. We did the same experiment by 40-layer WRN and 49-layer FR-WRN of a width of 2 and 4 on CIFAR-10 and CIFAR-100 with a similar total number of parameters. Table 4.3.4 shows the results of our FR-WRN compared with WRN. Fig. 4.3.3 and Fig. 4.3.4 show the test errors on CIFAR-10 and CIFAR-100 at different training epochs. The experimental results show that FR-WRNs are better than WRNs on CIFAR-10 and CIFAR-100, and SD can further improve the performance by alleviating overfitting. FR-WRN49-4+SD achieved 3.73% test error on CIFAR-10 and 19.16% test error on CIFAR-100, and it outperformed WRN-40-4+SD by 6.3% on CIFAR-10 and 5.5% on CIFAR-100. Through analysis and experiments, we conclude that our Feature Reuse Residual architecture can also promote the performance of other residual networks, such as Pre-ResNets and WRNs.

Tab. 4.3.4. Test error(%) on CIFAR-10 and CIFAR-100 by WRNs and FR-WRNs.

500 Epoch	depth	# params	CIFAR-10		CIFAR-100	
			error(%)	error(%)+SD	error(%)	error(%)+SD
WRNs	40-2	2.2M	4.63	4.25	24.42	22.21
	40-4	8.9M	4.07	3.98	21.85	20.28
FR-WRNs	49-2	2.2M	**4.50**	**4.18**	**23.21**	**21.45**
	49-4	8.7M	**3.99**	**3.73**	**20.92**	**19.16**

Tab. 4.3.5. Test error(%) on CIFAR-10 and CIFAR-100 by FR-ResNet with different depths.

Depth	CIFAR-10 FR-ResNet	CIFAR-100 FR-ResNet
135-layer	4.76	24.88
194-layer	4.97	24.26
314-layer	5.09	23.25

4.3.5 Effect of Feature Reuse, Depth and Width

Based on preceding experiments, we can conclude that increasing width or depth can improve the performance. In order to investigate the effect of width and depth to feature reuse residual network, we explored the following experiments.

The FR-ResNets derive from the original ResNets, and the vanishing gradients problems appear when depth goes deep. As shown in Table 4.3.5, the test error gradually increases from 135-layer to 194-layer, then to 314-layer on CIFAR-10. Interesting, in the case of CIFAR-100, FR-ResNets present consistent improvement. These experiments indicated that the vanishing problem still exists in deep FR-ResNet, and feature reuse method is more effective on the competitive dataset, for example, CIFAR-100.

In the case of Pre-ResNet, it reduced the vanishing problem. We constructed FR-Pre-ResNet based on Pre-ResNet and experimented with different depth, as shown in Table 4.3.6. As can be observed, the test error gradually reduced as the number of layers increased. For the 1202-layer FR-Pre-ResNet with a batch size of 32, it achieved the

Tab. 4.3.6. Test error(%) on CIFAR-10 and CIFAR-100 by FR-Pre-ResNet with different depths.

Depth	CIFAR-10 FR-Pre-ResNet+SD	CIFAR-100 FR-Pre-ResNet+SD
135-layer	4.35	21.53
194-layer	3.90	20.73
242-layer	3.85	20.01
1202-layer (15.7M,bs=32)	3.74	17.85

Tab. 4.3.7. Test error(%) on CIFAR-10 and CIFAR-100 by FR-WRN with different widths and depths.

Depth and Wideth	CIFAR-10 FR-WRN+SD	CIFAR-100 FR-WRN+SD
FR-WRN49-2	4.18	21.45
FR-WRN49-4	3.73	19.16
FR-WRN76-2	3.88	20.21
FR-WRN76-4	3.39	18.64
FR-WRN94-4 (17.3M)	3.34	17.99

3.74% test error on CIFAR-10 and 17.85% test error on CIFAR-100. So, we conclude that the vanishing gradients can be alleviated, even on very deep FR-Pre-ResNet.

In the case of WRN, the vanishing problem is not obvious for the shallow network, but adding more feature planes and parameters introduce overfitting. We explored experiments with FR-WRN with different widths and depths on CIFAR-10 and CIFAR-100, as shown in Table 4.3.7. The experiments show that the network performance can be improved with both depth and width increasing. But when we widened the FR-WRN, the problem of overfitting appeared. So, we need to reduce it by SD. As can be observed, FR-WRN-94-4+SD achieved a 3.34% test error on CIFAR-10 and a 17.99% test error on CIFAR-100. However, we found that the 1202-layer FR-Pre-ResNet+SD compared with

FR-WRN-94-4+SD can achieve lower test error on CIFAR-100 with fewer parameters. It demonstrates that increasing depth is more effective for feature reuse residual networks on CIFAR-100.

Based on these experiments and analysis, we conclude that both adding depth and width of feature reuse residual networks are effective for model learning capability. Therefore, we must carefully choose the tradeoff between the depth and width to achieve satisfying results.

4.3.6 SVHN Classification Results

For SVHN dataset, following the common practice, we used all the training samples but without performing data augmentation. Mean and standard deviation normalization is also applied to preprocess the data. To demonstrate the effectiveness of our method on SVHN, we used WRN40-4 and FR-WRN49-4 with a similar total number of parameters to train SVHN, and the results are reported in Table 4.3.8. As can be observed, FR-WRN49-4+SD outperformed WRN40-4+SD by 7.4% on SVHN, and Fig. 4.3.5 showed the training curves. These experiments showed our approach can achieve improvement on SVHN dataset too.

Tab. 4.3.8. Test error(%) on SVHN by WRNs and FR-WRNs with different depths.

SVHN	depth	# params	error(%)	error(%)+SD
WRN	40-4	8.9M	**1.69**	1.75
FR-WRN	49-4	8.7M	1.78	**1.62**

Tab. 4.3.9. Test accuracy(%) on IP102 by FR-ResNets.

IP102	# depth	# params	F1	Acc (%)
FR-ResNet	34	20.67M	53.58	54.73
	50	30.78M	54.18	55.24

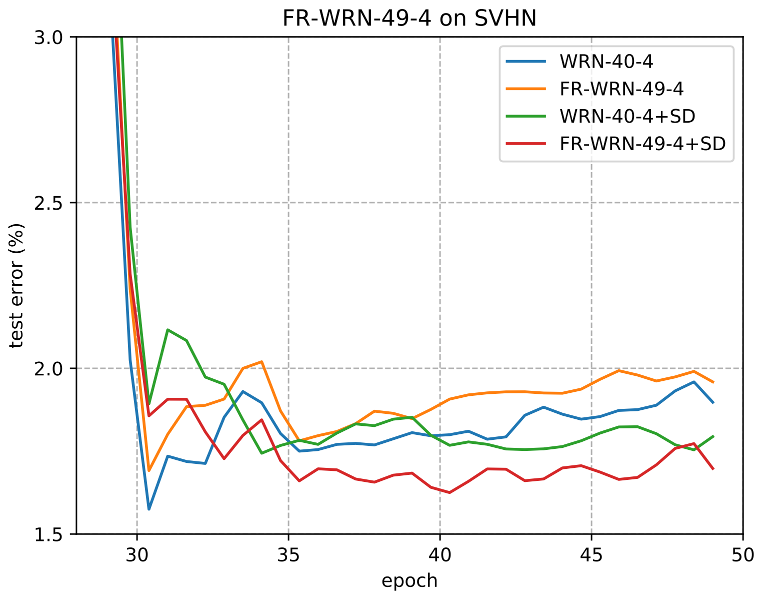


Fig. 4.3.5. Smoothed test error on CIFAR-10 by WRN40-4, WRN40-4+SD, FR-WRN49-4 and FR-WRN49-4+SD during training, corresponding to results in Table 4.3.8. FR-WRN49-4+SD (the red curve) yields lower test errors than the other curves.

4.3.7 IP102 Classification Results

The testing results on CIFAR and SVHN datasets demonstrated the effectiveness of our approach. In this part, we applied our method to recognize the insect pests.

The ResNets models for ImageNet contain four residual block groups, which required 64, 128, 256, 512 filters for basic residual block or 256, 512, 1024, 2048 for bottleneck residual block. More planes will increase the number of parameters and introduce overfitting problem for IP102. Therefore, to limit the parameters, we only conducted feature reuse residual network based on basic residual block. For IP102 dataset, we follow the implementation in [41]. SGD was adopted with a mini-batch size of 64. The learning rate is initialized by 0.01 and is divided by 10 every 40 epochs. The weight decay is 0.0005, and momentum is 0.9. The data augmentation strategies were adopted in our experiments for training: first, the image is resized into 256×256 square image;

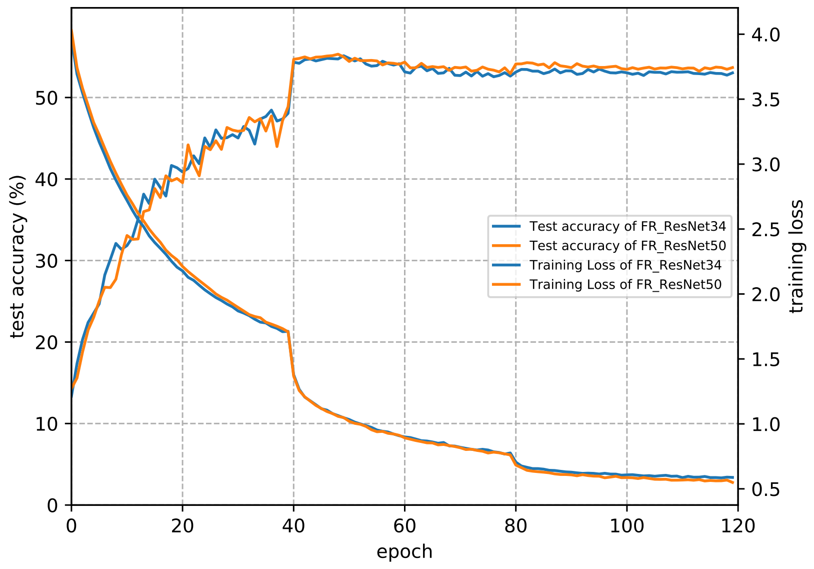


Fig. 4.3.6. Test accuracy on IP102 by 34-layer FR-ResNets and 50-layer FR-ResNet, corresponding to results in Table 4.3.9.

second, a rectangular region is randomly cropped with aspect ratio randomly sampled in [3/4, 4/3] and area randomly sampled in [0.08, 1]; third, the cropped region resized into a 224×224 square image; last, mean and standard deviation normalization is also applied. During test, we followed the processing of training except for randomly augmentation and cropped out the 224×224 regions in the center of the resized image during validation. We trained our models on the training set and evaluated the performance on the test set. Our implementations are based on Pytorch 1.0 with one Nvidia Titan X.

We constructed FR-ResNet with different depths and evaluated accuracy performance on IP102 compared with ResNet baseline models. The results are reported in Table 4.3.9. Moreover, we compared FR-ResNet with several state-of-the-art models: AlexNet, ResNet-50, ResNet-101, Googlenet, VGG-16, and DeseNet121 to demonstrate their performance on IP102 dataset, and the results are reported in Table 4.3.10. As can be observed, compared with Table 4.3.9 and Table 4.3.10, 34-layer FR-ResNet had a test

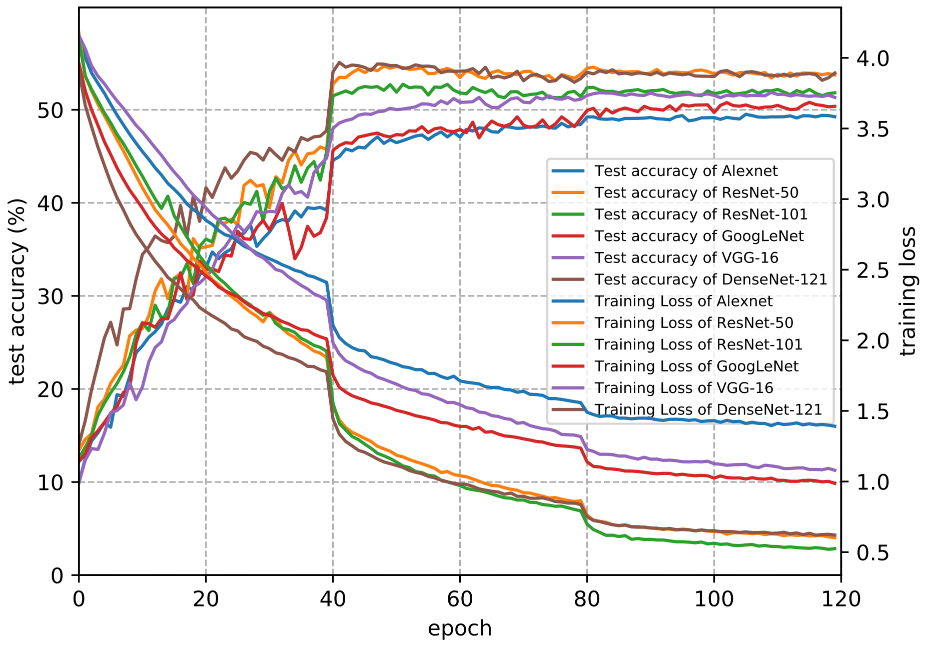


Fig. 4.3.7. Test accuracy on IP102 by several state-of-the-art method, corresponding to results in Table 4.3.10.

accuracy of 54.73% and 53.58 F1 score on test set, which outperformed all models performance in Table 4.3.10. The 50-layer FR-ResNet can acquire better performance than 34-layer FR-ResNet. As Fig. 4.3.6 and Table 4.3.7 shown, ResNet-101 can achieve lower training loss than ResNet-50, while its test accuracy is worse than ResNet-50 because the increased parameters led to overfitting. Through these experiments, it demonstrated the effectiveness of our method on IP102 dataset.

4.4 Discussion

We found the impact of Feature Reuse Residual unit is twofold. Fist, the feature from previous layers is used for subsequent layers. Second, feature reuse residual unit has a stronger capacity of representation than the original residual unit.

Tab. 4.3.10. Test accuracy(%) on IP102 by FR-ResNets.

IP102	# params	F1	Acc (%)
AlexNet [7]	57.42M	48.22	49.41
ResNet-50 [8]	23.72M	52.93	54.19
ResNet-101 [8]	42.63M	52.00	53.07
Googlenet [12]	10.24M	51.24	52.17
VGG-16 [11]	134.68M	51.20	51.84
DenseNet-121 [9]	7.06M	52.97	54.59

4.4.1 Feature Reuse

From Eqn.(4.1.3), the branched residual signal and the input signal of a residual block are concatenated before summation, and we realized the Feature Reuse Residual block based on the original residual block. BN and ReLU are omitted for simplifying. We can split the identity mapping into two parts. Therefore, as illustrated in Fig. 4.4.1(a), we could conjecture that each output of Feature Reuse Residual block contains two parts: one is from a residual block and the other is from the input directly. Through this method of learning half and reusing half, we enhanced the capacity of network and intensified the relationship of adjacent residual block. The experimental results in proceeding sections show our models outperformed the baseline models significantly with a similar total number of parameters or fewer parameters and demonstrated our suppose that reuse feature from previous layers in residual block can improve the performance on IP102 dataset.

4.4.2 Stronger Capacity of Representation

In the case of original ResNet and Pre-ResNet, each residual block contains two continuous 3×3 convolutional layers. While Feature Reuse Residual block contains three continuous 3×3 convolutional layers as shown in Fig. 4.1.1(b), and the results in Table 4.3.2, Table 4.3.3 and Table 4.3.4 show this structure can achieve a better performance with a similar total number of parameters. For each feature reuse residual block, the in-

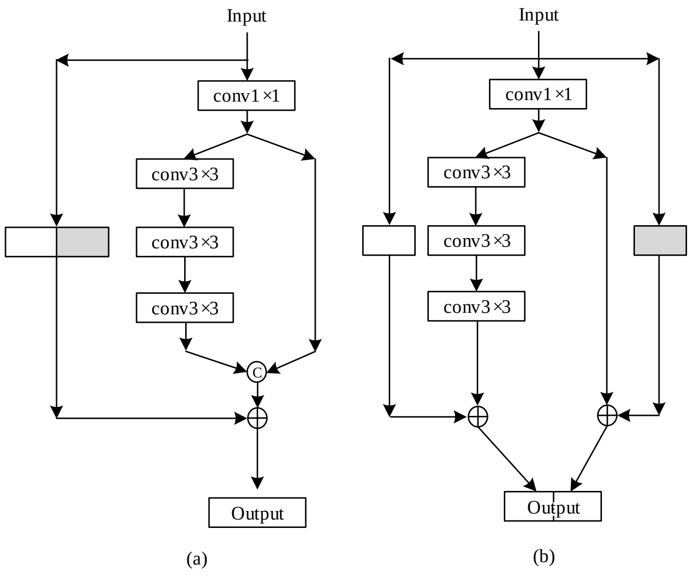


Fig. 4.4.1. Structure of feature reuse residual unit (a), (b) unraveled view of (a) showing that the output contains two parts: one is from a residual block and the other is from the input directly.

creased 3×3 convolutional layer with ReLU function enhance nonlinearity. However, the total number of ReLU unit in Feature Reuse Residual networks is less than the counterpart residual networks with a similar total number of parameters. In the case of WRN, the capacity of each wide residual unit is stronger than pre-activation residual block. After combined WRN with feature reuse method, the performance further enhanced with less ReLU unit, as shown in Table 4.3.3 and Table 4.3.4. So we can conclude that Feature Reuse Residual network have a stronger capacity of representation than their counterparts with less ReLU unit.

4.5 Summary

In this work, we proposed the feature reuse residual network (FR-ResNet) for image classification task. The central idea of the structure was described in this paper involves learning half and reuse half feature in each Feature Reuse Residual block. Based on the simple structure, we constructed the FR-ResNet and evaluated the classification performance on several benchmark datasets. The experimental results on these datasets showed that FR-ResNet could achieve better accuracy recognition performance compared with the baseline models.

Chapter 5

Deep Feature Fusion Residual Network

For FR-ResNet, we found that adding the model width will introduce more parameters, especially for high-resolution image classification tasks, which makes it impossible to construct deeper FR-ResNet to extract more features to enhance the model performance with a similar number of parameters. Therefore, to address this problem, we modified the original feature reuse residual block and proposed a new deeper feature fusion residual block to perform image classification tasks.

5.1 Methodology

In ResNets, the residual learning block with identity mapping can be formulated as follows,

$$y_l = h(x_l) + F(x_l, w_l) \tag{5.1.1}$$

$$x_{l+1} = f(y_l) \tag{5.1.2}$$

Here x_l and x_{l+1} refer to the input and output of the l-th residual block in the network. The function $h(x_l)$ refers to identity mapping: $h(x_l) = x_l$. F refers to the residual function, and

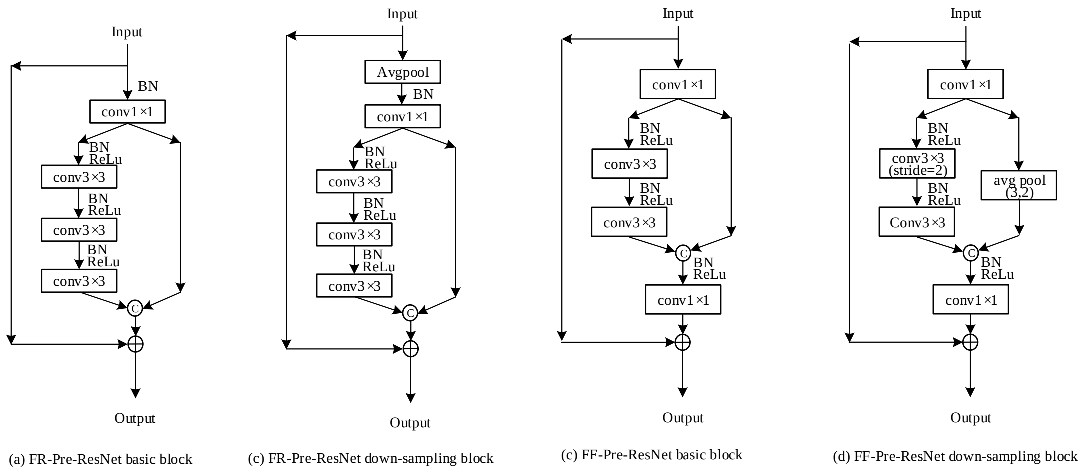

Input
BN
conv1×1
BN
ReLu
conv3×3
conv3×3
conv3×3
Avgpool
avg pool (3,2)
conv3×3 (stride=2)
Conv3×3
C
Output
(a) FR-Pre-ResNet basic block
(c) FR-Pre-ResNet down-sampling block
(c) FF-Pre-ResNet basic block
(d) FF-Pre-ResNet down-sampling block

Fig. 5.1.1. The architecture of Pre-ResNet and FF-Pre-ResNet.

w_l represents the parameters in the l-th residual block. The function of f expresses the ReLU.

In Pre-ResNet, both $h(x_l)$ and f are served as the identity mappings to transmit information through the network. The following computation can perform it:

$$x_{l+1} = h(x_l) + F(x_l, w_l) \tag{5.1.3}$$

Based on these works, we attempt to explore a validness feature fusion residual block. We follow the setting in Pre-ResNet by assigning both $h(x_l)$ and f serve as identity mappings and adopt the setting in Chapter 4 for FR-ResNet as shown in Fig. 5.1.1 (a) and (b). Then we proposed the feature fusion residual block, as shown in Fig. 5.1.1(c). Two 1×1 Conv layers are added into the residual block. One is used to balance the parameter of the two branches, and the other is used to reduce the channel dimension. Thus, it can be formulated as follows:

$$G(x_l) = F(g(x_l), w_l) o g(x_l) \tag{5.1.4}$$

$$x_{l+1} = h(x_l) + g'(G(x_l), w'_l) \tag{5.1.5}$$

Here o refers to the concatenate operation, and $G(x_l)$ denotes to the concatenated result. The function of g represents the first 1×1 convolution layer. The function of g', which is the second 1×1 convolution layer and realized by a BN-ReLU-Conv block, is used to halve the feature map dimension. The details will be described in the following section. Meanwhile, we also explore the number of 3×3 Conv layer in each residual block and the number of the feature fusion residual block in each group affected on model performance. The following section will extend the comparison of these matters.

5.2 Model Optimization

To max DFF-ResNet performance, we need to explore the important principles in two folds, including residual block size and the number of residual blocks in each residual group. Experiments are constructed on CIFAR-100 dataset to assess these principles.

To explore the the number of 3×3 Conv layer affected on model performance, we implemented some models with a different number of 3×3 Conv layers. We follow the setting in Chapter 4 and use $B(M)$ represents the residual block structure. We construct models with $B(1,3,1)$, $B(1,3,3,1)$, and $B(1,3,3,3,1)$. The results are reported in Fig. 5.2.1. As the result showed, the 218-layer DFF-Pre-ResNet and the 302-layer DFF-Pre-ResNet can achieve the best result as the structure is $B(1,3,3,1)$. Thus we choose $B(1,3,3,1)$ in the following experiments.

The original ResNet contains three residual groups with $2n$ layers in each residual group, and it has feature maps of sizes 32, 16, 8, respectively. In the down-sampling block, reducing the feature map size and increasing feature map dimensions are performed to keep a similar computational complexity. Consequently, each residual group has a similar computation complexity for ResNet. However, whether the contribution of each group to network performance is equal? Activated by this thought, we rethink the amount of residual blocks reasonability in each residual group. We construct different amounts of residual blocks in each group, and the experimental results show that adding

Tab. 5.2.1. The architecture of DFF-Pre-ResNets for CIFAR datasets.

Group	Group1(32x32)	Group2(16x16)	Group3(8x8)
# layers	$\lceil n*k*m \rceil$	$\lceil n*k \rceil$	n
# filters	16	32	64

the number of residual blocks in the earlier residual groups can increase the accuracy performance. Let us introduce the factors k and m, where k and m are the number of residual block multiple factors in different groups. Table 5.2.1 summarizes the architecture. The experimental results indicate that DFF-Pre-ResNet obtains the best performance when $k = 1.3$ and $m = 1.1$. The comparison and discussion of this matter are extended in the following section.

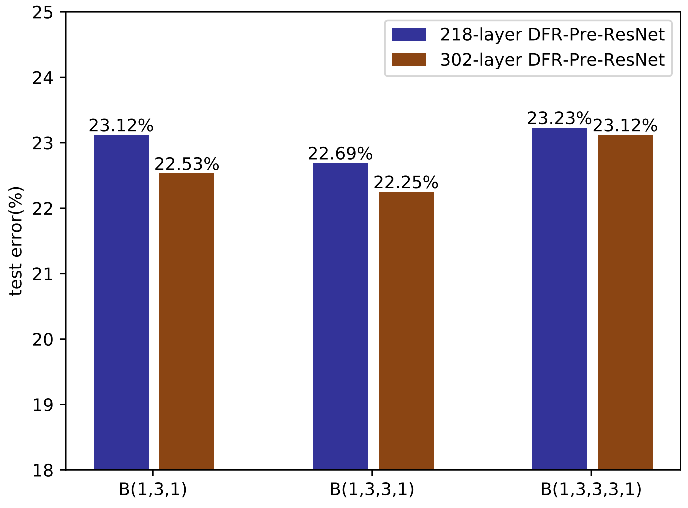


Fig. 5.2.1. The comparison result of different architecture on CIFAR-100 dataset. The structure of $B(1,3,3,1)$ achieves the best results for 218-layer and 302-layer DFF-Pre-ResNet.

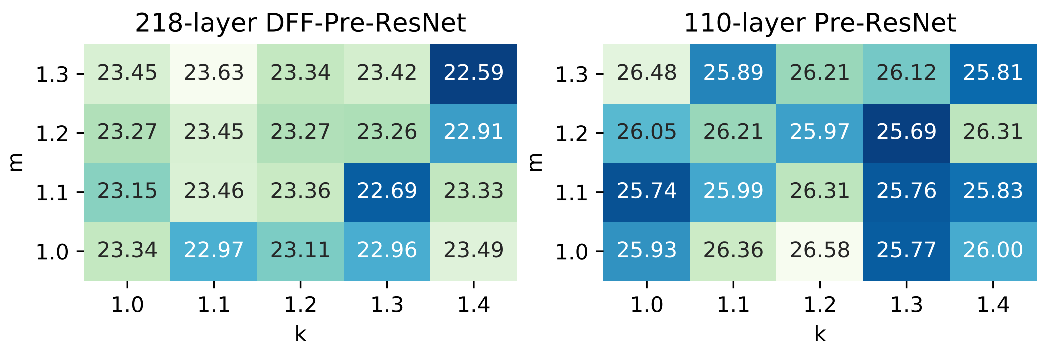


Fig. 5.3.1. Test error (%) of adding residual block in earlier residual groups applied to 218-layer DFF-Pre-ResNet and 110-layer Pre-ResNet on CIFAR-100 dataset under different hyper-parameters (k,m).

5.3 Experiments and Analysis

Some experiments are conducted to validate our proposed approach. First, we experimented with the impact of hyper-parameters of k and m to our model, and we test the model on CIFAR, SVHN, and IP102 benchmark datasets. These testing results verified the validness and adaptiveness of our method.

5.3.1 Influence of Hyper-parameters

In order to explore the influence of the two hyper-parameters (multiple ratio k and m) on test error , as shown in Fig. 5.3.1, we explored the ablation experiments on CIFAR-100 with different hyper-parameter settings. For a fair comparison, we implement the models under a similar total number of parameters. For 218-layer DFF-Pre-ResNet, the accuracy performance increased when the data range of k is between 1.1 and 1.3 compared with $k = 1.0$. It means that adding more residual blocks in earlier two residual groups bring benefits for DFF-Pre-ResNet. Meanwhile, if we only add residual blocks in the first group, it also brings a slight performance increase when m is 1.1 or 1.2 compared with $m = 1.0$. As Fig. 5.3.1 shown, the model achieves the best performance when the value of k and m are 1.4 and 1.3, respectively. However, the depth of the model becomes deeper when the k and m value increased under a similar total number of parameters, so it needs more time in each training epoch. When $k = 1.3$ and $m = 1.1$, it needs 64 seconds on Nvidia RTX 2080Ti. While, it needs 75 seconds as $k = 1.4$ and $m = 1.3$, and the test performance only achieves tiny improvement. Therefore, for the trade-off accuracy performance and training time, we choose $k = 1.3$ and $m = 1.1$ for 218-layer DFF-Pre-ResNet. We also use $k = 1.3$ and $m = 1.1$ for all experiments unless specified elsewhere. For comparison, we do the same experiments on 110-layer Pre-ResNet. As can be observed, it does not show the same pattern, and the accuracy performance increased is finite.

5.3.2 Implementation on CIFAR and SVHN datasets

The hyper-parameters setting is the same as in Chapter 4. For all datasets, we use SGD with batch-size of 128, 0.0001 weight decay, and 0.9 momentum. Kaiming Xavier algorithm [32] is used to initialize the weights. In case of CIFAR datasets, we adopted 0.1 as the initial learning rate and divided by a factor of 10 at 250th and 375th, ending at 500 epochs. In the case of SVHN dataset, the learning rate is set to be 0.1 and divided by a factor of 10 at 30th and 35th, ending at 50 epochs. The stochastic drop-path method is adopted to enhance test performance and alleviate overfitting. When depth exceeds 100 layers, we set p_l with the linear decay rule of $p_0 = 1.0$ and $p_l = 0.5$, and we set $p_0 = 1.0$ and $p_l = 0.8$ as the depth is less than 100 layers. In the following sections, we will use "SD" to denote training our model with the stochastic drop-path method.

5.3.3 CIFAR-10 Classification by DFF-ResNet

As shown in Table 5.3.1 and Fig. 5.3.2, we conducted different depth of DFF-Pre-ResNet, FR-Pre-ResNet, and Pre-ResNet, and the test error performance on CIFAR-10 dataset are reported. The 218-layer DFF-Pre-ResNet without SD achieved a competitive 4.18% test error on the test set, which outperformed the 110-layer Pre-ResNet without

Tab. 5.3.1. Comparison of test error (%) on CIFAR-10.

CIFAR-10	Depth	Params	Error(%)	Error(%)+SD
Pre-ResNets	110	1.7M	5.22	4.71
	164	2.6M	4.75	4.69
FR-Pre-ResNets	135	1.7M	4.41	4.35
	194	2.5M	4.36	3.90
DFF-Pre-ResNets	218	1.7M	**4.18**	**4.19**
	302	2.5M	**4.08**	**3.98**

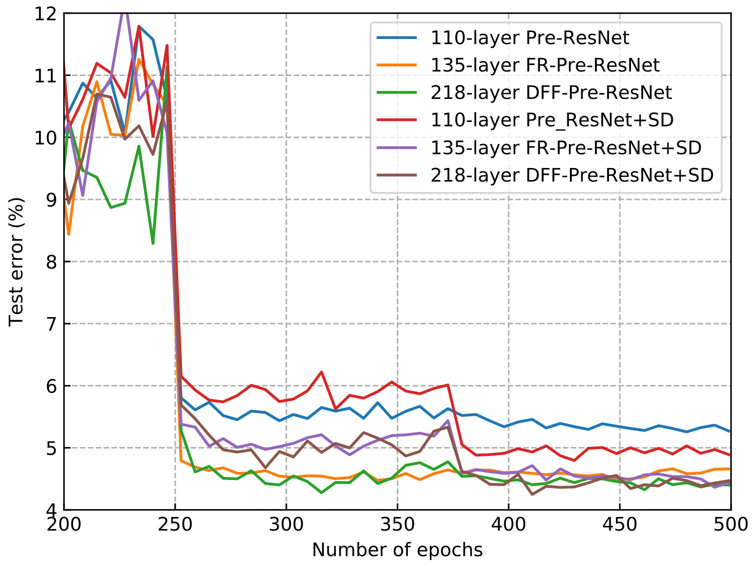


Fig. 5.3.2. Test error curves (smoothed) on CIFAR-10 by DFF-Pre-ResNet and baseline models during training period with corresponding results reported in Table 5.3.1. DFF-Pre-ResNet yields a lower test error than other models.

SD by 19.9% and 135-layer FR-Pre-ResNet without SD by 5.5%. The 218-layer DFF-Pre-ResNet with SD achieved a competitive 4.19% test error on the test set, which outperformed the 110-layer Pre-ResNet+SD by 11.0% and 135-layer FR-Pre-ResNet+SD

Tab. 5.3.2. Comparison of test error (%) on CIFAR-100.

CIFAR-100	Depth	Params	Error(%)	Error(%)+SD
Pre-ResNets	110	1.7M	25.93	23.99
	164	2.6M	25.06	22.98
FR-Pre-ResNets	135	1.7M	23.38	21.53
	194	2.5M	22.39	20.73
DFF-Pre-ResNets	218	1.7M	**22.69**	**20.92**
	302	2.5M	**22.25**	**20.53**

by 3.7%. Meanwhile, the 302-layer DFF-Pre-ResNet without SD achieved a similar situation compared to 194-layer FR-Pre-ResNet and 164-layer Pre-ResNet. For 302-layer DFF-Pre-ResNet+SD, it had a 3.98% test error on the test set, which was slightly worse than 194-layer FR-Pre-ResNet+SD but still outperformed the 164-layer Pre-ResNet with SD significantly. Depended on these experimental results and analysis, we can conclude that the DFF-Pre-ResNet has a stronger capacity than FR-Pre-ResNet and Pre-ResNet with a similar total number of parameters.

5.3.4 CIFAR-100 Classification by DFF-ResNet

In the case of the test error performance on CIFAR-100, the results are reported in Table 5.3.2 and Fig. 5.3.3. We also constructed the different depth of DFF-Pre-ResNet, FR-Pre-ResNet, and Pre-ResNet. The 218-layer and 302-layer DFF-Pre-ResNet without SD had 22.69% and 22.25% test error on the test set. The 218-layer and 302-layer DFF-Pre-ResNet+SD achieved a 20.92% and 20.53% test error on the test set, and they outperformed the 135-layer and 194-layer FR-Pre-ResNet+SD by 2.83% and 0.96%, respectively. In addition, the 218-layer and 302-layer DFF-Pre-ResNet also have significantly better performance than 110-layer and 164-layer Pre-ResNet, respectively.

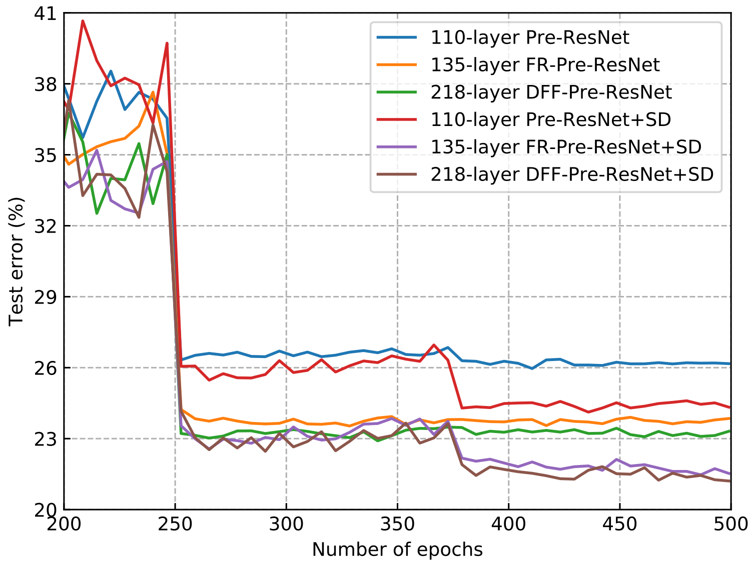


Fig. 5.3.3. Test error curves (smoothed) on CIFAR-100 by DFF-Pre-ResNet and baseline models during training period with corresponding results reported in Table 5.3.2. DFF-Pre-ResNet yields a lower test error than other models.

5.3.5 Deep Feature Fusion for WRN

In order to explore the performance influence for WRN, we constructed 52-layer DFF-WRN with different width, and the results are reported in Table 5.3.3, Fig. 5.3.4 and Fig. 5.3.5. As can be observed, DFF-WRN52-2 and DFF-WRN52-4 have lower test error than corresponding models with fewer parameter. The DFF-WRN52-4 had 4.08% and 20.73% test error on CIFAR-10 and CIFAR-100, respectively, which outperformed WRN40-4 with fewer parameter. Meanwhile, DFF-WRN52-4+SD had a 3.51% test error and 19.09% test error on CIFAR-10 and CIFAR-100, which outperformed WRN40-4+SD by 11.81% on CIFAR-10 and 5.87% on CIFAR-100. Depended on these analysis and experiments, the adaptiveness of our approach to WRN is demonstrated.

Tab. 5.3.3. Comparison of test error (%) on CIFAR-10 and CIFAR-100 datasets.

500 Epoch	Depth	Params	CIFAR-10		CIFAR-100	
			Error(%)	Error(%)+SD	Error(%)	Error(%)+SD
WRNs	40-2	2.2M	4.63	4.25	24.42	22.21
	40-4	8.9M	4.07	3.98	21.85	20.28
DFR-WRNs	49-2	2.2M	4.50	4.18	23.21	**21.45**
	49-4	8.7M	**3.99**	3.73	20.92	19.16
DFF-WRNs	52-2	2.0M	4.41	**3.93**	**23.19**	**21.45**
	52-4	7.9M	4.08	**3.51**	**20.73**	**19.09**

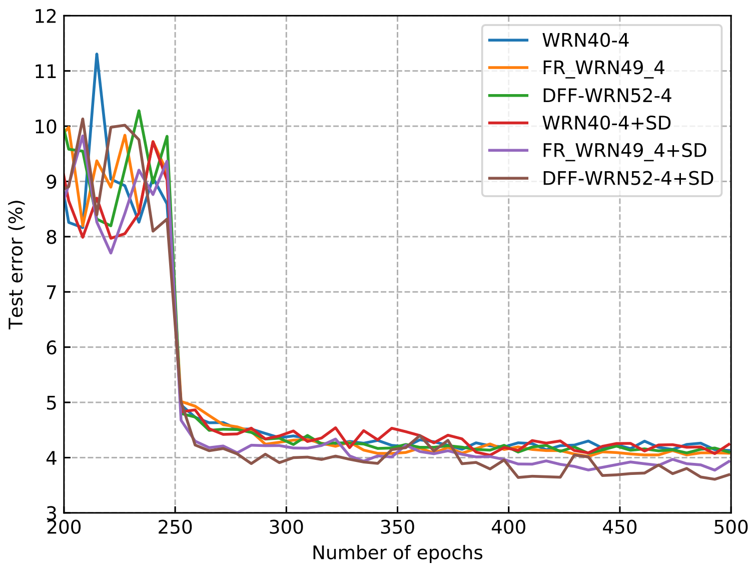


Fig. 5.3.4. Test error curves (smoothed) on CIFAR-10 by DFF-WRN and baseline models during training period with corresponding results reported in Table 5.3.3. DFF-WRN yields a lower test error than WRN.

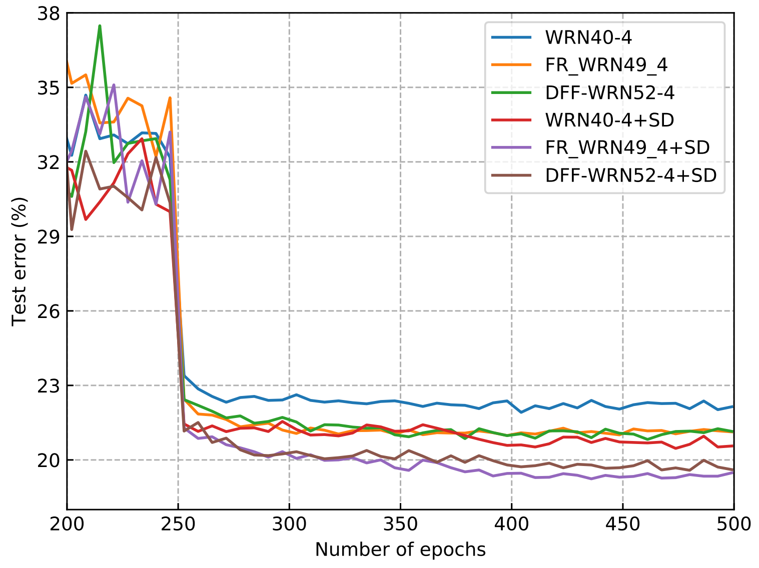


Fig. 5.3.5. Test error curves (smoothed) on CIFAR-100 by DFF-WRN and baseline models during training period with corresponding results reported in Table 5.3.3. DFF-WRN yields a lower test error than WRN.

5.3.6 Effect of Depth and Width

From the previous experimental results in this section, it indicates that increasing depth or width could improve the test error effectively. For the sake of investigating the influence of depth and width on DFF-ResNets, we did the following experiments.

In the case of DFF-Pre-ResNets, we constructed them with different depth to investigate the influence of depth to the models, and the test performance are represented in Table 5.3.4. It indicates that the test error gradually decreases on CIFAR datasets as layers increased. For extremely deep model, the 1050-layer DFF-Pre-ResNet achieved the 3.67% test error on CIFAR-10 and 18.71% test error on CIFAR-100. It demonstrates that adding the model depth can improve the model's test error performance effectively.

In the case of DFF-WRN, we constructed them depended on WRN and explored them with different width. The experimental results are showed in Table 5.3.5. As the

Tab. 5.3.4. Comparison of test error (%) on CIFAR-10 and CIFAR-100 by DFF-Pre-ResNet+SD with different depths.

Depth	CIFAR-10 DFF-Pre-ResNet	CIFAR-100 DFF-Pre-ResNet
218-layer	4.19	20.92
302-layer	3.98	20.53
378-layer	3.75	19.92
1050-layer	3.67	18.71

Tab. 5.3.5. Comparison of test error (%) on CIFAR-10 and CIFAR-100 by DFF-WRN+SD with different widths.

Width	CIFAR-10 DFF-WRN	CIFAR-100 DFF-WRN
DFF-WRN52-2	3.93	21.45
DFF-WRN52-4	3.51	19.09
DFF-WRN52-8	3.31	17.83
DFF-WRN52-8+mixup	2.42	15.59

width increased, the test error gradually decreased. The DFF-WRN52-8 had the 3.31% test error on CIFAR-10 and 17.83% test error on CIFAR-100. On the other hand, we adopt the mixup [102] augmentation methods to further improve the test error performance. The DFF-WRN52-8+SD+mixup achieved the 2.42% test error on CIFAR-10 and 15.59% test error on CIFAR-100.

These experiments further demonstrated that both increasing depth and width for DFF-ResNets could bring performance improvement. To achieve satisfying results, we should determine the depth and width carefully.

5.3.7 SVHN Classification Results

We also experimented with the model performance on SVHN dataset and chose DFF-WRN52-4 to compare with WRN. As the results in Table 5.3.6, DFF-WRN52-4+SD outperformed WRN40-4+SD by 12.0% on SVHN showed the test error curves. For DFF-

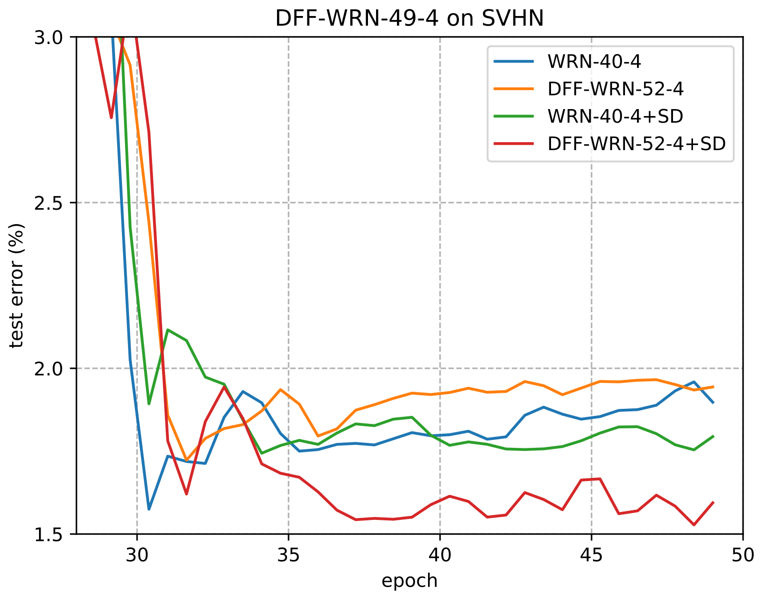


Fig. 5.3.6. Test error curves (smoothed) on SVHN by DFF-WRN and baseline models. The corresponding results are reported in Table 8.

WEN52-8, it can achieve a 1.53% test error on SVHN. These facts indicated that our model could also bring improvement on SVHN dataset.

5.3.8 Classification Result on IP102

We follow the dataset augmentation settings in FR-ResNet. In experiments, we first trained DFF-Pre-ResNets on the training set. Then, we evaluated the model on validation set to get the optimal model parameters. Last, the F1 score and accuracy performance are reported on the test set.

In the original ResNet for ImageNet, it contains four residual block groups. We follow this setting and construct DFF-Pre-ResNet with four residual block groups. For simplify, we only add the number of residual blocks in two earlier groups. As the results depicted in Table 5.3.7 and Fig. 5.3.7, we constructed the different depth of DFF-Pre-ResNet to compare with FR-ResNets and other state-of-the-art methods. The 62-layer

Tab. 5.3.6. Comparison of test error (%) on SVHN.

SVHN	Depth	Params	Error (%)	Error (%)+SD
WRN	40-4	8.9M	1.69	1.75
FR-WRN	49-4	8.7M	1.78	1.62
DFF-WRN	52-4	7.9M	1.76	**1.54**
	52-8	31.4M	-	**1.53**

Tab. 5.3.7. Comparison of F1 score and test accuracy (%) on IP102 by DFF-Pre-ResNets and other state-of-the-art methods.

IP102	# params	F1	Acc (%)
AlexNet [7]	57.42M	48.22	49.41
ResNet-50 [8]	23.72M	52.93	54.19
ResNet-101 [8]	42.63M	52.00	53.07
Googlenet [12]	10.24M	51.24	52.17
VGG-16 [11]	134.68M	51.20	51.84
DenseNet-121 [9]	7.06M	52.97	54.59
FR-ResNet-34	20.67M	53.58	54.73
FR-ResNet-50	30.78M	**54.18**	55.24
DFF-Pre-ResNet-62	22.54M	53.98	55.39
DFF-Pre-ResNet-82	30.20M	**54.18**	**55.43**

DFF-Pre-ResNet achieved 55.39% test accuracy and 53.98% F1 scores on the test set surpassing 50-layer Pre-ResNet by 1.76% and 1.55%, respectively. The 82-layer DFF-Pre-ResNet had the same F1 score compared with 50-layer FR-ResNet with fewer parameters, and it achieved a better accuracy performance. According to these facts, it indicated the validness of our approach to IP102 dataset.

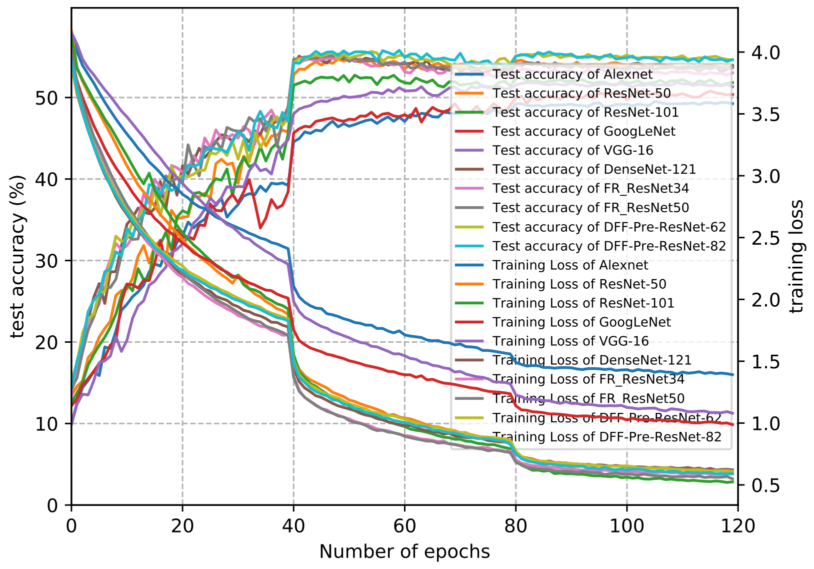


Fig. 5.3.7. Test accuracy and training loss curves on evaluation set during the training period.

5.4 Discussion

We found that the influence of DFF-ResNet is twofold. First, the feature fusion residual block make mode deeper to extract feature more validness. Second, adding residual blocks in earlier residual groups promote model generalization.

5.4.1 Effect of Feature Fusion Residual Block

ResNet has demonstrated that increasing the depth of the network can enhance model performance significantly. Depended on this hypothesis, in this paper, we proposed DFF-ResNet. As Eqn. 5.1.4, Equ. 5.1.5 and Fig. 5.1.1 shown, the feature fusion residual basic block adds two 1×1 Conv layer compared with basic residual block. Because of this modification, the feature fusion block could reach the same depth with fewer parameters. Thus, under the similar total number of parameters, we can construct a deeper model than FR-Pre-ResNet through stacking feature fusion residual block, which benefits to model performance. As the experiment results reported from Table 5.3.1 and

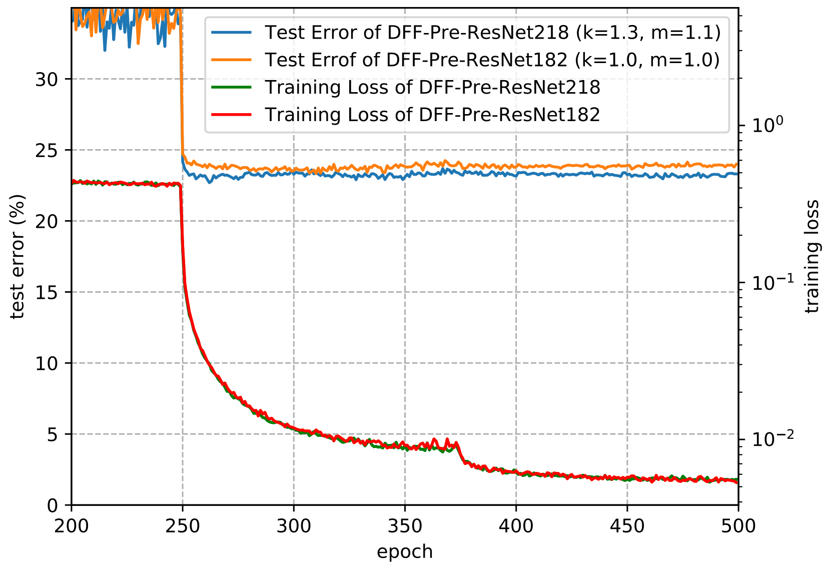


Fig. 5.3.8. Performance comparison between the 218-layer DFF-Pre-ResNet ($k = 1.3, m = 1.1$) and 182-layer DFF-Pre-ResNet ($k = 1.0, m = 1.0$), using CIFAR-100 dataset.

Table 5.3.2, these results demonstrate that DFF-ResNet outperforms the baseline models.

5.4.2 Effect of Adding Residual Blocks in Earlier Groups

For each residual group in the original ResNet, it has a similar computation complexity. We explored the effect of different numbers of residual blocks in each residual group. As Fig. 5.3.1 shown, adding the features from shallow residual groups can enhance the test error performance significantly, and the model had the best performance when $k = 1.3$ and $m = 1.1$. In order to explore the impact of adding residual blocks in earlier groups, we also compared the training loss and test error curves of DFF-Pre-ResNet with different of hyper-parameters of k and m, as shown in Fig. 5.3.8. For a fair comparison, we constructed the two models with the same number of parameters (1.7M). As can be observed, the 218-layer DFF-Pre-ResNet has superior test accuracy, and it demonstrated the greater ability to generalize compared to 182-layer DFF-Pre-ResNet. Thus, the results indicate that adding residual blocks in earlier residual groups can promote the model generalization ability.

5.5 Summary

In this work, the central idea of our model focus on making the model becoming deeper than FR-ResNet under a similar total number of parameters. Meanwhile, to further improve the model performance, we explored the influence of the number of residual blocks in earlier residual groups, which indicate that it could bring benefits to our models. We evaluated DFF-ResNet classification performance on several benchmark datasets with different depth. The results indicated that our models could achieve better performance than other state-of-the-art methods and FR-ResNets.

Chapter 6

Deep Multi-Branch Fusion Residual Network

Activated by the previous works, a new residual block are introduced to learn multi-scale representation in this part. In each block, it contains three branches: one is parameter-free, and the others contain several successive convolution layers. Moreover, an attention module is embedded into the new residual block to recalibrate the channel-wise feature response and to model the relationship of the three branches. By stacking this kind of block, we constructed the Deep Multi-branch Fusion Residual Network (DMF-ResNet).

6.1 Methodology

Learning multi-scale representation is deemed to be a valid method to improve model performance. Therefore, in order to further improve the capacity of the model, we combine two types of residual architectures with a parameter-free branch to learn multi-scale representation, as shown in Fig. 6.1.1(c), which is named as multi-branch fusion residual block. We use $x_l \in R^{H \times W \times C}$ denotes the l-th layer input feature map. Then, the following computation can perform it:

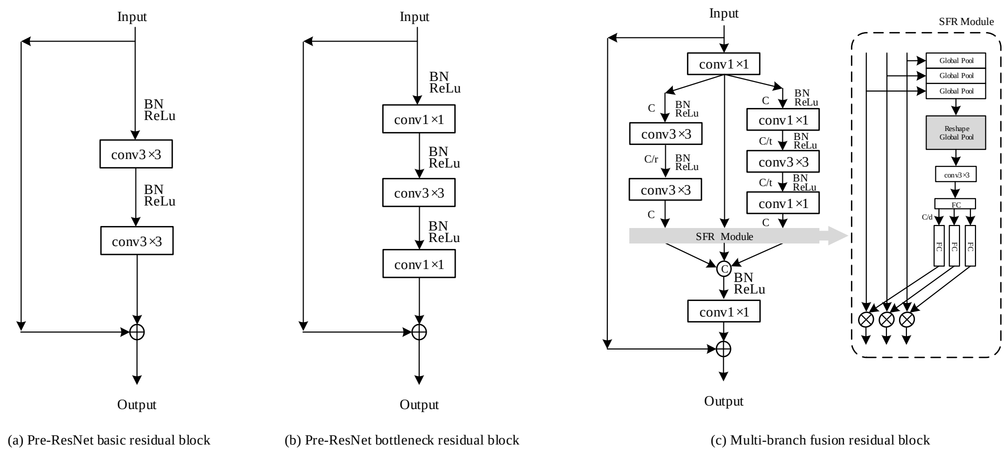


Fig. 6.1.1. Different residual block used in this paper.

$$u = g_1(x_l, \hat{w}_{1\times1}) \tag{6.1.1}$$

$$\begin{aligned} B_1 &= u \\ B_2 &= F_{basic}(u, w_{basic}) \\ B_3 &= F_{bottleneck}(u, w_{bottleneck}) \end{aligned} \tag{6.1.2}$$

Here the function g_1 denotes to the 1×1 convolution layer with the parameter of $\hat{w}_{1\times1}$. B_1, B_2, and B_3 refer to the extracted feature from three branches, and F_{basic} and $F_{bottleneck}$ refer to the residual functions of basic and bottleneck residual branches, respectively. w_{basic} and $w_{bottleneck}$ refer to the parameter of two residual branches. However, if we concatenated three branches directly, model performance can not achieve optimal results. The parameter proportion of different branches and the width of the model have significant impact on model performance. The comparison of these matters will be continued in the following section.

In order to further improve the capacity of multi-branch fusion residual block, we proposed a module to recalibrate channel-wise feature response adaptively and to model the relationship of the three branches, as shown in Fig. 6.1.1(c). This module realized this purpose by three operations - Squeeze, Fuse and Recalibrate, thus we named it as SFR module.

Squeeze. Following the setting in SENet [36], we also adopted the global average pooling to generate global information $b_k \in R^c, k = 1,2,3$ from each branch $B_k, k = 1,2,3$ in its spatial dimensions H×W, and the c-th element of b_k can be calculated by:

$$b_k^c = F_{gp}\left(B_k^c\right) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} B_k^c\left(i,j\right), k = 1,2,3 \tag{6.1.3}$$

Where F_{gp} denotes the global average function, $B_k = \left[B_k^1, B_k^2, ..., B_k^C\right], k = 1,2,3.$

Fuse. As shown in Fig. 6.1.1(c), the squeezed signals are concatenated as $\hat{s} = \left[{b_1}^T, {b_2}^T, {b_3}^T\right] \in R^{C\times 3}$. Then, $\hat{s}$ is reshaped to generate the folded feature map $\tilde{s} \in R^{\frac{C}{m}\times 3m}$, where the fold-ratio of m is used to control the shape of feature map. Subsequently, a 3×3 convolution kernel scans the folded feature map to enhance the nonlinear representation capacity,

$$\check{s} = F_w\left(\tilde{s}, w_{3\times 3}\right) \tag{6.1.4}$$

Where $\check{s} \in R^{C\times \frac{C}{m}\times 3m}$, F_w denotes to 3×3 convolution function, and $w_{3\times 3}$ denotes to the parameter of 3×3 convolution layer. Then, we obtain the mean reuslt in channel dimension, and the flatten layer is used to reshape the convolution results for subsequent FC layers, as $s = \left(F_{flatten}\left(\check{s}\right)\right)^T \in R^{3C}$. Further, a compact feature $z \in R^{\frac{C}{d}}$ is implemented to reduce the model complexity:

$$z = \tilde{F}_{fc}\left(s, W_1\right) = \delta\left(W_1 s\right) \tag{6.1.5}$$

Where $W_1 \in R^{\frac{C}{d}\times 3C}$, d is the reduction ratio to control the bottleneck structure, and δ refers to the relu function.

Recalibrate. As stated in previous section, our goal is to rescale the value for each channel and to model the relationship of three branches. Therefore, we implement three soft attention vectors $M_1, M_2, M_3 \in R^{C\times d}$ for B_1, B_2, B_3, respectively. Note that M_k^c is the

c-th row of M_k.

$$\begin{aligned} M_k = \hat{F}_{fc}(z, W_2) = \sigma(W_2 z) \\ \widetilde{B}_k^c = M_k^c \cdot B_k^c \end{aligned} \tag{6.1.6}$$

Here $W_2 \in R^{C \times \frac{C}{d}}$, $\widetilde{B}_k = \left[\widetilde{B}_k^1, \widetilde{B}_k^2, ..., \widetilde{B}_k^C\right], k = 1,2,3$. Then, $\widetilde{B}_1, \widetilde{B}_2, \widetilde{B}_3$ are concatenated together, as $\widetilde{B} = [\widetilde{B}_1, \widetilde{B}_2, \widetilde{B}_3]$. The concatenated feature pass through a 1×1 convolution layer to reduce the feature map dimension.

$$x_{l+1} = h(x_l) + g_2(\widetilde{B}, \check{w}_{1\times1}) \tag{6.1.7}$$

Where x_{l+1} refers to the output of the l-th residual block in the network, and g_2 refers to the function of the 1×1 convolution layer with the parameter of $\check{w}_{1\times1}$. The function $h(x_l)$ is an identity mapping: $h(x_l) = x_l$.

6.2 Model Optimization

For the sake of maxing DMF-ResNet model performance, some critical principles should be determined, including parameter proportion of different branch, and the width of the model. We test the model performance on the CIFAR-100 benchmark dataset to evaluate these principles.

Parameter proportion of different branches on model performance. In each multi-branch fusion residual block, the three branches represent different scale representation features. In these branches, one is parameter-free, and the other two branches contain several successive convolutional layers. However, as experimental results showed, simply concatenating these branches together can not achieve the superior effect. Because the parameter proportion of two residual signal branches have a significant on model performance, which means the feature extracted from the two residual branches need to be adjusted to max the capacity of the block. In the original ResNet, deeper bottleneck architecture has similar time complexity compared with basic residual architecture. The deeper bottleneck architecture uses a stack of 1×1, 3×3, and 1×1 convolutions, where

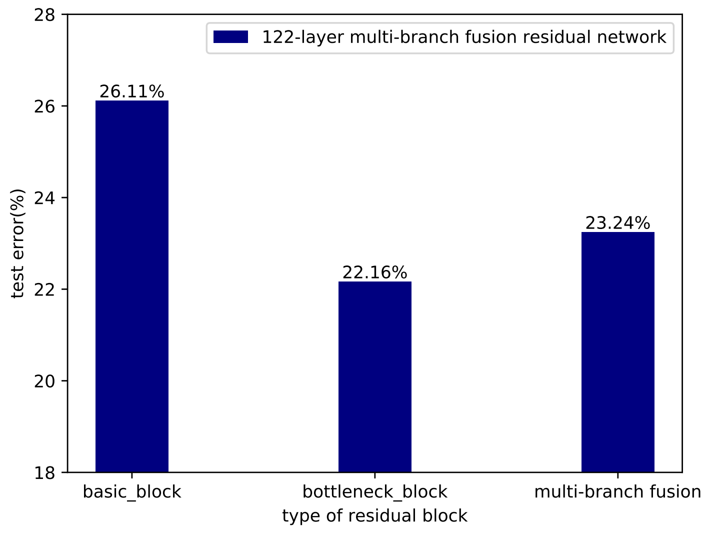


Fig. 6.2.1. Test error (%) on CIFAR-100. Multi-branch fusion architecture achieves lower test error than basic architecture, but still higher than bottleneck architecture.

1×1 layers are used to reduce and then increase dimension. Meanwhile, the input and output dimensions are expanded four times compared with the feature map dimension of basic residual architecture. In order to learn semantic information equally from each scale, we construct each branch having the same output feature map size and dimension, which leads to the bottleneck branch having fewer parameters compared with the basic residual architecture branch. In order to max the model performance, we should balance the number of parameters between the two residual signal branches. Let us introduce the factor r and t, as shown in Fig. 6.1.1(c), where r and t are ratios of reducing the feature map dimension for basic and bottleneck residual architecture branch, respectively. The experimental results indicate that DMF-ResNets achieve the best performance when $r = 2$ and $t = 4$, which means decrease the parameter proportion of the basic residual branch can improve the capacity of the block.

The impact of model width. Adjusting the parameter proportion from different branches can learn each scale representation effectively. However, as shown in Fig. 6.2.1,

Tab. 6.2.1. The test error (%) on CIFAR-100 with different width under a similar total number of parameters.

Width	Params	Error (%)
1	2.54M	23.24
2	2.45M	21.38
3	2.76M	22.22

we test these models with a similar number of parameters. The test error on CIFAR-100 of DMF-ResNet is lower than Pre-ResNet with basic architecture, while it is still higher than Pre-ResNet with bottleneck architecture. We conjectured that the mismatch between model width and model performance leads to this problem. In the original ResNet, to keep similar compute complexity, the feature map dimension of 1×1 convolution layers are expanded. Thus, the feature extracted capacity of the bottleneck branch in multi-branch fusion residual block is not maximized. In order to address this problem, we constructed the model with different width under a similar total number of parameters, and Table 6.2.1 shows the experimental results. As the results showed, the model achieves the best result when the model width is 2. Therefore, we adopt $width = 2$ for DMF-ResNet in the following experiments (except where otherwise stated).

6.3 Experiments and Analysis

We first reported the influence of hyper-parameters on model performance in this section. Then, we empirically demonstrated the effectiveness of our approach on CIFAR datasets and investigated the impact of model depth and width. Subsequently, we implemented some ablation experiments to verify the validness of multi-branch fusion and SFR module. Based on these explorations, we further constructed DMF-ResNets for high-resolution image classification tasks and evaluated these models' accuracy performance on IP102 dataset.

Tab. 6.3.1. Comparison the test error (%) of DMF-ResNet with different reducing ratio of r and t under a similar total number of parameters on CIFAR-100. The model achieves the best result as $r = 2$ and $t = 4$.

r \ t	1	2	4	8
1	22.66	22.37	22.61	22.41
2	22.69	21.77	**21.38**	21.82
4	22.37	21.72	21.80	21.92

6.3.1 Influence of hyper-parameters

For the sake of exploring the impact of the two hyper-parameters (reducing ratio r and t) on model performance, we constructed some ablation experiments on CIFAR-100 datasets under a similar total number of parameters. Meanwhile, to eliminate the interference, we constructed these models without the SFR module, and the experimental results are shown in Table 6.3.1. As the results showed, increasing the value of r and t at the same time can bring benefits to our model, and the model achieves the best test error result as $r = 2$ and $t = 4$. It means the multi-branch fusion residual block learns multi-scale representation more effectively with $r = 2$ and $t = 4$. Under this condition, we calculate the number of parameters for each residual branch. The result shows that the bottleneck branch extracts feature with fewer parameters than the basic branch. Thus, it demonstrated that the parameter effectiveness of the bottleneck branch is higher than the basic branch.

6.3.2 Implementations on CIFAR Datasets

To demonstrate the validness of our method, we first evaluated our models on CIFAR-10 and CIFAR-100 benchmark datasets. For CIFAR datasets, the weights are initialized by Kaiming Xavier algorithm [32], and all models adopted SGD algorithm. The momentum is 0.9, with a min-batch size of 128. The learning rate is set to be 0.1, and it drops to 1/10 and 1/100 at 150th and 225th, ending at 300 epochs. All experiments are constructed on Pytorch platform. Meanwhile, in multi-branch fusion residual block, the final settings of model performance depended on four hyper-parameters: the fold-ratio of m, the feature dimension reducing factor of r and t, and the reduction ratio of d. In our experiments, the model performance is not very sensitive to the fold-ratio of m. As the experimental results showed, the model obtains the best result when $m = 8$. Thus, we select $m = 8$ for CIFAR-10 and CIFAR-100 datasets in the following experiments. Following the setting of the reduction ratio in SENet [36], we set $d = 4$ for CIFAR-10 and CIFAR-100 datasets. The effect of the reduction ratio of r and t has been discussed in the previous section.

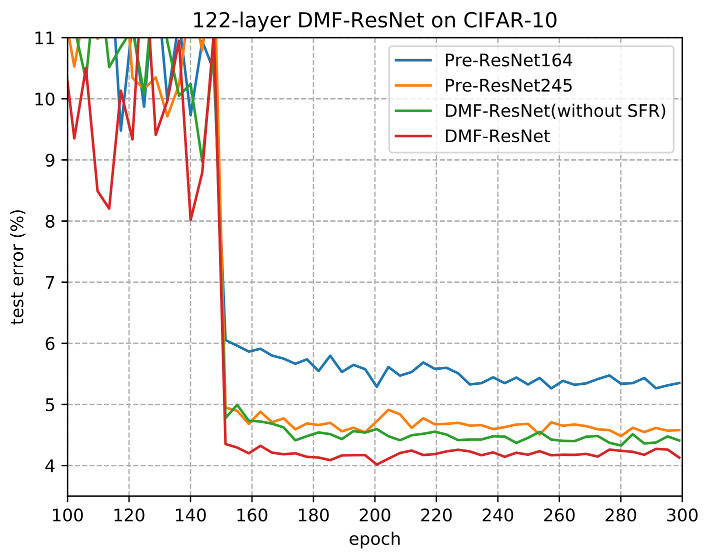


Fig. 6.3.1. Test error curves on CIFAR-10 by 164-layer Pre-ResNet, 245-layer Pre-ResNet, 122-layer DMF-ResNet without SFR module and 122-layer DMF-ResNet during training, corresponding to results in Table 6.3.2. The 122-layer DMF-ResNet (the red curve) is shown yielding a lower test error than other models.

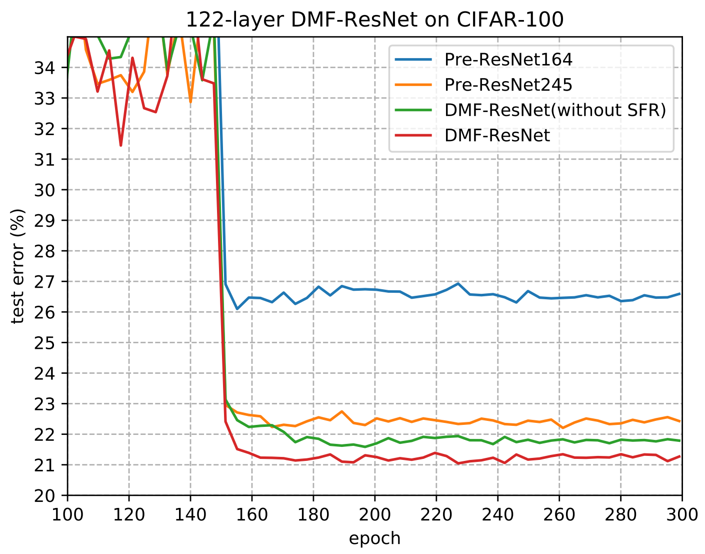


Fig. 6.3.2. Test error curves on CIFAR-100 by 164-layer Pre-ResNet, 245-layer Pre-ResNet, 122-layer DMF-ResNet without SFR module and 122-layer DMF-ResNet during training, corresponding to results in Table 6.3.2. The 122-layer DMF-ResNet (the red curve) is shown yielding a lower test error than other models.

6.3.3 CIFAR Classification by DMF-ResNet

The standard data augmentation strategy is used following the setting in previous experiments. For each image, 4 pixels are padded on each side to form an image with a size of 40×40. Then, a random 32×32 crop is applied to produce 32×32 images with horizontally mirroring half of the image. Mean and standard deviation normalization are also adopted.

We experimented with four models on CIFAR datasets: 164-layer Pre-ResNet with basic residual block, 245-layer Pre-ResNet with bottleneck residual block, 122-layer DMF-ResNet without SFR module, and 122-layer DMF-ResNet. All these models have a similar total number of parameters, and the test error performance and training curves on CIFAR datasets are showed in Fig. 6.3.1, Fig. 6.3.2, and Table 6.3.2. As the results shown, the 164-layer Pre-ResNet with basic residual block had a 5.23% and 26.11% test

Tab. 6.3.2. Test error (%) on CIFAR-100.

	Params	CIFAR-10 Error (%)	CIFAR-100 Error (%)
164-layer Pre-ResNet (basic architecture)	2.6M	5.23	26.11
245-layer Pre-ResNet (bottleneck architecture)	2.5M	4.42	22.16
122-layer DMF-ResNet (without SFR)	2.4M	4.32	21.38
122-layer DMF-ResNet	2.7M	**4.01**	**20.85**

error on CIFAR-10 and CIFAR-100, respectively. The 245-layer Pre-ResNet with bottleneck residual block achieved a competitive 4.42% and 22.16% test error on CIFAR-10 and CIFAR-100, respectively, which is better than Pre-ResNet with basic residual block. The 122-layer DMF-ResNet without SFR module outperformed 245-layer Pre-ResNet with bottleneck residual block by 0.1% on CIFAR-10 and 0.78% on CIFAR-100, which demonstrated that fusing the extracted feature from three branches can bring benefits for model performance. Meanwhile, the 122-layer DMF-ResNet with the SFR module achieved better test error performance than without the SFR module, and it achieved 4.01% and 20.85% test error on CIFAR-10 and CIFAR-100, respectively. Therefore, the results demonstrate that the SFR module also can bring improvements to our model.

6.3.4 The impact of depth and width

In order to explore the effect of depth and width on DMF-ResNet, we implemented the following experiments. These experiments are evaluated on CIFAR datasets, and Table 6.3.3 and Table 6.3.4 report the experimental results. As results showed, increasing depth or width could bring benefits to our models.

In terms of depth, we implemented 122-layer, 182-layer, 302-layer, and 1052-layer

Tab. 6.3.3. Test error (%) on CIFAR-10 and CIFAR-100 with different depth.

Depth	Params	CIFAR-10 Error (%)	CIFAR-100 Error (%)
122	2.7M	4.01	20.85
182	4.1M	3.97	20.77
302	6.8M	3.82	20.16
1052 (batch-size=32)	23.7M	3.63	18.73

Tab. 6.3.4. Test error (%) on CIFAR-10 and CIFAR-100 with different width.

Depth	Params	CIFAR-10 Error (%)	CIFAR-100 Error (%)
122-2	2.7M	4.01	20.85
122-4	10.8M	3.77	19.12
122-8	42.9M	3.52	17.51
302-4	26.9M	3.53	18.26
122-8+mixup	42.9M	2.60	16.88

DMF-ResNet to explore the influence of depth for our model. As results are showed in Table 6.3.3, the test error gradually decreased on CIFAR-10 and CIFAR-100 datasets when depth increased. The 302-layer DMF-ResNet had a 3.82% test error on CIFAR-10 test set and a 20.16% test error on CIFAR-100 test set. For extremely deep DMF-ResNet, due to the limited resource, the 1052-layer model was trained with a batch-size of 32, and it can still achieve a 3.63% test error on CIFAR-10 test set and 18.73% test error on CIFAR-100 test set. Based on these experimental results, we can conclude that increasing the depth can bring benefits for our model performance.

In terms of width, we implemented DMF-ResNet with a different width to explore the influence of width for our model. As the results are showed in Table 6.3.4, the test error gradually decreased on CIFAR-10, and CIFAR-100 datasets as the width increased on 122-layer DMF-ResNet. The DMF-ResNet-122-8 had a 3.52% test error on CIFAR-10 and 17.51% test error on CIFAR-100. Meanwhile, we also constructed the DMF-ResNet-302-4, and it had a 3.53% test error on CIFAR-10 test set and 18.26% test error on CIFAR-100 test set. Thus, the result indicates that increasing the depth and width at the same time can also improve model performance. On the other hand, augmentation methods, such as mixup [102], can further improve model performance. The DMF-ResNet122-8+mixup had the 2.60% test error and 16.88% test error on CIFAR-10 and CIFAR-100, respectively.

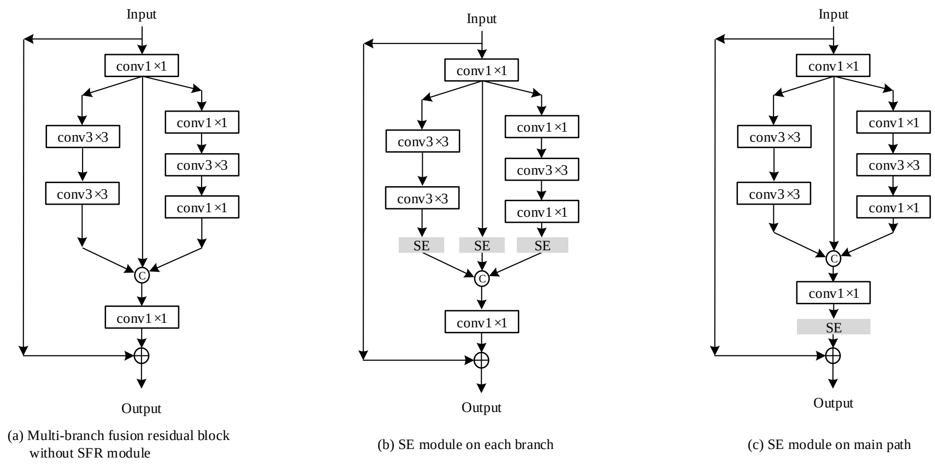


Fig. 6.3.3. Structure of multi-branch fusion residual block with the SE module in different localization for ablation study, corresponding to results in Table 6.3.6.

6.3.5 Ablation Study

For the sake of demonstrating the effectiveness of the multi-branch fusion and SFR module, we constructed some ablation experiments as follows. These experiments are evaluated on CIFAR-100 dataset.

Multi-branch Fusion. In DMF-ResNet, we fused three branches between two 1×1

Tab. 6.3.5. Test error (%) on CIFAR-100 to demonstrate the effectiveness of multi-branch fusion.

Model	Params	CIFAR-100 Error (%)
A (basic)	2.6M	26.11
B (basic, $r = 2$)	2.5M	25.62
C (basic, $r = 2, w = 2$)	2.5M	25.04
D (bottleneck)	2.6M	22.16
122-layer DMF-ResNet (without SFR)	2.4M	21.38

convolutional layers. One branch is parameter-free, and the other two are residual signals with basic residual block or bottleneck residual block. In order to demonstrate the effectiveness of the multi-branch fusion approach, we constructed four models to compare with 122-layer DMF-ResNet. Model A is the 164-layer Pre-ResNet with basic residual block. Model B is the 164-layer Pre-ResNet with basic residual block and channel reduction ratio of 2 ($r = 2$). Doubling the width of model B, we obtained model C. Model D is the 254-layer Pre-ResNet with bottleneck residual block. The results are reported in Table 6.3.5. All models are constructed under a similar total number of parameters. Compared to model A with model B, channel reduction between two 3×3 convolution layers make the model deeper under a similar number of parameters, which improves the model performance. Compared to model B with model C, increasing width has the same effect. However, model D had the lowest test error than the other three models. DMF-ResNet without SFR module contains the residual signal in models C and D, and it has a 21.38% test error on CIFAR-100 test set. The result outperforms model D by 0.78%. Therefore, it demonstrated the effectiveness of our multi-branch fusion approach.

Impact of SFR module. For the sake of verifying the effectiveness of the SFR module, we implemented some ablation experiments. We constructed three models to compare with DMF-ResNet, as shown in Fig. 6.3.3. To simplify, we omit the batch

Tab. 6.3.6. Test error (%) on CIFAR-100 to demonstrate the effectiveness of SFR module.

Network	Params	CIFAR-100 Error (%)
(a) without SFR module	2.5M	21.38
(b) SE on each branch	2.7M	21.12
(c) SE on main path	2.5M	21.66
122-layer DMF-ResNet	2.7M	20.85

normalization and relu layer. The first residual block is the multi-branch fusion residual block without the SFR module. The second residual block adds a SE block in each branch. The third residual block only adds a SE block in the main residual signal. All these models are constructed under a similar total number of parameters and tested model performance on CIFAR-100 dataset. Table 6.3.6 shows the experimental results. As the results showed, compared (a) with (c), adding a SE block in the main signal can not improve the model performance. Compared (a) with (b), adding a SE block in each branch can bring benefits to model performance. However, model (b) is short of modeling the relationship between three branches. The 122-layer DMF-ResNet had a 20.85% test error on CIFAR-100 test set, which outperforms model (b) by 0.27%. Based on these analyses, we can conclude that the SFR module can enhance our model performance effectively.

6.3.6 Classification results on IP102

Based on the previous experiments and analyses, to further demonstrate the effectiveness of our approach, we constructed DMF-ResNet for high-resolution image classification tasks. Then we applied it in a specific domain to recognize insect pests.

For the high-resolution image classification task, we implemented DMF-ResNet with different depths, which of the overall architectures are listed in Table 6.3.7. As described in the previous section, doubling the width of the model can reduce the test error significantly under a similar total number of parameters. However, increasing the width will introduce more parameters. Therefore, in order to control the number of parameters, we constructed these models with three stages, which is different from the original ResNet with four stages. We increased the number of multi-branch residual blocks in the second stage to construct models with different depths, including 77-layer, 97-layer, and 117-layer DMF-ResNet. Meanwhile, In our experiments, we select $m = 16$ in the following experiments. Following the setting of the reduction ratio in SENet [36], we set $d = 16$ for IP102 dataset.

We evaluated our models with different depths on IP102 dataset. We adopt 0.01 as the initial learning rate, and it drops to 1/10 and 1/100 at 40th and 80th, ending at 120 epochs. SGD is used with a batch-size of 32. The weight decay is set to be 0.0005, and the momentum is set to be 0.9. For data augmentation strategies, we adopt the following common approaches in the training phase. First, the image is randomly cropped a rectangular region, which is randomly sampled in [3/4, 4/3] and area randomly sampled in [0.08, 1] from resized 256×256 squared image. Second, the cropped region is resized into the size of 224×224. Third, randomly horizontal flips and standard deviation normalization are also applied in these experiments. During the evaluation period, the cropped 224×224 region from the center of the resized 256×256 image is used to classify. In these experiments, the model is trained on the training set and evaluated on the validation set to obtain the optimization model. Then we acquire the accuracy performance on the test set.

We implement three DMF-ResNets with different depths to compare with ResNet, Pre-ResNet, and other state-of-the-art methods. The accuracy performance on the test set is reported in Table 6.3.8, and Fig. 6.3.4 shows the test accuracy curves on the evaluation set during training. As the results showed, 77-layer DMF-ResNet achieved a 57.67 F1 scores and 58.48% test accuracy on the test set surpassing 50-layer ResNet 1.26 and 1.09% respective with fewer parameters. Meanwhile, the 77-layer DMF-ResNet achieved better model performance than other state-of-the-art methods. As the depth going deep, the DMF-ResNet test accuracy and F1 score increased. The 117-layer DMF-ResNet had a 58.37 F1 score and 59.22% test accuracy on the test set. Based on these experiments, we empirically demonstrated the validness of our approach to IP102 dataset.

For the sake of visualizing the effect of our approach, we use the technique of Grad-Cam [103] to highlight the important regions in the image for our insect pest classification task. To evaluate the effect of multi-fusion method and SFR module more clearly, we compare results from three models, including ResNet-50, DMF-ResNet without SFR module, and DMF-ResNet. We randomly select some images from IP102 dataset, and Table 6.3.9 presents the results. Compared ResNet-50 with DMF-ResNet without SFR module columns, the highlighted region in DMF-ResNet without SFR moduel column is wider than ResNet-50. It indicates that multi-scale learning obtained more abundant extracted features for the classification task. Compared DMF-ResNet without SFR module with DMF-ResNet columns, we can observe that the highlighted regions are finetuned to acquire more precise information for our task. Therefore, based on these analyses, we further demonstrate the effectiveness of our approach by visualizing the important regions for our task.

Tab. 6.3.7. Deep multi-branch fusion residual network architectures configuration.

layer name	output size	77-layer	97-layer	117-layer
conv1_x	112×112	7×7, 64, stride 2		
conv2_x	56×56	3×3, max pool, stride 2		
		$\begin{bmatrix}3\times3, 64\\3\times3, 128\end{bmatrix}\times5, \begin{bmatrix}1\times1, 32\\3\times3, 32\\1\times1, 128\end{bmatrix}\times5$	$\begin{bmatrix}3\times3, 64\\3\times3, 128\end{bmatrix}\times5, \begin{bmatrix}1\times1, 32\\3\times3, 32\\1\times1, 128\end{bmatrix}\times5$	$\begin{bmatrix}3\times3, 64\\3\times3, 128\end{bmatrix}\times5, \begin{bmatrix}1\times1, 32\\3\times3, 32\\1\times1, 128\end{bmatrix}\times5$
conv3_x	28×28	$\begin{bmatrix}3\times3, 128\\3\times3, 256\end{bmatrix}\times6, \begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times6$	$\begin{bmatrix}3\times3, 128\\3\times3, 256\end{bmatrix}\times10, \begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times10$	$\begin{bmatrix}3\times3, 128\\3\times3, 256\end{bmatrix}\times14, \begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times14$
conv4_x	14×14	$\begin{bmatrix}3\times3, 256\\3\times3, 512\end{bmatrix}\times4, \begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}3\times3, 256\\3\times3, 512\end{bmatrix}\times4, \begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}3\times3, 256\\3\times3, 512\end{bmatrix}\times4, \begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$
	1×1	average pool, 102-d fc, softmax		

Tab. 6.3.8. The F1 score and test accuracy (%) on IP102 dataset by DMF-ResNet and other state-of-the-art methods.

IP102	Depth	Params	F1	Acc (%)
AlexNet [7]	8	57.42M	48.08	49.63
ResNet-50 [8]	50	23.72M	56.41	57.39
ResNet-101 [8]	101	42.63M	55.37	56.02
Pre-ResNet-50 [14]	50	23.70M	55.18	55.86
VGG-16 [11]	16	134.68M	53.18	54.43
Densenet-121 [9]	121	7.06M	56.81	57.73
DMF-ResNet	77	22.10M	57.67	58.48
	97	25.96M	58.06	59.11
	117	29.70M	**58.37**	**59.22**

6.4 Discussion

In this section, we further discuss the effectiveness of our approach in two folds. First, we will discuss the effectiveness of the multi-branch fusion and SFR module. Second, we will discuss the parameter efficiency.

6.4.1 The effectiveness of the multi-branch fusion and SFR module

Compared with the original ResNet, DMF-ResNet combined the extracted feature from three branches to learn the multi-scale representation. The experimental results in Table 6.3.2 and Table 6.3.5 supported that learning multi-scale representation can enrich the feature for the classification task. Furthermore, the SFR module can further improve model performance. Through visualizing the effect of these approaches on some images, as shown in Table 6.3.9, the wider highlighted regions indicate that the receptive filed

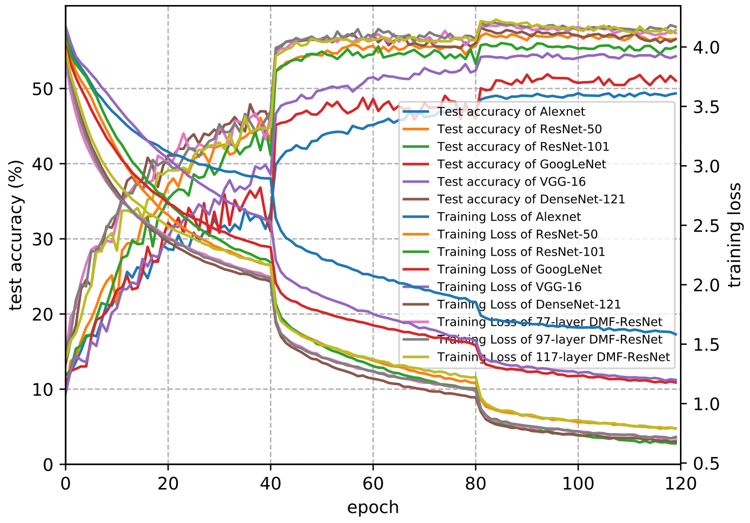
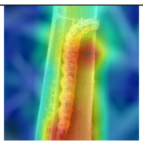
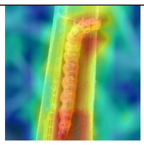
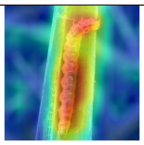
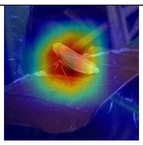
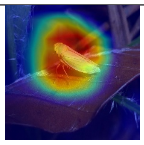
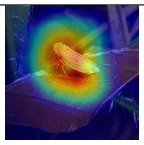
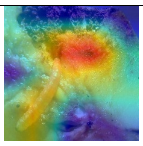
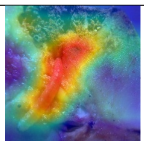
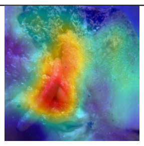
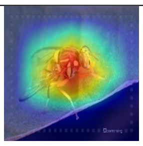
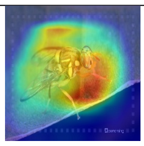
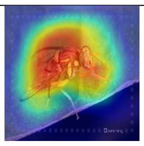


Fig. 6.3.4. The evaluation curves on IP102 dataset by DMF-ResNet and other state-of-the-art methods during training period.

Tab. 6.3.9. The highlighted important region.

ID	Original image	ResNet-50	DMF-ResNet (without SFR)	DMF-ResNet
1				
2				
3				
4				

is enlarged by learning the multi-scale representation. Meanwhile, the SFR module can finetune the response region by recalibrating channel-wise feature responses and modeling the relationship of these branches. Based on the result shown in Table 6.3.9, the highlight region is wider and precise. Therefore, it can be used to extract features in two-stage fine-grained image classification models to localize the object more accurately.

6.4.2 Parameter efficiency

In each multi-branch fusion residual block, we explored the different branches of parameter proportion on model performance, and the test error performance is reported in Table 6.3.1. As the results showed, the model achieves the best model performance as $r = 2$ and $t = 4$. Therefore, compared to the bottleneck branch with the basic branch, the ratio of the number of parameters in two branches is 1:9. Because these branches have the same feature dimension, thus each branch provided the same semantic information for model performance. Based on these analyses, we can obtain that the extracting feature capacity of the bottleneck branch is more effective than basic branch with fewer parameters. So we can conclude that a more effective convolution branch can further enhance the model performance, and the results also provide our direction to improve our model performance in the future.

6.5 Summary

In this work, to learn the multi-scale representation to improve the model performance, we fused the extracted feature from three branches in each residual block. Moreover, we proposed the SFR module to recalibrate channel-wise feature responses and to model the relationship between these branches. The experimental results verified the effectiveness of our approach on CIFAR-10 and CIFAR-100 datasets. Even for extremely deep DMF-ResNet, our model can achieve compelling results. Then, we constructed our model with different depths and tested the F1 score and model accuracy on IP102 dataset.

Compared to the baseline models and other state-of-the-art methods, our model can obtain the best model performance on IP102 dataset, which had proved the validness of our approach for the high-resolution image classification task. Through visualizing the highlighted regions on images, we can further explain the effect of our approach for the image classification task. Therefore, based on these empirical studies, we have verified the effectiveness of our approach.

Chapter 7

Conclusions and Future Work

In our thesis, we proposed the feature reuse residual network (FR-ResNet) for insect pest recognition. The central idea of the structure was described in our work involves learning half and reuse half feature in each Feature Reuse Residual block. Based on the simple structure, we constructed the FR-ResNet and evaluate the classification performance on IP102 dataset which is an challenging insect pest recognition benchmark dataset. The experimental results on IP102 showed that FR-ResNet can achieve better accuracy recognition performance compared with the baseline models. We also demonstrated that our approach can be adopted by other residual networks and outperform the original networks on CIFAR-10, CIFAR-100, and SVHN datasets. Through these empirical studies, the effectiveness of our approach was demonstrated, and this approach can be easily implemented in other residual networks.

To further enhance the model performance and get a tradeoff between test accuracy and model parameters, the deep feature fusion residual network (DFF-ResNet) is proposed. The central idea of our model focus on fusion feature from previous layer and making the model becoming deeper than FR-ResNet. Meanwhile, to further improve the model performance, we explored the influence of the number of residual blocks in earlier residual groups, which indicate that it could improve the test error performance effectively. For the sake of verifying the validness and adaptiveness of our approach, we followed the experimental strategies on FR-ResNet evaluated the test error performance on CIFAR and SVHN benchmark datasets with different width and depth. The experimental results indicated that our models could achieve better performance than baseline models. Besides, for high-resolution image classification task, our approach is applied to recognize insect pest and evaluated on IP102 benchmark dataset. As the testing performance showed, our models achieve better test accuracy performance than FR-ResNet model and other state-of-the-art approaches. Thus, based on above studies, It demonstrated the validness and adaptiveness of our approach, and it is convenient to be embedded into other common residual networks.

Based on the proposed residual networks and the attention mechanism methods, we proposed a new residual network to learn the multi-scale representation to improve the model performance. We fused the extracted feature from three branches in each residual block. Moreover, we proposed the SFR module to recalibrate channel-wise feature responses and to model the relationship between these branches. The experimental results verified the effectiveness of our approach on CIFAR datasets. Even for extremely deep DMF-ResNet, our model can achieve compelling results. Then, we constructed our model with different depths and tested the F1 score and model accuracy on IP102 dataset. Compared to the baseline models and other state-of-the-art methods, our model can obtain the best model performance on IP102 dataset, which had proved the validness of our approach for the high-resolution image classification task. Meanwhile, we visualized the highlighted regions on images to explain the effect of our approach for the image classification task. Therefore, based on these empirical studies, we have verified the effectiveness of our approach.

In future work, we will try to extend the proposed networks to different technical fields and explore more effective convolutional neural network structure.

Bibliography

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018. 1

[2] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture for generative adversarial networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 4401–4410. 1

[3] F. Ren and J. Deng, “Background knowledge based multi-stream neural network for text classification,” *Applied Sciences*, vol. 8, no. 12, p. 2472, 2018. 1

[4] W. Zhang, M. Wang, Y. Zhu, J. Wang, and N. Ghei, “A hybrid neural network approach for fine-grained emotion classification and computing,” *Journal of Intelligent & Fuzzy Systems*, vol. 37, no. 3, pp. 3081–3091, 2019. 1

[5] F. Ren and N. Liu, “Emotion computing using word mover’ s distance features based on ren_cecps,” *PloS one*, vol. 13, no. 4, p. e0194136, 2018. 1

[6] F. Ren, Y. Dong, and W. Wang, “Emotion recognition based on physiological signals using brain asymmetry index and echo state network,” *Neural Computing and Applications*, pp. 1–11, 2018. 1

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105. 1, 4.3.10, 5.3.7, 6.3.8

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. 1, 3.2, 3.5, 4.2, 4.3.10, 5.3.7, 6.3.8

[9] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2261–2269. 1, 2.4, 3.3, 4.3.10, 5.3.7, 6.3.8

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of The ACM*, vol. 60, no. 6, pp. 84–90, 2017. 1, 2.4, 3.1

[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. 1, 3.5, 4.3.10, 5.3.7, 6.3.8

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9. 1, 2.4, 2.4, 3.1, 3.5, 4.3.10, 5.3.7

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778. 1

[14] ——, "Identity mappings in deep residual networks," in *European conference on computer vision*. Springer, 2016, pp. 630–645. 1, 3.2, 4.3.4, 6.3.8

[15] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017. 1, 2.4

[16] Z. Shen, Z. Liu, J. Li, Y.-G. Jiang, Y. Chen, and X. Xue, “Dsod: Learning deeply supervised object detectors from scratch,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1919–1927. 1, 3.3

[17] S. Bell, C. Lawrence Zitnick, K. Bala, and R. Girshick, “Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2874–2883. 1

[18] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 6, pp. 1137–1149, 2016. 1

[19] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in *European conference on computer vision*. Springer, 2016, pp. 21–37. 1

[20] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440. 1

[21] F. Yu and V. Koltun, “Multi-scale context aggregation by dilated convolutions,” *arXiv preprint arXiv:1511.07122*, 2015. 1

[22] Y. Lu, Y. Chen, D. Zhao, and J. Chen, “Graph-fcn for image semantic segmentation,” in *International Symposium on Neural Networks*. Springer, 2019, pp. 97–105. 1

[23] H. Park, L. L. Sjösund, Y. Yoo, J. Bang, and N. Kwak, “Extremec3net: Extreme lightweight portrait segmentation networks using advanced c3-modules,” *arXiv preprint arXiv:1908.03093*, 2019. 1

[24] F. Ren and Z. Huang, "Automatic facial expression learning method based on humanoid robot xin-ren," *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 6, pp. 810–821, 2016. 1

[25] D. Feng and F. Ren, "Dynamic facial expression recognition based on two-stream-cnn with lbp-top," in *2018 5th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS)*. IEEE, 2018, pp. 355–359. 1

[26] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. 1, 3.1

[27] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833. 1

[28] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of physiology*, vol. 160, no. 1, p. 106, 1962. 1

[29] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986. 1

[30] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989. 2.1

[31] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018. 2.1, 3.1

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE*

international conference on computer vision, 2015, pp. 1026–1034. 2.1, 3.1, 4.3.1, 5.3.2, 6.3.2

[33] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004. 2.4, 3.5

[34] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 886–893. 2.4, 3.5

[35] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. 2.4, 3.1

[36] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141. 2.4, 2.4, 3.4, 6.1, 6.3.2, 6.3.6

[37] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015. 2.4, 3.1

[38] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826. 2.4

[39] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009. 2.5

[40] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011. 2.5

[41] X. Wu, C. Zhan, Y.-K. Lai, M.-M. Cheng, and J. Yang, "Ip102: A large-scale benchmark dataset for insect pest recognition," in *Proceedings of the IEEE Con-*

ference on Computer Vision and Pattern Recognition, 2019, pp. 8787–8796. 2.5, 3.5, 4.3.7

[42] V. Badrinarayanan, A. Kendall, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for image segmentation,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017. 3.1

[43] M. Lin, Q. Chen, and S. Yan, “Network in network,” *international conference on learning representations*, 2014. 3.1

[44] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, “Xnor-net: Imagenet classification using binary convolutional neural networks,” in *European conference on computer vision*. Springer, 2016, pp. 525–542. 3.1

[45] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, “Fast and accurate deep network learning by exponential linear units (elus),” *arXiv preprint arXiv:1511.07289*, 2015. 3.1

[46] L. Trottier, P. Gigu, B. Chaib-draa *et al.*, “Parametric exponential linear unit for deep convolutional neural networks,” in *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2017, pp. 207–214. 3.1

[47] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997. 3.1

[48] R. K. Srivastava, K. Greff, and J. Schmidhuber, “Highway networks,” *arXiv preprint arXiv:1505.00387*, 2015. 3.1

[49] F. Shen, R. Gan, and G. Zeng, “Weighted residuals for very deep networks,” in *2016 3rd International Conference on Systems and Informatics (ICSAI)*. IEEE, 2016, pp. 936–941. 3.2

[50] S. Zagoruyko and N. Komodakis, “Wide residual networks,” *arXiv preprint arXiv:1605.07146*, 2016. 3.2, 4.2, 4.3.4

[51] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," *european conference on computer vision*, pp. 646–661, 2016. 3.2, 4.3.1

[52] K. Zhang, M. Sun, T. X. Han, X. Yuan, L. Guo, and T. Liu, "Residual networks of residual networks: Multilevel residual networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 6, pp. 1303–1314, 2017. 3.2, 4.3.1

[53] D. Han, J. Kim, and J. Kim, "Deep pyramidal residual networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5927–5935. 3.2

[54] W. Liu, G. Wu, F. Ren, and X. Kang, "Dff-resnet: An insect pest recognition model based on residual networks," *Big Data Mining and Analytics*, vol. 3, no. 4, pp. 300–310, 2020. 3.2

[55] W. Liu, G. Wu, and F. Ren, "Deep multi-branch fusion residual network for insect pest recognition," *IEEE Transactions on Cognitive and Developmental Systems*, 2020. 3.2

[56] A. Veit, M. J. Wilber, and S. Belongie, "Residual networks behave like ensembles of relatively shallow networks," *Advances in neural information processing systems*, vol. 29, pp. 550–558, 2016. 3.2

[57] W. Nan, Z. Zhigang, M. Jingqi, L. Huan, L. Junyi, and Z. Zhenyu, "Face recognition method based on enhanced edge cosine loss function and residual network," in *2019 Chinese Control And Decision Conference (CCDC)*. IEEE, 2019, pp. 3320–3324. 3.2

[58] B. Fernando, E. Fromont, D. Muselet, and M. Sebban, "Discriminative feature fusion for image classification," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3434–3441. 3.3

[59] Q.-S. Sun, S.-G. Zeng, Y. Liu, P.-A. Heng, and D.-S. Xia, “A new method of feature fusion and its application in image recognition,” *Pattern Recognition*, vol. 38, no. 12, pp. 2437–2448, 2005. 3.3

[60] W. Song, S. Li, L. Fang, and T. Lu, “Hyperspectral image classification with deep feature fusion network,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 6, pp. 3173–3184, 2018. 3.3

[61] X. Cao, R. Li, L. Wen, J. Feng, and L. Jiao, “Deep multiple feature fusion for hyperspectral image classification,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 11, no. 10, pp. 3880–3891, 2018. 3.3

[62] Z. Li and F. Zhou, “Fssd: feature fusion single shot multibox detector,” *arXiv preprint arXiv:1712.00960*, 2017. 3.3

[63] W. Guan, Y. Zou, and X. Zhou, “Multi-scale object detection with feature fusion and region objectness network,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 2596–2600. 3.3

[64] J. Chu, Z. Guo, and L. Leng, “Object detection based on multi-layer convolution feature fusion and online hard example mining,” *IEEE Access*, vol. 6, pp. 19 959–19 967, 2018. 3.3

[65] P. Zhang, W. Liu, Y. Lei, and H. Lu, “Hyperfusion-net: Hyper-densely reflective feature fusion for salient object detection,” *Pattern Recognition*, vol. 93, pp. 521–533, 2019. 3.3

[66] Z. Zhang, X. Zhang, C. Peng, X. Xue, and J. Sun, “Exfuse: Enhancing feature fusion for semantic segmentation,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 269–284. 3.3

[67] S.-J. Park, K.-S. Hong, and S. Lee, “Rdfnet: Rgb-d multi-level residual feature fusion for indoor semantic segmentation,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 4980–4989. 3.3

[68] S. Lee, S.-J. Park, and K.-S. Hong, “Rdfnet: Rgb-d multi-level residual feature fusion for indoor semantic segmentation,” in *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2017, pp. 4990–4999. 3.3

[69] M. Gao, H. Chen, S. Zheng, and B. Fang, “Feature fusion and non-negative matrix factorization based active contours for texture segmentation,” *Signal Processing*, vol. 159, pp. 104–118, 2019. 3.3

[70] S. Chaib, H. Liu, Y. Gu, and H. Yao, “Deep feature fusion for vhr remote sensing scene classification,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 8, pp. 4775–4784, 2017. 3.3

[71] F. Ni, J. Zhang, and Z. Chen, “Pixel-level crack delineation in images with convolutional feature fusion,” *Structural Control and Health Monitoring*, vol. 26, no. 1, p. e2286, 2019. 3.3

[72] S. Chaib, H. Liu, Y. Gu, and H. Yao, “Deep feature fusion for vhr remote sensing scene classification,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 8, pp. 4775–4784, 2017. 3.3

[73] G. Huang, S. Liu, L. Van der Maaten, and K. Q. Weinberger, “Condensenet: An efficient densenet using learned group convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2752–2761. 3.3

[74] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, “Shufflenet v2: Practical guidelines for efficient cnn architecture design,” in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 116–131. 3.3

[75] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008. 3.4

[76] T. Bluche, "Joint line segmentation and transcription for end-to-end handwritten paragraph recognition," in *Advances in Neural Information Processing Systems*, 2016, pp. 838–846. 3.4

[77] J. Libovickỳ and J. Helcl, "Attention strategies for multi-source sequence-to-sequence learning," *arXiv preprint arXiv:1704.06567*, 2017. 3.4

[78] T. Shen, T. Zhou, G. Long, J. Jiang, S. Wang, and C. Zhang, "Reinforced self-attention network: a hybrid of hard and soft attention for sequence modeling," *arXiv preprint arXiv:1801.10296*, 2018. 3.4

[79] C. Cao, X. Liu, Y. Yang, Y. Yu, J. Wang, Z. Wang, Y. Huang, L. Wang, C. Huang, W. Xu *et al.*, "Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2956–2964. 3.4

[80] J. Choe and H. Shim, "Attention-based dropout layer for weakly supervised object localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2219–2228. 3.4

[81] ——, "Attention-based dropout layer for weakly supervised object localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2219–2228. 3.4

[82] Y. Zhu, J. Xie, Z. Tang, X. Peng, and A. Elgammal, "Semantic-guided multi-attention localization for zero-shot learning," in *Advances in Neural Information Processing Systems*, 2019, pp. 14 943–14 953. 3.4

[83] L. Chen, H. Zhang, J. Xiao, L. Nie, J. Shao, W. Liu, and T.-S. Chua, "Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5659–5667. 3.4

[84] L. Huang, W. Wang, J. Chen, and X.-Y. Wei, "Attention on attention for image captioning," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 4634–4643. 3.4

[85] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang, "Bottom-up and top-down attention for image captioning and visual question answering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6077–6086. 3.4

[86] M. Pedersoli, T. Lucas, C. Schmid, and J. Verbeek, "Areas of attention for image captioning," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1242–1250. 3.4

[87] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, "Residual attention network for image classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3156–3164. 3.4

[88] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *European conference on computer vision.* Springer, 2016, pp. 483–499. 3.4

[89] M. D. Zeiler and R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," *arXiv preprint arXiv:1301.3557*, 2013. 3.4

[90] X. Li, W. Wang, X. Hu, and J. Yang, "Selective kernel networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 510–519. 3.4

[91] S. Woo, J. Park, J.-Y. Lee, and I. So Kweon, “Cbam: Convolutional block attention module,” in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 3–19. 3.4

[92] B. Espejo-Garcia, N. Mylonas, L. Athanasakos, S. Fountas, and I. Vasilakoglou, “Towards weeds identification assistance through transfer learning,” *Computers and Electronics in Agriculture*, vol. 171, p. 105306, 2020. 3.5

[93] J. Haupt, S. Kahl, D. Kowerko, and M. Eibl, “Large-scale plant classification using deep convolutional neural networks.” in *CLEF (Working Notes)*, 2018. 3.5

[94] M. M. Ghazi, B. Yanikoglu, and E. Aptoula, “Plant identification using deep neural networks via optimization of transfer learning parameters,” *Neurocomputing*, vol. 235, pp. 228–235, 2017. 3.5

[95] P. A. Dias, A. Tabb, and H. Medeiros, “Apple flower detection using deep convolutional networks,” *Computers in Industry*, vol. 99, pp. 17–28, 2018. 3.5

[96] H. Zhang, G. He, J. Peng, Z. Kuang, and J. Fan, “Deep learning of path-based tree classifiers for large-scale plant species identification,” in *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE, 2018, pp. 25–30. 3.5

[97] S. W. Chen, S. S. Shivakumar, S. Dcunha, J. Das, E. Okon, C. Qu, C. J. Taylor, and V. Kumar, “Counting apples and oranges with deep learning: A data-driven approach,” *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 781–788, 2017. 3.5

[98] J. B. Castro, R. Q. Feitosa, and P. N. Happ, “An hybrid recurrent convolutional neural network for crop type recognition based on multitemporal sar image sequences,” in *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, 2018, pp. 3824–3827. 3.5

[99] R. Li, R. Wang, J. Zhang, C. Xie, L. Liu, F. Wang, H. Chen, T. Chen, H. Hu, X. Jia *et al.*, “An effective data augmentation strategy for cnn-based pest localization and recognition in the field,” *IEEE Access*, vol. 7, pp. 160 274–160 283, 2019. 3.5

[100] K. Dimililer and S. Zarrouk, “Icspi: Intelligent classification system of pest insects based on image processing and neural arbitration,” *Applied Engineering in Agriculture*, vol. 33, no. 4, p. 453, 2017. 3.5

[101] Z. Liu, J. Gao, G. Yang, H. Zhang, and Y. He, “Localization and classification of paddy field pests using a saliency map and deep convolutional neural network,” *Scientific reports*, vol. 6, p. 20410, 2016. 3.5

[102] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, “mixup: Beyond empirical risk minimization,” *arXiv preprint arXiv:1710.09412*, 2017. 5.3.6, 6.3.4

[103] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-cam: Visual explanations from deep networks via gradient-based localization,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626. 6.3.6