Journal of Computers Vol. 36 No. 6, December 2025, pp. 73-85
https://doi.org/10.63367/199115992025123606006

A Novel Lightweight Convolutional Neural Network Model with An Integration of Residual and Inception Network

Xiao-Qing Lin*

School of Information Engineering, Liaodong University,
Dandong, 118003, China

linxiaoqing@liaodongu.edu.cn

Received 2 August 2024; Revised 11 August 2024; Accepted 12 August 2024

Abstract. A new deep convolutional network model (ResNet-Inception, we called) is built in the paper, that combines the advantages of residual network (ResNet) and Inception architecture. The Inception core of ResNet uses convolution kernels of different sizes within the same network to enhance the model's perceptiveness and batch normalization is used to alleviate gradient vanishing. The residual connections allow the network to directly learn the difference between input and output, rather than directly learning complex mapping functions, which simplifies the learning process and improves the model's expressive power. By preserving the original feature information, residual connections help the model better generalize to unseen data. We designed a new convolutional neural network (CNN) by an integration of ResNet and Inception blocks, aiming to reduce the number of parameters while improving the recognition performance of the network. We give clear empirical evidence that training with Inception modules accelerates the training of residual networks significantly. With an ensemble of ResNet-18 and Inception blocks, we outperform the other two compared network structures, pure Inception network, Inception-v1, pure residual network, ResNet-18 on Cifar-10 dataset.

Keywords: CNN model, data augmentation, dropout regularization

1 Introduction

With the continuous development of computer hardware and the increasing maturity of deep learning theory, CNNs have experienced rapid growth. CNN is widely used in fields such as computer vision and natural language processing as well as in fields such as healthcare, finance, and intelligent transportation. CNN is a feedforward neural network with convolutional operations and deep structure in the network. Essentially, it is a multi-layer neural network structure, consisting of an input layer, an intermediate layer (a hidden layer with multiple layers), and an output layer. Neurons in the network respond within a certain range. The domain of neurons is particularly prominent in image processing. The multi-layer network structure gradually combines the low-level features of the image through convolution operations to form high-level features. The high-level semantic information is extracted from local information, forming a multi-layer transmission and gradual fusion process. The feature extraction and classification recognition are connected together to achieve the task of image recognition.

The limited quantity of training data can hamper supervised machine learning, that generally need large amounts of data to avoid overfitting [1-2]. Thus, the small number of datasets used poses a huge challenge for training CNNs. Data augmentation is the process of increasing the quantity and diversity of available data. In addition, it also helps teach models about invariance in the data domain [3]. The work in [4] generate artificial images from the existing ones through methods such as moving, changing width and height of the original image. A scale aware data augmentation approach was proposed for scene text detection, which can achieve better generalization ability and more accurate localization [5]. Data augmentation can also be effectively used to combine data invariance and significantly promote the generalization of existing deep learning models [6]. Now, the highest ranked models on the MNIST dataset widely use data augmentation techniques such as scaling and rotation to improve model performance [7-8]. In [9], the author demonstrated how to use simple geometry rotation and translation transformations can be misleading. The autonomous vehicle platform in CNN [10] may be

* Corresponding Author

deceived into turning right and hitting the guardrail by mistake, just slightly darken the input image. Therefore, designing data augmentation methods correctly is crucial for improving model generalization ability and enhancing security. The number of convolutional layers built is relatively small, and the parameters that can be utilized and learned in each layer are also reduced. For limited data, the ability to generalize similar parameters for estimating new image data has high requirements. By using data augmentation and regularization methods [11], it is possible to perform rotation, cropping, and other operations on the dataset when the amount of data is small, thereby expanding the current dataset. Using the random deactivation, during the learning process of CNNs, some features are randomly activated for training, which can effectively generalize reconfigurable CNNs. The main features are learned according to requirements, and high-precision recognition of targets is performed.

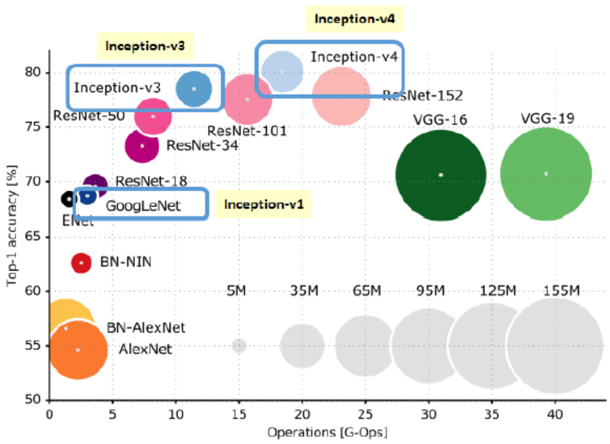


Fig. 1. Classification accuracy commonly used in CNNs [12]

As mentioned in ResNet [13], it does not mean that the deeper the network, the better the effect. We were inspired by Fig. 1. The networks in the upper left corner are the best choices, including ResNet and Inception series. Although VGG network has high accuracy, it requires a large amount of computation. Therefore, we built a new deep CNN model by integrating Inception modules into ResNet, aiming to reduce the number of parameters while increasing network recognition performance. Due to the recognition accuracy of CNNs and potential overfitting issues in small-scale datasets, data augmentation techniques [2] have been implemented to improve the problem of overfitting. A fully developed and reconfigurable CNN model can be applied to small datasets to complete training based on small-scale datasets. After a series of convolutional operations, the features of the target in the input image are extracted and subjected to dimensionality reduction processing in each pooling layer before being passed to the final fully connected layer for integration and classification.

The rest of the paper is arranged as follows: Section 2 introduces the related work. Section 3 describes ResNet-Inception model. Section 4 analyzes the experimental results on the dataset of Cifar-10. Section 5 concludes.

2 Related Work

In the section, we describe the classical CNNs. Fig. 2 shows the development of CNNs. In the past few years, researchers have studied the typical architecture and optimization schemes of CNN. It is reviewed that typical CNN models and their basic components from 2012 to 2015 [14]. There are also some well-known studies discussing different CNN algorithms and their applications in [15-19], it is summarized that various latest CNN architectures based on the design patterns of processing units. In [20-21], CNN classification based on acceleration techniques are analyzed.

We would like to stress that the class/style files and the template should not be manipulated and that the guidelines regarding font sizes and format should be adhered to. This is to ensure that the end product is as homogeneous as possible.

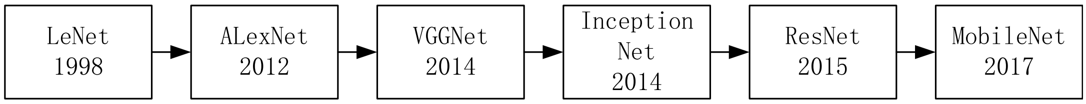


Fig. 2. The development of classic CNNs

LeNet [15] was proposed by LeCun in 1998 and is the opening work of convolutional networks. By sharing convolutional kernels, the parameters of the network are reduced. When counting the number of layers in a CNN, only the convolutional calculation layer and the fully connected calculation layer are generally counted. The remaining operations can be considered as appendages of the convolutional computation layer, and LeNet has a total of five layers of network. AlexNet [22] is a type of CNN proposed by Alex et al. in the 2012 ImageNet competition. They won the championship with this model, proving the effectiveness of CNN in complex models. Using GPU training can achieve results within an acceptable time range, promoting the development of supervised learning.

VGGNet [23] was born in 2014 and was the runner up in the ImageNet competition, with a top-5 error rate reduced to 7.3%. By using small convolution kernels, the recognition accuracy is improved while reducing parameters. The network structure of VGGNet is well-organized and very suitable for hardware acceleration. InceptionNet [24] was born in 2014 and was the champion of the ImageNet competition that year, with a top-5 error rate of 6.67%. InceptionNet introduces Inception structure blocks and uses multiple sizes of convolution kernels within the same layer of the network to extract features of different sizes, improving the model's perceptiveness. Batch normalization is used to alleviate gradient vanishing. The core of InceptionNet is its basic unit, the Inception structure block. Both GoogleNet, also known as Inception v1, and subsequent versions of InceptionNet such as v2, v3, v4, are networks built on Inception building blocks which apply 1×1 convolution kernels to each pixel of the input feature map, and set the number of 1×1 convolution kernels to be less than the depth of the input feature map.

ResNet [13] was the champion of the ImageNet competition in 2015, with a top-5 error rate of 3.5%. It is believed that simply stacking network layers will cause neural network degradation, resulting in the loss of the original appearance of the previous features in the later features. Therefore, a skip line is used to directly connect the previous features to the later ones, and output a nonlinear output containing convolution. ResNet proposes inter layer residual skip connections, introduces forward information, alleviates gradient vanishing, and makes it possible to increase the number of neural network layers. The '+' in the Inception block is stacked along the depth direction, similar to increasing the number of layers in a thousand-layer cake. The '+' in ResNet is the sum of the corresponding element values in two feature maps, which is equivalent to adding the corresponding element values in two matrices. The shortcut connections add neither extra parameter nor computational complexity. MobileNet [25] is a small CNN proposed by Google in 2017, which can adapt to the low memory and speed requirements of mobile or embedded applications. By sacrificing some performance to reduce model size, it has been widely used.

CNNs may also be combined with other technologies such as recurrent neural networks (RNN) and long short-term memory networks (LSTM) to form more powerful hybrid models [26-27]. A composite scaling method is proposed in which composite coefficients and AutoML are used to scale CNN evenly from multiple dimensions, by uniformly scaling the depth, width, and resolution of the network and the accuracy and efficiency of the model can be greatly improved even with limited computing resources [28]. In [29], the researchers proposed a domain specific EfficientNetwork as a classification method on face spoofing detection against paper attack and video-replay attack. Transformer-based CNN is a model that combines CNN and transformer architecture that utilizes the advantages of CNN in local feature extraction and transformer in global information modeling. The advantage of transformer lies in using attention to capture global contextual information and establish long-range dependencies on the target, thereby extracting stronger features [30-32].

3 Proposed Network Model

The section introduces the proposed network model, ResNet-Inception, as shown in Fig. 3. The backbone network used is ResNet-18, into which 4 Inception blocks were integrated, followed by straightening, dropout, and full connection, and finally output. ResNet-Inception has 8 residual blocks, each with a structure as shown in Fig. 3.

3.1 Architecture of the Proposed Network Model (ResNet-Inception)

The number of layers in a CNN network is not the main factor determining its quality. CNN has three scaling dimensions, namely depth, width, and resolution. Depth only refers to the depth of a network, which is equivalent to the number of layers in the network. Width is the width of the network. One measure of width is the number of channels in the convolution layer, while resolution is simply the image resolution passed to the CNN. So, we construct the ResNet-Inception network from the perspective of the width and depth. Inception block achieves multi-scale feature extraction by using 1x1 convolution, 3x3 convolution, 5x5 convolution, and pooling operations, as shown in Fig. 5. Inception block addresses the issues of gradient vanishing and model degradation in which the use of residual connections of ResNet can preserve the original features, making the network's learning smoother and more stable, further improving the accuracy and generalization ability of the model. So, Residual blocks with 64, 128, 256 and 512 convolution kernels were exploited respectively as illustrated in Fig. 3. The ReLU activation function and Dropout operation were used to achieve nonlinear mapping and prevent overfitting, as shown in Fig. 4 and Fig. 5. In residual block, max pooling is used every two layers. By concatenating feature maps of different sizes together, a new feature map is formed, further enhancing the expressive power of the model. Below, we will briefly introduce the basic structure of deep CNNs, as well as the relevant theories and methods used in our network.

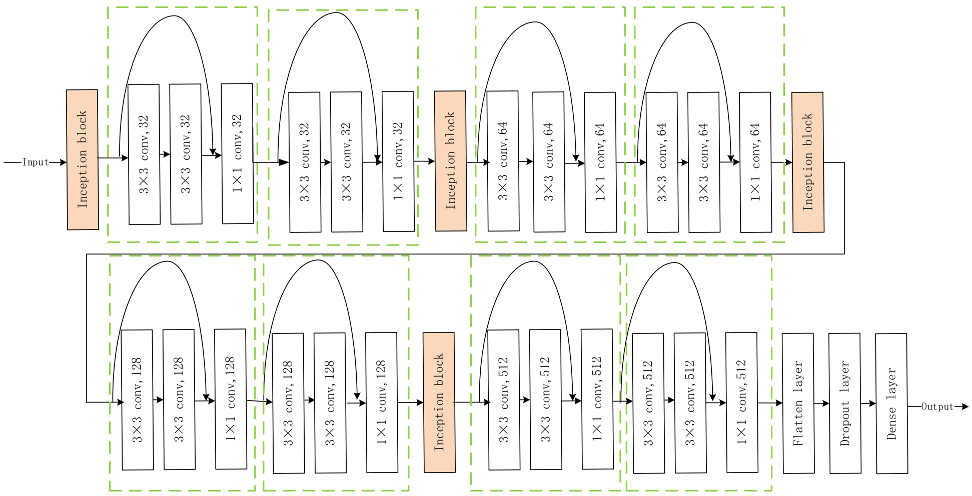
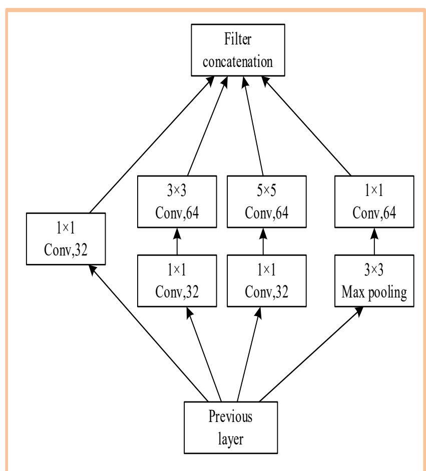


Fig. 3. The ResNet-Inception network



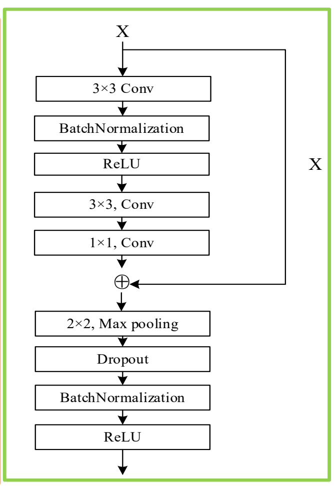


Fig. 4. The residual block **Fig. 5.** The inception block

3.2 CNNs Architectures

CNNs play a core role in image classification, as they can effectively process and classify image data through their specific layer structure and design. CNNs are mainly composed of three types of layers: convolutional layers, pooling layers, and fully connected layers as illustrated in Fig. 6. The combination of these layers makes CNN particularly suitable for processing image data, especially when dealing with location related data such as images, videos, audio, etc. Below is a detailed description of each layer structure of CNN.

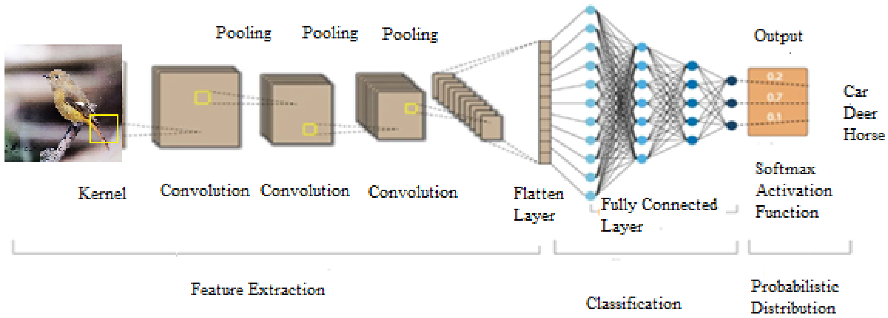


Fig. 6. Convolution neural network for image classification

Convolution: The convolution layer's operation is represented by formula (1). F represents image, g represents the kernel function and h represents the convolution result. The convolution calculation can be expressed by Fig. 7. One convolution is equivalent to shifting the convolution kernel on the input image. The convolution kernel starts from the top left of the input image, multiplies and adds the corresponding values, and calculates the numerical value at the corresponding position of the feature map through equation (1) for each step of shifting. Each value in a CNN feature map corresponds to a convolution operation in a certain region of the input image. The receptive field in CNN refers to the corresponding region of the input image or feature map that corresponds to a point on the feature map. Through multi-layer convolution, CNN can obtain features at different levels. According to the two-dimensional convolution calculation expression, the calculation of h (1,1) in the feature map is as follows:

$$h(i,j) = (f * g)(i,j) = \sum_k \sum_l g(k,l)\big(f(i+k, j+l)\big) \tag{1}$$

$$\underset{f}{\begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}} \times \underset{g}{\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}} = \underset{h}{\overset{h(1,1)}{\begin{bmatrix} 1 & 2 & 2 \\ 2 & 2 & 2 \\ 1 & 0 & 2 \end{bmatrix}}}$$

Fig. 7. Example of 2D convolution calculation

h (1,1) = g(0,0) *f(1,1) +g(0,1) * f(1,2) +g(1,0) *f(2,1) +g(1,1) * f(2,2) =1+1+0+0=2

After passing through the convolutional layer, the input feature map needs to be fed into the activation function for nonlinear transformation to obtain the activation output. Common activation functions include Sigmoid, Tanh, ReLU, etc. We use the ReLU function for activation. The equation is shown in (2).

$$F(x) = \begin{cases} 0, x \leq 0 \\ x, x > 0 \end{cases} \tag{2}$$

Among them, all negative values in the ReLU function are treated as 0, while all positive values remain unchanged, playing a role in one-sided suppression. This function structure allows the activation function to only activate some features instead of all features when responding to them, resulting in better sparse activation. This also ensures better accuracy of the reconfigurable CNN in feature extraction, extracting feature values that are closer to the target. It can be clearly seen that without adding an activation function to the entire CNN, the network structure only performs simple convolution operations, and the input and output data are only linear combinations after convolution operations are completed, and cannot approach the nonlinear functions that the CNN wants to achieve.

Pooling: Max pooling is sensitive to the presence of certain patterns in the pooling area. which is more popular than the mean-pooling operation because of its better performance [33]. We then apply a max pooling operation over the feature map derived from the convolution layer to produce a high-level representation feature. as shown in Fig. 8, the max pooling operation will slide the extraction window from the input feature map and output the maximum value of each channel to represent the features in this window. The sliding window size calculated is 2×2, and the feature map is down sampled to halve the size of the feature map output by the previous layer. Observing the max value of different features instead of the average value can provide more information about the features. Max pooling takes the max pixel value in the pooling area as the output, preserving the prominent feature information in the image. The feature map output after pooling with a stride of 2 is obtained, as shown in Fig. 8.

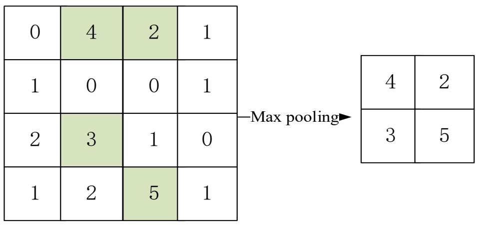


Fig. 8. Maximum pooling

Due to the fact that the operation on the original dataset only stays on the original images, there is no new dataset available for training and optimizing the generalization ability of the neural network. Because data augmentation can only reduce overfitting, it cannot eliminate the impact of overfitting. Therefore, on the basis of data augmentation, we also use the method of random inactivation to further reduce overfitting, that is, before connecting the feature map to the classifier, the method of random inactivation is adopted. Dropout [34] is a commonly used regularization method. Dropout method can be used for any layer in a reconfigurable CNN. Through different operations such as convolution and dimensionality reduction, the generated new data is passed on to the next layer for further processing.

The reason why CNNs produce overfitting is that the correlation between each data is too strong, which leads to insufficient ability of CNNs to generalize such data, resulting in insufficient final generalization ability. The core idea of dropout regularization method is to reduce the correlation between each data, so that related data will not be learned simultaneously during the learning process of CNN, and the correlation between two data will also be learned, thereby improving the recognition accuracy of road risk targets on a small data scale. Strong regularization such as maxout [35] or dropout [34] is applied to obtain the best results. After adopting Dropout regularization, the Dropout regularization method will randomly discard a portion of the output features of the CNN layer using Dropout regularization through a certain probability function, determine the discarded data, multiply the remaining data by weights, and then form a new dataset with reduced correlation, which will be passed on to the next connected network structure for other operations. Apply Dropout with a probability of 50% to each of the two hidden layers, and the flowchart of Dropout regularization method is shown in Fig. 9 and Fig. 10.

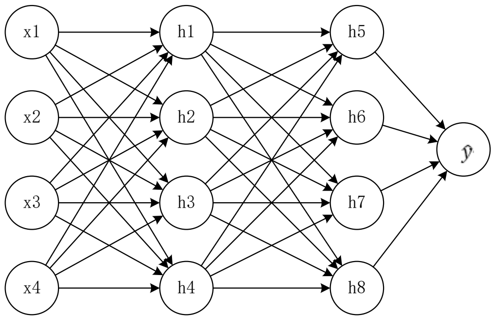


Fig. 9. CNN before adding dropout

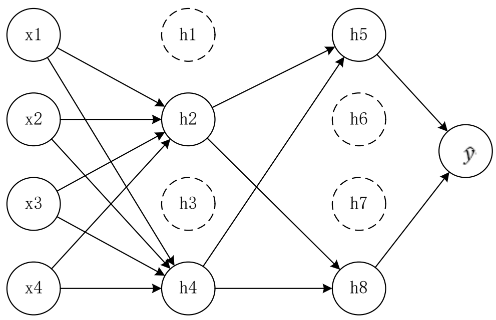


Fig. 10. The structure of the model after Dropout application

Full Connected and Output: Because the values obtained through convolutional layer operations cannot be directly connected to fully connected layers. So, it is necessary to perform dimensionality reduction on the output data. Reduce and compress multidimensional input data into one-dimensional data, and then re output it to the fully connected layer. Each unit in the fully connected layer will be densely matched and connected to all units in the previous layer. A fully connected Softmax layer is for classification. The fully connected layer is applied to score all possible labels according to the features in the pooling layer. The output can be computed with a Softmax function as in (3). $z^l \in R^K$ is the output of the last linear layer.

$$p = \left(y = i\right) = \frac{e^{z_i}}{\Sigma_{j=0}^{K-1} e^{z_j}}, i \in \left\{0, \ldots, K-1\right\} \tag{3}$$

Here, a combination of cross entropy and Softmax is used as the loss function to avoid numerical instability as shown in (4). It is implemented by the function, tf.nn.softmax_cross_entropy_with_logits in tensorflow. The smaller the cross entropy, the closer the actual and expected probability distributions are, and the smaller the loss. The input logits are first calculated using the sigmoid function, and then their cross entropy is calculated. However, the calculation method of cross entropy has been optimized to prevent overflow of the obtained results.

$$L = -\frac{1}{n}\sum_{i-1}^{n}\left[y_i \log_2^{\hat{y}_i} + \left(1 - y_i\right)\log_2\left(1 - \hat{y}_i\right)\right] \tag{4}$$

3.3 Date Augmentation

CNNs involve a large number of feature related parameters that need to be trained and learned during training. The most serious issue among them is overfitting. The meaning of overfitting is that CNNs perform very well on training data, but have poor accuracy when applied to test data, and cannot generalize unknown data with good ability. Among them, generalization ability refers to the performance of a trained model on unknown data. If the size and scale of the dataset are not limited, then a CNN can observe all the features of the training data without overfitting, indicating its good generalization ability. To achieve this goal, regularization methods will be used, which reduce overfitting to avoid this problem. In the above-mentioned issues, we use data augmentation [36] and random inactivation [37] to reduce overfitting. Data augmentation is a fundamental yet highly effective regularization method. Especially for CNNs. When the number of training samples is small, due to the influence of target actions, lighting, position, clothing, etc. during shooting, many captured image data cannot be directly used. Therefore, there are very few images available for training CNNs.

It will cause serious overfitting problems in the trained CNN, resulting in poor accuracy and generalization ability in the final recognition stage. In order to solve the above problems, we operate on the images in the original dataset based on spatial basis, and selects appropriate overfitting reduction methods on a small-scale CNN to improve the final recognition accuracy of the entire CNN. We will perform operations such as angle flipping, horizontal or vertical movement, random scaling of the image, horizontal flipping of parts of the entire image, and padding of new pixels after translation or rotation within a certain constraint range on a single RGB image. As illustrated in Table 1, we created an ImageDataGenerator instance and specified a series of parameters for data augmentation. The early stop callback is used to monitor the validation loss. If the loss rate of the validation set does not improve in five consecutive epochs, the training will be terminated prematurely. These methods will be used to exponentially expand the current limited sample data. The trained CNN will not have the opportunity to repeatedly see the same input data. But all input data is correlated, which means that the extracted features are correlated and can effectively reduce the impact of overfitting and we compared these in the experimental section.

Table 1. Data augmentation code with tensorflow

```
from tensorflow.keras.prepocessing.image import ImageDataGenerator
Generator = ImageDataGenerator(
  rotation_range = 20,
  width_shift_range = 0.15,
  height_shift_range = 0.15
  shear_range = 0.15
  zoom_range = 0.15
  horizontal_flip = True
  vertical_flip = True)
generator.fit(x_train)
early_stop = tf.keras.callbacks.EarlyStopping(monitor = 'val_loss',
  patience = 5)
history = model.fit(train_x, train_y, validation_split = 0.2, epochs=30,
callbacks = [early_stop] )
```

4 Experimental Results

We use a hardware platform with Intel i9-13950HX CPU, GetForce RTX 4090 GPU with 32GB memory. TensorFlow is as a neural network framework. Cifar-10 dataset is used. Cifar-10 is a color image dataset closer to universal objects. It was compiled by Hinton's students Alex Krizhevsky and llya Sutskever for identifying ubiquitous objects. There are a total of 10 categories of RGB color images: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck, as shown in Fig. 11. The size of each image in the Cifar-10 dataset is 32×32, with 6000 images in each category. There are a total of 50000 training images and 10000 test images in the dataset.

Fig. 11. Samples for Cifar-10 dataset

After downloading and decompressing the files from the official website, they mainly include batches.meta, data_batch_1, data_batch_2, data_batch_3, batch_batch_4, data_batch_5, readme.html, test_batch, etc. The file can be read through deserialization, and each batch file contains a python dictionary structure in which B 'data' is a 10000×3072 array, with elements in each row forming a 3-channel image of 32×32×3=3072, totaling 10000 images, B 'labels' is a list with a length of 10000, corresponding to the label of each image in the data, B 'batch label' is the name of this batch, and B 'filenames': is a list with a length of 10000, corresponding to the name of each image in the data.

4.1 Performance Analysis

By adding Inception blocks to the residual network, the network width is increased. Due to the use of multi-scale convolutions such as 1×1 convolution, 3×3 convolution, and 5×5 convolution in the Inception blocks, more features are extracted, and the residual network retains more previous features. We designed this CNN model to verify the combination of two network structures, not just the superposition of structures, but also the enhancement of network performance, providing a basis for adjusting the model structure in the future.

We combine CNNs with data enhancement and regularization methods with reduced overfitting. CNNs are trained separately. Using the preprocessed dataset as input, compare the impact of data augmentation and regularization methods on the recognition accuracy and loss of CNNs in different datasets. Because the number of model iterations increases, the training time of the model becomes longer, so we have added breakpoint continuation training.

Train Inception-v1 [24], ResNet-18 [13] and ResNet-Inception on the same dataset, as well as ResNet-Inception with feature extraction, data augmentation, and adjusted reuse. Compare the performance and loss of these three models in recognition under Cifar-10 dataset. After 30 epochs iterations, we printed the accuracy and loss curves of Inception-v1, ResNet-18 and ResNet-Inception A and ResNet-Inception B as shown in Fig. 13. The training and validation accuracy rates of each epoch are illustrated in Table 2. Fig. 12 shows top-1 error (20-crop testing) on Cifar-10 validation in which the left curves denote training error rates, and the right curves denote validation error ones. Though our model initially had high error rates during training and testing, as the epoch period increased, the error rates of our model gradually decreased, showing a relatively stable downward trend, similar phenomena is presented in Fig. 13. The top-1 error rate values at the end of the model are shown in Table 3.

Table 2. Accurate rates on Cifar-10 with Inception-v1, ResNet-18, ResNet-Inception A and ResNet-Inception B of training and validation (ResNet-Inception A does not perform data augmentation and ResNet-Inception B does.)

Model / Epochs	Inception -v1		ResNet-18		ResNet-Inception A		ResNet-Inception B	
	Training	Validation	Training	Validation	Training	Validation	Training	Validation
1	0.3201	0.4606	0.2520	0.1853	0.2045	0.1936	0.2463	0.2358
2	0.5059	0.5684	0.4392	0.2169	0.3624	0.3006	0.4023	0.2657
3	0.5994	0.6043	0.5247	0.4838	0.5018	0.3721	0.5349	0.4060
4	0.6529	0.6588	0.5781	0.4102	0.5809	0.5128	0.6324	0.5814
5	0.6930	0.6649	0.5956	0.5031	0.6434	0.3958	0.6903	0.5399
6	0.7235	0.6821	0.6511	0.6104	0.6851	0.6385	0.7332	0.6774
7	0.7455	0.7185	0.6872	0.5766	0.7160	0.5428	0.7640	0.6530
8	0.7686	0.7073	0.7071	0.6559	0.7461	0.6438	0.7878	0.7318
9	0.7847	0.7077	0.7256	0.6732	0.7685	0.6822	0.8038	0.7348
10	0.8004	0.7336	0.7447	0.6952	0.7828	0.6635	0.8167	0.7356
11	0.8146	0.7431	0.7568	0.7017	0.8009	0.6461	0.8312	0.7603
12	0.8260	0.7484	0.7695	0.6296	0.8096	0.7604	0.8423	0.7792
13	0.8372	0.7414	0.7832	0.7569	0.8245	0.6904	0.8509	0.7791
14	0.8478	0.7564	0.7942	0.7013	0.8359	0.6721	0.8598	0.8016
15	0.8584	0.7408	0.8025	0.7507	0.8459	0.7721	0.8688	0.7952
16	0.8638	0.7501	0.8140	0.7559	0.8554	0.7967	0.8737	0.7669
17	0.8719	0.7603	0.8215	0.7864	0.8629	0.7939,	0.8826	0.6858
18	0.8741	0.7515	0.8304	0.7011	0.8680	0.8268	0.8874	0.8193
19	0.8821	0.7554	0.8358	0.7523	0.8762	0.7950	0.8930	0.8332

20	0.8883	0.7636	0.8412	0.7592	0.8816	0.8046	0.8981	0.8276
21	0.8920	0.7537	0.8474	0.7719	0.8861	0.8195	0.9018	0.7704
22	0.8964	0.7601	0.8521	0.8001	0.8874	0.8145	0.9064	0.8421
23	0.9001	0.7501	0.8588	0.7732	0.8945	0.8078	0.9112	0.8337
24	0.9047	0.7694	0.8626	0.7734	0.8998	0.8173	0.9157	0.8380
25	0.9062	0.7651	0.8688	0.758	0.9046	0.8233	0.9159	0.8276
26	0.9113	0.7654	0.8722	0.7570	0.9122	0.8071	0.9204	0.8474
27	0.9162	0.7626	0.8748	0.7240	0.9092	0.8267	0.9239	0.8438
28	0.9155	0.7696	0.8803	0.7424	0.9174	0.8260	0.9262	0.8399
29	0.9208	0.7630	0.8725	0.6397	0.9208	0.8354	0.9284	0.8549
30	0.9212	0.7664	0.8637	0.7581	0.9223	0.8414	0.9319	0.8663

Table 3. Top-1 error rates with with Inception-v1, ResNet-18, ResNet-Inception A and ResNet-Inception B

Model	Top-1 error rates
Inception-v1	0.2336
ResNet-18	0.2419
ResNet-Inception A	0.1586
ResNet-Inception B	0.1337

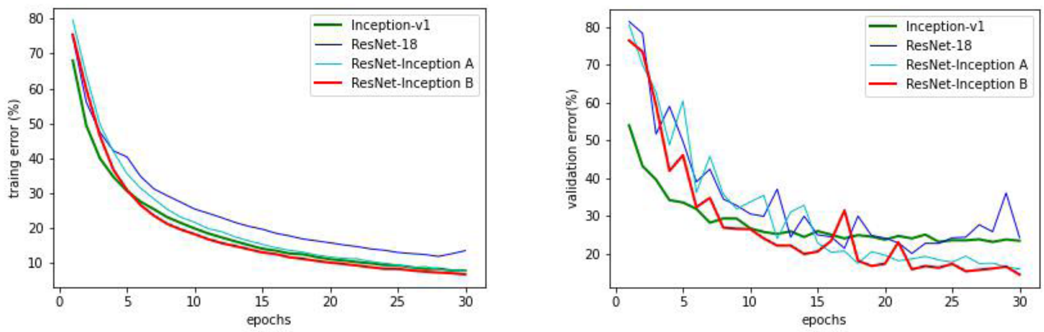


Fig. 12. Training error (left) and test error (right) on Cifar-10 with Inception-v1, ResNet-18, ResNet-Inception A and ResNet-Inception B (ResNet-Inception A does not perform data augmentation and ResNet-Inception B does.)

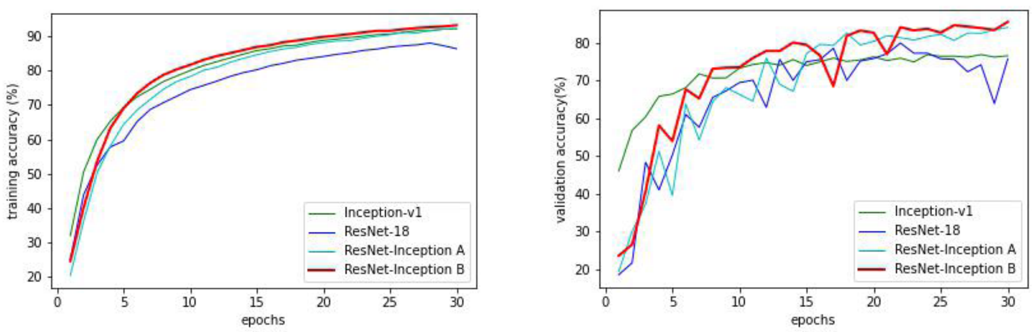


Fig. 13. Accuracy and loss for Inception-v1, ResNet-18, ResNet-Inception A and ResNet-Inceptioon B

Fig. 13 show the performance of Inception-v1, ResNet-18, ResNet-Inception A and ResNet-Inception B with data augmentation and regularization methods on the same training and testing sets. As the training and recognition process progresses, the performance of ResNet-Inception gradually approaches that of the two datasets, and the accuracy continues to improve. This indicates that the addition of data augmentation and regularization methods can effectively improve the recognition accuracy of shallow reconfigurable CNN models. Data augmentation effectively solves the problem of data scarcity by processing existing data, not only making the model perform well on training data, but also on testing data, improving the model's performance in different situations and making it have stronger generalization ability.

5 Conclusion and Future Study

In the paper, we present a novel reconfigurable CNN model, ResNet-Inception for small-scale data in which inception blocks are integrated into the residual network. The accuracy of the trained reconfigurable CNN model is verified, and the obtained results are analyzed and explained. Experiments have shown that our network model outperforms both the compared Inception-v1 and the ResNet-18. In the future, it will be used in object detection applications and other ones for the datasets built by ourselves. We will apply the proposed model to achieve road risk target detection. The problem of road risk target detection for complex roads requires building one's own datasets. In the case of a limited dataset, it is necessary to attempt to establish a more comprehensive and efficient CNN recognition model to achieve the target detection task.

Acknowledgement

This work was supported by the China Liaoning Province Science and Technology Plan Joint Program (Applied Basic Research Project) (Grant No. 2023JH2/101700316).

References

[1] R.B. Arantes, G. Vogiatzis, D.R. Faria, Learning an augmentation strategy for sparse datasets, Image and vision computing 117(2022) 1-12.
https://doi.org/10.1016/j.imavis.2021.104338

[2] P. Thanapol, K. Lavangnananda, P. Bouvry, F. Pinel, F. Leprévost, Reducing Overfitting and Improving Generalization in Training Convolutional Neural Network (CNN) under Limited Sample Sizes in Image Recognition, in: Proc. 2020 International Conference on Information Technology, 2020.
https://doi.org/10.1109/InCIT50588.2020.9310787

[3] E.D. Cubuk, B. Zoph, D. Mané, V. Vasudevan, Q.V. Le, AutoAugment: Learning Augmentation Strategies from Data, in: Proc. 2019 IEEE Conference on Computer Vision and Pattern Recognition, 2019.
https://doi.org/10.1109/CVPR.2019.00020

[4] C. Shorten, T.M. Khoshgoftaar, A survey on Image Data Augmentation for Deep Learning, Journal of Big Data 6(1) (2019) 1-48.
https://doi.org/10.1186/s40537-019-0197-0

[5] P. Dai, Y. Li, H. Zhang, J. li, X. Cao, Accurate Scene Text Detection via Scale-Aware Data Augmentation and Shape Similarity Constraint, IEEE Transactions on Multimedia 24(2022) 1883-1895.
https://doi.org/10.1109/TMM.2021.3073575

[6] M. Paschali, W. Simson, A.G. Roy, M.F. Naeem, R. Göbl, C. Wachinger, N. Navab, Data Augmentation with Manifold Exploring Geometric Transformations for Increased Performance and Robustness. <https://arxiv.org/abs/1901.04420>, 2019 (accessed 10.06.2024).

[7] L. Wan, M. Zeiler, S. Zhang, Y. LeCun, R. Fergus, Regularization of Neural Networks using DropConnect, in: Proc. the 30th International Conference on Machine Learning, 2013.

[8] D. Ciregan, U. Meier, J. Schmidhuber, Multi-column deep neural networks for image classification, in: Proc. 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012.
https://doi.org/10.1109/CVPR.2012.6248110

[9] K. Pei, Y. Cao, J. Yang, S. Jana, Towards Practical Verification of Machine Learning: The Case of Computer Vision Systems. <https://arxiv.org/abs/1712.01785>, 2017 (accessed 02.04.2024).

[10] K. Pei, Y. Cao, J. Yang, S. Jana, Deepxplore: automated whitebox testing of deep learning systems, in: Proc. the 26th Symposium on Operating Systems Principles, 2017.
https://doi.org/10.1145/3132747.3132785

[11] Y. Liu, S. Yu, T. Lin, Hessian regularization of deep neural networks: A novel approach based on stochastic estimators of Hessian trace, Neurocomputing 536(2023) 13-20.
https://doi.org/10.1016/j.neucom.2023.03.017

[12] W. Zhao, L. Dong, Machine Learning, second ed., Posts and Telecommunications Press, Beijing, China, 2022 (Chapter 10).

[13] K. He, X. Zhang. S. Ren, J. Sun, Deep Residual Learning for image recognition, in: Proc. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.
https://doi.org/10.1109/CVPR.2016.90.

[14] J.-X. Gu, Z.-H. Wang, J. Kuen, L.-Y. Ma, A. Shahroudy, B. Shuai, T. Liu, X.-X. Wang, G. Wang, J.-F. Cai, T. Chen, Recent advances in convolutional neural networks, Pattern Recognition 77(2018) 354-377.
https://doi.org/10.1016/j.patcog.2017.10.013

[15] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE 86(11)(1998) 2278-2324.
https://doi.org/10.1109/5.726791

[16] S. Cong, Y. Zhou, A review of convolutional neural network architectures and their optimizations, Artificial Intelligence Review 56(2023) 1905-1969.
https://doi.org/10.1007/s10462-022-10213-5

[17] S. Srinivas, S.K. Sarvadevabhatla, K.R. Mopuri, N. Prabhu, S.S.S. Kruthiventi, R.V. Babu, A taxonomy of deep convolutional neural nets for computer vision, Frontiers Robotics and AI 2(2016) 36.
https://doi.org/10.3389/frobt.2015.00036

[18] L. Alzubaidi, J. Zhang, A.J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaria, M.A. Fadhel, M. Al-Amidie, L. Farhan, Review of deep learning: concepts, CNN architectures, challenges, applications, future directions, Journal of Big Data 8(1)(2021) 53.
https://doi.org/10.1186/s40537-021-00444-8

[19] A. Khan, A. Sohail, U. Zahoora, A.S. Qureshi, A survey of the recent architectures of deep convolutional neural networks, Artificial intelligence Review 53(8)(2020) 5455-5516.
https://doi.org/10.1007/s10462-020-09825-6

[20] Q. Zhang, M. Zhang, T. Chen, Z. Sun, Y. Ma, B. Yu, Recent advances in convolutional neural network acceleration, Neurocomputing 323(2019) 37-51.
https://doi.org/10.1016/j.neucom.2018.09.038

[21] T. Choudhary, V. Mishra, A. Goswami, J. Sarangapani, A comprehensive survey on model compression and acceleration, Artificial Intelligence Review 53(7)(2020) 5113-5155.
https://doi.org/10.1007/s10462-020-09816-7

[22] A. Krizhevsky, L. Sutskever, G.E. Hinton, ImageNet Classification with Deep Convolutional neural networks, Communications of the ACM 60(6)(2017) 84-90.
https://doi.org/10.1145/3065386

[23] K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition. <https://arxiv.org/abs/1409.1556>, 2014 (accessed 11.05.2024).

[24] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going Deeper with Convolutions, in: Proc. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.
https://doi.org/10.1109/CVPR.2015.7298594

[25] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. <https://arxiv.org/abs/1704.04861>, 2017 (accessed 04.02.2024).

[26] C. Olivieri, F. De Paulis, A. Orlandi. C. Pisani, S. Tessitore, G. Giannuzzi, Application of an integrated RNN-ensemble method for the short-term forecast of inter-area oscillations modal parameters, Electric Power Systems Research 225(2023) 1-10.
https://doi.org/10.1016/j.epsr.2023.109790

[27] A. Saxena, H.-M. Sun, TokenDoc: Source Authentication with a Hybrid Approach of Smart Contract and RNN-Based Models with AES Encryption, IEEE Transactions on Engineering Management 71(2023) 12418-12432.
https://doi.org/10.1109/TEM.2023.3282293

[28] M. Tan, Q.V. Le, EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. <https://arxiv.org/abs/1905.11946v5>, 2019 (accessed 01.03.2024).

[29] C. Yao, S. Wang, J. Zhang, W. He, H. Du, J. Ren, R. Bai, J. Liu, rPPG-Based Spoofing Detection for Face Mask Attack using Efficientnet on Weighted Spatial-Temporal Representation, in: Proc. 2021 IEEE International Conference on Image Processing (ICIP), 2021.
https://doi.org/10.1109/ICIP42928.2021.9506276

[30] N. Zhang, F. Nex, G. Vosselman, N. Kerle, Lite-Mono: A Lightweight CNN and Transformer Architecture for Self-Supervised Monocular Depth Estimation, in: Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023.
https://doi.org/10.1109/CVPR52729.2023.01778
[31] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby, An image is worth 16x16 words: Transformers for image recognition at scale, in: Proc. International Conference on Learning Representations (ICLR 2021), 2021.
[32] J. Guo, K. Han, H. Wu, Y. Tang, X. Chen, Y. Wang, C. Xu, CMT: Convolutional Neural Networks Meet Vision Transformers, Computer Science, in: Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022.
https://doi.org/10.1109/CVPR52688.2022.01186
[33] Z. Cao, W. Li, S. Li, F. Wei, Y. Li, Attsum: Joint learning of focusing and summarization with neural attention, in: Proc. COLING 2016, the 26th International Conference on Computational Linguistics, 2016.
https://aclanthology.org/C16-1053/
[34] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R.R. Salakhutdinov, Improving Neural Networks by Preventing Co-adaptation of Feature Detectors. <https://arxiv.org/abs/1207.0580>, 2012 (accessed 02.04.2024).
[35] I.J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, Y. Bengio, Maxout Networks, in: Proc. ICML'13: the 30th International Conference on International Conference on Machine Learning, 2013.
[36] C.L. Zhang, J.H. Luo, X.S. Wei, J. Wu, In Defense of Fully Connected Layers in Visual Representation Transfer, in: B. Zeng, Q. Huang, A. El Saddik, H. Li, S. Jiang, X. Fan (Eds.), Advances in Multimedia Information Processing – PCM 2017. PCM 2017. Lecture Notes in Computer Science, vol 10736. Springer, Cham, 2018.
https://doi.org/10.1007/978-3-319-77383-4_79
[37] S. Zhang, Y. Gong, J. Wang, The Development of Deep Convolutional Neural Networks and Their Applications in the Field of Computer Vision, Chinese Journal of Computers 42(3)(2019) 453-482.
http://cjc.ict.ac.cn/online/onlinepaper/zs-2019319124816.pdf