Proceedings of the 3rd International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/4/2023365

Discussion on image recognition under different conditions of ResNet and DenseNet

Guangrui Li

Southwest Jiaotong University, Chengdu, Sichuan province, 614202, China

sc19gl@leeds.ac.uk

Abstract. As two mainstream and excellent neural networks, ResNet and DenseNet have been the main research directions of many scholars and papers. This paper mainly focuses on these two neural networks through a theoretical analysis, summary discussions, and practical experiments. The author explains their previous advantages and disadvantages, and, as a potentially obsolete technology, whether ResNet will be better or more efficient than DenseNet in some cases. The final experimental results show that although ReseNet is a somewhat outdated neural network model, it still outperforms DenseNet in some aspects. In addition, this paper provides some further optimization schemes for reference through the analysis of the two neural network models. At the same time, the basic structure and algorithm principle of the two neural networks are further explained in the analysis to make it clearer for readers.

Keywords: DenseNet, ResNet, Algorithm Principle, Efficiency Optimization.

1. Introduction

In the neural network, better feature expressions can be obtained through the accumulation of layers. With the increase of the layers of a network, the features of different layers that can be extracted will also increase, and more semantic information can be obtained if the features extracted by the network are abstract. When using CNN algorithm to learn, the features are intuitive [1]. What's more, some portion of the features extracted especially in early layers can demonstrate a certain degree of opacity [1]. Therefore, better experimental results and performance can be obtained by simply increasing the number of the neural network layers. However, excessively increasing the number of neural network layers will not bring better results. The number of neural network layers in deep learning is not the more the better. This is mainly for two reasons, Gradient Vanishing or Explosion and the performance degradation of the network model. What is more, CNN has more shortcomings. For example, its receptive fields are limited, it cannot adapt to input content, and its computational complexity grows quadratically with the spatial resolution [2]. This fact shows that pure stacking layers cannot improve the efficiency and correct rate of the network. So, DenseNet and ResNet are established as two models, and the deeply-supervised nets (DSN) method is proposed to minimize the classification error and at the same time, make the learning process of hidden layers direct and transparent [3].

The main purpose of this article is to discuss the differences between the DenseNet and ResNet neural network models under different conditions. In order to evaluate these two models in an intuitive and reasonable way, image recognition is applied to judge these two neural network models by the correct

DOI: 10.54254/2755-2721/4/2023365

rate of image recognition and time consumption. In this experiment, two DenseNet and ResNet models with the same number of layers are trained with the same data set. The parameters changed in this experiment are the number of neural network layers and the size of the dataset. Through repeated experiments, tests, and comparisons, the author aims to explore the differences between the two neural network models in various situations.

2. DenseNet and ResNet Models

2.1. Design of DenseNet and ResNet

The cascade structure, which is similar to the DenseNet structure design, can be traced back to the network literature in the 1980s [4]. The DenseNet proposes an innovative dense connection method. All layers are connected with each other, and every layer accepts all the layers before it as an additional input. In the DenseNet, every single layer is concatenated with all other layers before it in the channel dimension and its output will also be added in the input of the next layer. This means that, for a network of X layers, the DenseNet contains a huge connection of total $\frac{X(X+1)}{2}$. In comparison, a traditional convolutional neural network has L connections with the L layers, which is a one-to-one model. As a dense connection, the DenseNet directly splices the feature maps of different layers, which can realize feature reuse and improve efficiency. From the perspective of dimension, in the DenseNet, the connection between different layers becomes a dimensional superposition.

The DenseNet requires fewer parameters than traditional convolutional networks, so dense connections bring feature reuse without re-learning redundant feature maps, the operation of dimension splicing extracts rich feature information, and the less use of the convolution can obtain a lot of feature maps.

The traditional forward propagation network structure can be regarded as the transmission of state in different layers. Each layer reads the state information and writes it to the next layer. During this process, the state will change but at the same time, those that need to be the saved information will continue to be passed on. In the DenseNet, there is a clear distinction between the information that added or retained to the network. The layers of DenseNet are very narrow. Only a small set of feature maps to the network's collective knowledge are added, and the rest of them are kept unchanging. Ultimately, the classifier makes decisions according to all the feature maps in the network.

In addition to more efficient parameters, DenseNet also has another big advantage. It improves a lot of the information flow and the gradients running inside the network. The densely connected structure enables each layer to obtain gradients directly from the function and the original input, which is very beneficial for training deeper networks.

Compared with the DenseNet, the ResNet applies the residual learning theory to eliminate the degradation problem of the deep network caused by too many layers. Assuming that the input is X for a stacked layer structure, then the features learned are set as $H(X)$. It is hoped that it can learn with the residual $F(X) = H(X) - X$, which means the original learning feature will actually be $F(X) + X$ [5]. This is because that residual learning is easier than normal leaning when learning from raw features. This can be shown in a intuitive way if the residual is considered as 0 in an extreme way. In this situation the stacking layer only run the identity mapping, and at least there will not be performance degradation. In fact, the residual cannot be 0 and will result in better performance because the stacking layer will also learn new features on a basis of the input features. This can be compared as a "short circuit" in a circuit. Therefore, it is a short-way connection. Compared with the DenseNet, residual learning is relatively easier to realize. In other words, residual learning requires less learning time and operation process because residuals are generally smaller and easier to learn. ResNets pass the data from one layer to another through identity connections. By randomly dropping layers during training to achieve better information flow, ResNet is shortened by the stochastic depth [6]. And ResNet is composed of many such ‘Residual Units’. Each unit can be expressed in the expression below [7]:

$$y_1 = h(x_l) + f(x_l + W_l) \quad (1)$$

DOI: 10.54254/2755-2721/4/2023365

$$x_{l+1} = f(y_l) \tag{2}$$

2.2. Arithmetic Expressions of DenseNet and ResNet

In order to explain the two models more intuitively and clearly, the mathematical expressions are used here to illustrate. Although for most of the researchers, the structure of different models is still a black box and lacks a completely mathematical understanding and expression about the algorithmic behavior, several attempts are still made in terms of theoretical aspect [8]. Assume that the input image is X_0, the number of neural network layers is L, l is the layer index, $Hl(.)$ represents the combination of nonlinear operations, including BN, ReLu, 3x3 convolution, and the output of the lth layer is X_l.

In the ResNet, because the shortcuts map is added, the output of the l^{th} layer can be expressed by formula (3) [9].

$$X_l = H_l(X_l - 1) + X_{l-1} \tag{3}$$

In the DenseNet, the current layer and all subsequent layers are densely connected. At this time, the output of the l^{th} layer can be expressed by formula (4) [9].

$$X_l = H_l([X_0, X_1, \dots, X_{l-1}]) \tag{4}$$

2.3. Network Structures of DenseNet and ResNet

The CNN network generally needs to go through Pooling to minimize the size of the features, while the dense connection structure of DenseNet needs requires consistency in the size of the feature map. To solve the dilemma, the DenseNet network uses new structure named DenseBlock and Transition, where DenseBlock is a module consisting of multiple layers, the feature maps of each layer maintain the same size, and dense connections are applied among layers. Another module named Transition is established for connecting adjacent DenseBlocks. The feature maps of each layer In DenseBlock keep the same size and connect in the channel dimension. A structure consists of three parts, a BN, a ReLU function, and a 3x3 Conv, are adopted by the nonlinear combination function in DenseBlock. There is also a need to mention that, different from ResNet, all layers in DenseBlock output [formula] feature maps after convolution. In other words, for the obtained feature map, K is the number of channels, or K convolutions are used nuclear. As a hyperparameter, K is called the growth rate in DensNet. To achieve better performance, a smaller K is generally used. Assuming that there are K0 channels in the feature map of the input layer, then the number of the channel delivered into the L layer is $K + K(L - 1)$. That is to say, as the number of the layers increases, although the parameter K is set as a small number, the input of DenseBlock will still be large. This is because of the reuse of the features which is unnecessary, and it increases the burden with no mean. Since the input will consistently increase when it goes deeper, the latter layer's input will be quite large, and there is a need to use the bottleneck layer inside DenseBlock, so as to reduce the amount of calculation, mainly by adding a 1x1 convolutional layer and the functions match it to the structure. The structure will be 2 BN, 2 ReLU function, a 1x1 Conv, and a 3x3 Conv, which is called DenseNet-B structure. A 1x1 Conv is used to reduce the number of features, aiming at raising computational efficiency. The Transition layer is used to connect the adjacent DenseBlocks. It mainly includes a 1x1 convolution and a 2x2 AvgPooling, so the structure is a BN, a ReLU function, a 1x1 Conv and a 2x2 AvgPooling. Besides, the transition layer can play the role of compressing the model. Assuming that there are Y feature map channels obtained through the connection of DenseBlock and the Transition, through the convolution layer, the Transition layer can generate αY features, with the compression rate of $\alpha \in (0,1]$. When α=1, the number of features going through the transition layer remains unchanging, which means a situation that there is no compression, and when α<1, this structure is called DenseNet-C. Moreover, for DenseBlock structure, DenseNet-BC refers to the use of the bottleneck layer and the transition combination structure with a compression coefficient less than 1.

For ResNet, two residual modules are often used. One is a two-layer residual module named Basic Block, and another one is a three-layer residual module named Bottleneck Block.

DOI: 10.54254/2755-2721/4/2023365

The two-layer residual module (Basic Block) mainly consists of two 3x3 convolutional layers and a short connection.

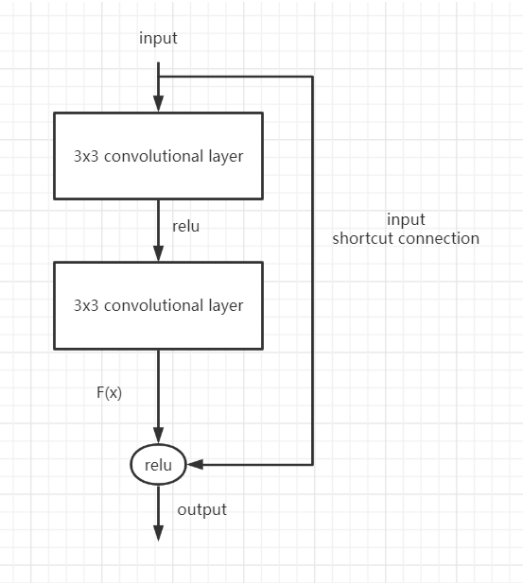

Figure 1. Traditional convolutional layer.

The part of the traditional convolutional layer (with two 3X3 layers and a ReLU activation function) can be expressed as $F(x) = W_1\sigma(W_2x)$. W_1, W_2 means the parameter of 2 convolutional layers, σ represents the ReLU activation function between two convolutional layers. As both two paths are with active ReLU function, the hole two-layer residual module can be presented in the formula $y = \sigma(F(x) + x) = \sigma(W_1\sigma(W_2x) + x)$.

However, there is a problem with the above two-layer residual module. There are too many parameters, and it takes too much time to train. For example, assume that there is a double-layer residual module with 64 channels of input and output, the parameter number is $\text{number} = (3 \times 3 \times 64 \times 64) \times 2 = 73728$. Too many parameters cause the operation speed to be greatly slowed down and complicated. So, the above two-layer residual module can be transformed into the following three-layer residual module (Bottleneck Block).

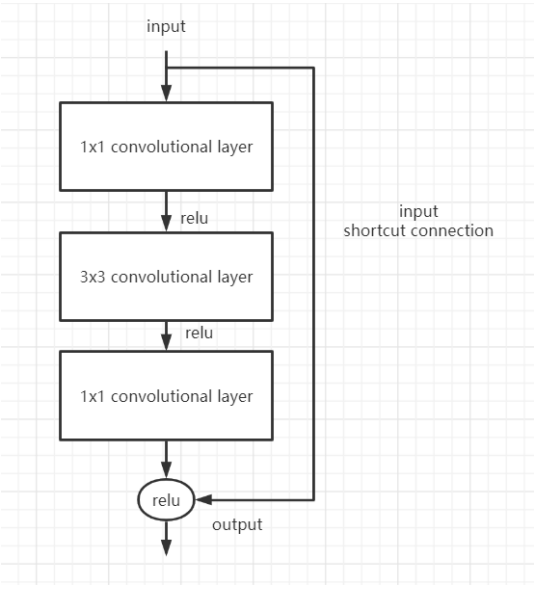

Figure 2. Split convolutional layer.

DOI: 10.54254/2755-2721/4/2023365

Assume that all the convolutional layers are 64 channels, then as a result, the parameter number will be $\text{number} = 1 \times 1 \times 64 \times 64 \times 2 + 3 \times 3 \times 64 \times 64 = 45056$. The number of parameters has been reduced from 73,728 to 45,056, which is only about 60% of the original (in fact, in some positions, the number of parameters of the three-layer residual module will become fewer, even only 5% of the same two-layer residual module). Experiments show that the three-layer residual module can achieve almost the same effect as the two-layer residual module.

Each layer of the three-layer residual module has its own division of labor. The 1x1 convolutional layer of the first layer aims at reducing the number of channels of the image; the 3x3 convolutional layer of the second layer is mainly for the reduction of the size of the image; the 1x1 convolutional layer of the third layer is mainly for the increase of the number of image channels. Roughly speaking, the ResNet network is mainly composed of the above-mentioned two-layer residual module (Basic Block) or three-layer residual module (Bottleneck Block) continuously superimposed. G. Huang et al. gave the specific architectures of 5 commonly used ResNet networks in their study. These 5 ResNet networks are of 18 layers, 34 layers, 50 layers, 101 layers, and 152 layers, respectively [9]. Among them, 18-layer and 34-layer are implemented with a double-layer residual block (Basic Block), while the networks with more than 50-layer are implemented with a three-layer residual block (Bottleneck Block). In fact, the 50-layer ResNet network is formed by replacing all the two-layer residual modules (Basic Block) in the 34-layer ResNet network with three-layer residual modules (Bottleneck Block).

Table 1. DenseNet model used in ImageNet Dataset [9].

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv; 3 × 3 conv] × 24	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 48	[1 × 1 conv; 3 × 3 conv] × 64
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv; 3 × 3 conv] × 16	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 48
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

3. Efficiency Comparison Results of Model Training

In this experiment, a dataset of about 800mb containing 1400 images was used for training. All the images have been edited into a proper size (32x32) and sorted into different categories. As the training proceeded, different layers of the two models have been used, different sizes of features are collected and the running time was recorded.

Table 2. Experiment Result.

Model	Params	Depth	Dataset	Correct
ResNet	1.8MB	34	500MB	61.5
ResNet	10.2MB	101	833MB	64.6
DenseNet	1.5MB	34	500MB	68.4
DenseNet	5.3MB	101	833MB	73.1

DOI: 10.54254/2755-2721/4/2023365

4. Conclusion

From the results, it can be seen that the DenseNet, as a new dense convolutional neural network, has surpassed the ResNet in terms of accuracy when the parameters are equal or smaller than the ResNet. In the paper, it is also mentioned that the DenseNet will not suffer from overfitting and the difficulty in optimizing residual networks. In other words, DenseNet has better latent capacity because it has the chance to use bigger and deeper models [9]. On the whole, the advantages of DenseNet are mainly reflected in the following aspects. DenseNet is easier to be trained because it improves the backpropagation of gradients due to dense connections. Since the final error signal can be obtained directly by each layer, implicit deep supervision is realized. Although the parameters become smaller, the calculation is still efficient since the DenseNet realizes the future reuse and a short-circuit connection through concat features. It also uses a smaller growth rate, and each layer has a relatively small feature map.

However, compared to ResNet, DenseNet still has some small problems, such as channel stacking due to the fully connected feature, which requires frequent memory reading. Improper implementation will lead to a slower computing speed and may consume a lot of GPU memory. Additional optimization like deploying the features received from the former layer into shared memory [10] is required when the amount of computing is large. So, when processing larger datasets without performing too many optimization operations, ResNet is better than DenseNet in terms of running speed. In addition, due to the special network structure of DenseNet, ResNet is more widely applicable.

References

[1] Zeiler M D and Fergus R 2013 *Stochastic pooling for regularization of deep convolutional neural networks,* In ICLR.

[2] Zamir S, Arora A, Khan S, Hayat M, Khan F and Yang M 2022 *Restormer: Efficient Transformer for High-Resolution Image Restoration.* In arXiv:2111.09881v2.

[3] Lee C Y, Xie S N, Gallagher P, Zhang Z Y and Tu Z W 2014 *Deeply-Supervised Nets*. In arXiv:1409.5185v2.

[4] Fahlman S E and Lebiere C 1989 *The cascade-correlation learning architecture*. In NIPS

[5] He K M, Zhang X Y, Ren S Q and Sun J 2015 *Deep Residual Learning for Image Recognition.* In arXiv:1512.03385v1.

[6] Huang G, Sun Y, Liu Z, Sedra D and Weinberger K Q 2016 *Deep networks with stochastic depth*. In ECCV.

[7] He K M, Zhang X Y, Ren S Q and Sun J 2016 *Identity Mapping in Deep Residual Networks.* In arXiv:1603.05027v3.

[8] Eigen D, Rolfe J, Fergus R and LeCun Y 2014 *Understanding deep architecture using a recursive convolutional network.* In arXiv:1312.1847v2.

[9] Huang G, Liu Z, Maaten L, Weinberger K Q 2018 *Densely Connected Convolutional Networks.* In arXiv:1608.06993.

[10] Pleiss G, Chen D, Huang G, Li T, Matten L and Weinberger K 2017 *Memory-Efficient Implementation of DenseNets*. In arXiv:1707.06990v1.