ZYZNet:Balancing Efficiency and Accuracy to Solve HCCR Task

ZuYuan ZHANG
Taishan College
Shandong University
1556823221@qq.com

Xiaojing ZHANG
Taishan College
Shandong University
1668231152@qq.com

Abstract

In this paper, we focus on a special case of Handwritten Chinese Character Recognition (HCCR), in which there is only one character in one image. To solve this problem, our group created a new CNN(Convolutional Neural Network) model inspired by VGGNet[1]. We tried to reduce resource consumption and improve efficiency, and obtain high accuracy, so ZYZNet was designed. ZYZNet has achieved better results after balancing efficiency and accuracy. To make our model more robust, we performed data augmentation. As a result, our final model could be robust to some transformations of the image, such as rotation and flipping.

1 Introduction

1.1 Handwritten Chinese Character Recognition Task

Handwritten Chinese Character Recognition is a task to recognize Chinese characters in an image or video. It is considered as an extremely difficult problem due to the wide variety, complicated structures, similarity between characters, and the variability of fonts or writing styles. Two decades ago, people usually used the structural matching method to obtain reliable stroke correspondence and enable structural interpretation to tackle this task.[2] But after the emergence of Neural Network, people now can use Deep Learning especially CNN(Convolutional Neural Network) to get better performance.

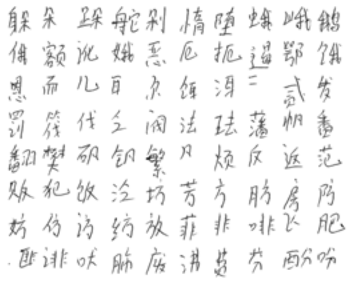


Figure 1: images for HCCR

1.2 Convolutional Neural Network

Convolutional Neural Network(CNN) is one of the Deep Learning algorithms. It is widely used in image and video recognition, recommendation system, image classification because it can capture the characteristics of images. CNN usually has three parts, an input layer, an output layer, and many hidden layers which usually consist of convolutional layers, pooling layers, and fully connected layers.

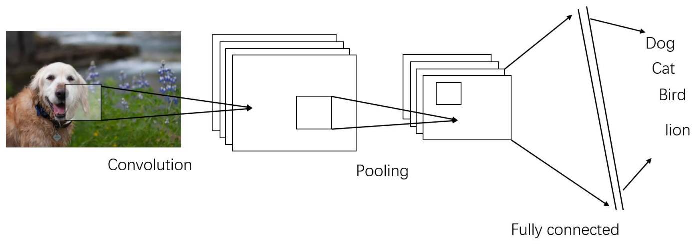


Figure 2: classic CNN structure including convolution layer, pooling layer and fully connected layer.

1.2.1 Convolutional Layer

Each convolutional layer has one or more small learnable filters. The size of these filters is usually 3×3 or 7×7. These filters move on the input matrix in some way. In each step, the filter covers some parts of the input matrix and computes the dot product between the filter and the covered matrix to form an output matrix.

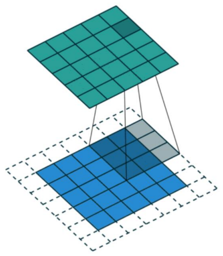
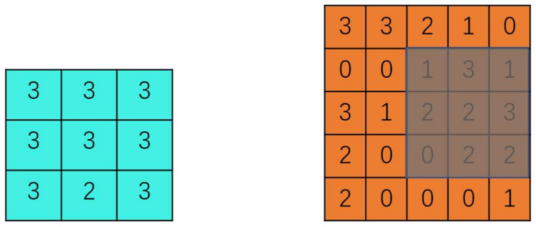


Figure 3: 3×3 filter with blue input matrix and green output matrix

1.2.2 Pooling Layer

Just like the convolutional layer, each pooling layer also has its filters and obeys the same way of moving on the input matrix. But the filters are not learnable, and in each step, for different types of pooling layers, the operations performed on the covered area are different. For the max-pooling layer, it will find the maximum number as the output in one step while for the average-pooling layer, the average of the numbers will be the output.



3	3	3
3	3	3
3	2	3

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

Figure 4: max-pooling layer with 3×3 filter, orange input matrix and light blue output matrix

1.2.3 Fully Connected Layer

The fully connected layer is a layer that each neuron is connected to every neuron in the previous layer. This layer is usually placed just in front of the output layer.

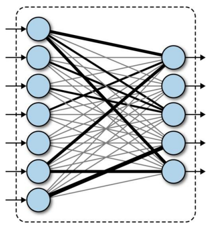


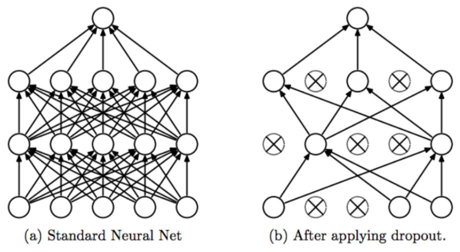
Figure 5: Fully connected layer

1.3 Activation Function

The activation function is usually used to bring some nonlinear factors to the neural network to tackle more difficult tasks. There are some classic activation functions such as ReLU[3], arctan, and Sigmoid.

1.4 Dropout Layer

During training, the dropout layer randomly discards a portion of the neurons to avoid overfitting and to achieve better generalization[4].



To tackle this task, our group first tried some classic models. Because the structure of VGGNet is simple, we chose VGGNet as our model basis. As there were fewer characteristics than the previous works of the classic models, we first changed the image size from 224×224 to 64×64 and then decreased the number of fully connected layers to only 1. To make our model more robust, we used some data augmentation methods like flipping and scaling and rotating the image at a random angle. To increase the generalization of the model, we finally added a dropout layer.

1.5 Division of Labor

Project Start Stage: Each person proposed a model and selected one person's model according to the effect. Zhang Zuyuan's model was selected.

Table 1: specific arrangement

	Zhang Zuyuan	Yu Yixiao	Zhang Xiaojing
code	Baseline training	Data processing Baseline training Robust model training	Baseline training Robust model training
paper	Rationale for building the network model Baseline Model architecture Experimental Results Acknowledgement Conclusions Images	Abstract Introduction Model improvements Experimental data	Data processing Background Related work Review
ppt	PPT framework Integration	Modify PPT details	Writing and modification

2 Background and Related Work

2.1 the Development of CNN Model

2.1.1 LeNet

LeNet[5], one of the earliest convolutional neural networks, was born in 1994. It uses convolution, parameter sharing, pooling and other operations to extract features, avoiding a lot of calculation costs, and finally uses a fully connected neural network for classification and recognition. The LeNet structure consists of two sets of convolutional layers and average-pooling layers, followed by a attening convolutional layer, then two fully connected layers and a softmax classier.

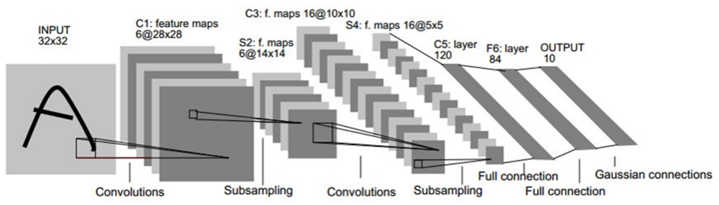


Figure 6: How Lenet works[5]

2.1.2 AlexNet

Due to the limitations of computers at that time, although LeNet achieved good results in image classification, it did not attract much attention. It was not until 2012 when the AlexNet[6] network proposed by Alex won the ImageNet competition, that convolutional neural networks and deep learning have regained widespread attention.

AlexNet was the largest convolutional neural network by then. Before this, the LeNet5 network had only 3 convolutional layers and 1 fully connected layer. Compared with LeNet, Alexnet has a deeper structure, and it adds a Relu function behind each convolutional layer, which solves the problem of gradient vanishing of the Sigmoid function and makes the convergence faster. Besides, it adds an LRN (Local Response Normalization) layer to make the accuracy higher and uses a dropout layer to reduce overfitting of the model.

2.1.3 GoogLeNet

GoogLeNet[7] v1 was born in 2014 and won the championship in the ILSVRC competition. Its performance is similar to that of the VGGNet born in the same year, but the amount of parameters is less than VGGNet. This model does not simply deepen the network but introduces the concept of Inception. The information of different scales of the image is extracted through multiple convolution kernels, and finally fused to obtain a better representation of the image.

2.1.4 Resnet

As the computing power continued to increase, deeper networks could be calculated, but the researchers found that the deeper the network, the less the model can be trained because it will encounter the problem of gradient disappearance or gradient explosion. Before the emergence of ResNet[8], people used BN, Relu, and other methods to alleviate this problem, but they still couldn't make the network deep enough.

In 2015, He Kaiming proposed the ResNet network, which was inspired by the idea of the LSTM control gate.

Compared with the traditional network: y=f(x), the formula of ResNet Block is: y=f(x) + x, which can be called skip connect.

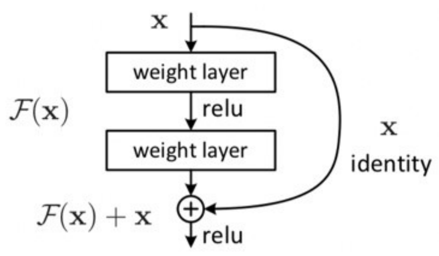

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 7: Residual Learning: a building block[8]

2.2 VGGNet

VGGNet[1] was proposed by the Visual Geometry Group of Oxford. The network proves that increasing the depth of the network can affect the final performance of the network to a certain degree. The two most widely used VGGNet models are VGGNet16 and VGGNet19. There is no essential difference between them, they only differ in network depth.

2.2.1 Principle of VGGNet

Compared with previous models, VGGNet uses 3×3 filter instead of 7×7 filter or 5×5 filter. For a given receptive field, using several small filters is better than using a large filter, because multiple non-linear layers can learn more complex patterns by increasing the network depth, and the cost is relatively small due to the fewer parameters. Simply put, the main purpose of this is to increase the depth of the network under the condition of ensuring the same receptive field and improve the effectiveness of the neural network to a certain extent.

For example, 3 3×3 filters with the stride of 1 can be regarded as a receptive field in size 7, and the total number of parameters is $3(9C^2)$. If we use the 7×7 filter, the total number of parameters is $49C^2$ (here C related to the number of channels). $27C^2$ is less than $49C^2$, which means that the number of parameters is reduced; and the 3×3 filter helps to better maintain image properties.

2.2.2 Structure of VGGNet

VGGNet has five sets of convolutional layers, and each set of convolutional layers is followed by a max-pooling layer. In the last few layers of VGGNet, three fully connected layers are used, and finally, there is a softmax layer.

2.2.3 Pros and Cons of VGGNet

The structure of VGGNet is very clear. The size of filters in convolutional layers are all 3×3 and in max-pooling layers are all 2×2, and using several small filters (3×3) in convolutional layers is better than using large filters (5×5 or 7×7).

What's more, each channel represents a FeatureMap, and more channels represent richer image features. The number of channels in the first layer of the VGGNet network is 64, and each subsequent layer has been doubled to a maximum of 512 channels. The increase in the number of channels allows for more information to be extracted.

Also, it verifies that performance can be improved by continuously deepening the network structure.

However, VGGNet still consumes a lot of computing resources and uses many parameters, most of which come from the first fully connected layer, resulting in more memory usage costs and longer model training time.

2.3 Related Applications

CNN has achieved great success in the field of image processing and image recognition. On the international standard ImageNet[9] data set, many successful models are based on CNN. One of the advantages of CNN over traditional image processing algorithms is that it avoids the complicated pre-processing process of the image and can directly input the original image.

2.3.1 Image Object Classification

Object classification is a basic problem in the field of machine vision. The classification of objects generally describes the image globally through some features and uses the classifier operation to determine whether there is a certain type of object in the image. For image data, CNN has excellent modeling and feature extraction capabilities, and have been widely used in theoretical analysis and practical applications of object classification.

2.3.2 Image Object Detection

Object detection is different from classification problems. It pays more attention to local features. It answers where an object exists in an image. Therefore, in addition to feature expression, object structure is the most obvious feature that distinguishes object detection from object classification.

3 ZYZNet

For most of the related research, the general trend is to design a deeper and more complex neural network to reach higher accuracy. VGGNet improves accuracy by increasing depth, but it is a waste of resources for HCCR and it is inefficient. To balance the efficiency and accuracy, ZYZNet tries to change the number of channels and the depth of the neural network to obtain the best accuracy efficiently. The following article will introduce a relevant method to solve the problem.

3.1 ZYZNet Baseline

3.1.1 ZYZNet Image Preprocessing

To reduce the amount of computation, ZYZNet inputs a 64×64 grayscale image. For example, if a point on the image is represented by grayscale, only 256 pixels need to be computed. For a color view, more than 16 million pixels need to be computed. Considering that we were only trying to recognize Chinese characters, the grayscale image would not lose much information.

VGGNet inputs a 224×224 image, but in HCCR, Chinese characters don't contain so much information. As it was said that using the data with 2^N size was a benefit for the GPU computing, we first tried to resize the image into 32×32, but the resolution ratio was too low. So we used a 64×64 image to make the computation more efficient and have high accuracy. Look at Figure 6:

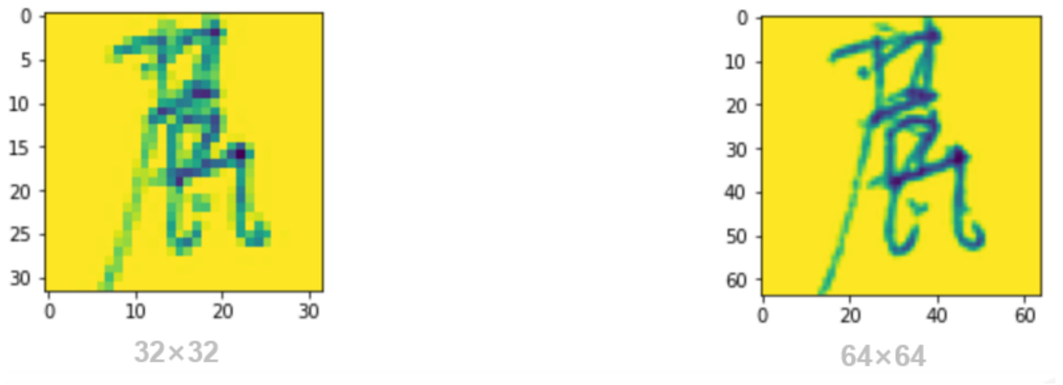


Figure 8: Contrast images with different number of pixels

3.1.2 Convolution and Pooling

ZYZNet uses 3×3 convolution filters and 2×2 max-pooling size like VGGNet[1]. Reusing 3×3 convolution filters can get the same receptive field as 5×5 convolution filters and 7×7 convolution filters. But Reusing 3×3 convolution filters will have fewer parameters and increase the nonlinear expression ability of the network.

As seen in Fig 7, 3×3 convolution filters are used to replace 5×5 convolution filters. Assuming that all data has C channels, then a separate 5×5 convolutional layer will contain $5C \times 5C = 25C^2$ parameters, while the combination of two 3×3 convolution layers has only $2 \times (3C \times 3C) = 18C^2$parameters. Similarly, the parameters applied to the 3×3 convolution filter in Fig 8 are also reduced. Thus, it can be intuitively shown that a combination of smaller convolutional layers is used instead of a larger convolutional layer. Use fewer parameters to get better results, which greatly reduces the amount of computation.

And by alternating the structure of the convolutional layer and activation layer, it can better extract feature information. Adding activation function and batchnorm2d in the middle of two convolution layers can extract the deep features of Chinese characters more smoothly and better.

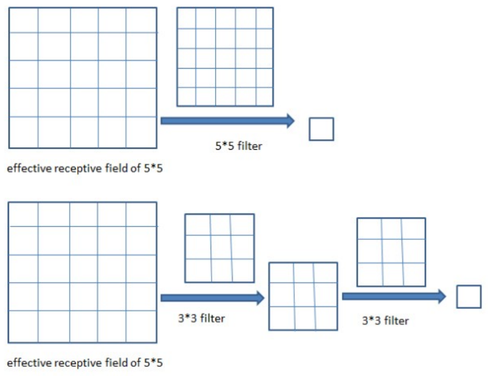
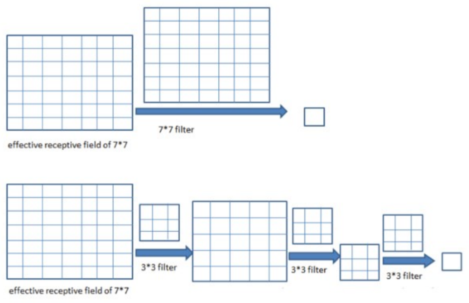

5*5 filter
effective receptive field of 5*5
3*3 filter
3*3 filter
effective receptive field of 5*5

Figure 9: After the convolution filters of two layers of 3×3, the receptive field i5×5, where the stride of the convolution filter is 1, the padding is 0



effective receptive field of 7*7
7*7 filter
effective receptive field of 7*7
3*3 filter
3*3 filter
3*3 filter

Figure 10: The receptive field after the three-layer 3×3 convolution filter operation is 7×7, where the convolution filter step size is 1, the padding is 0

3.1.3 Reduce the Fully Connected Layer

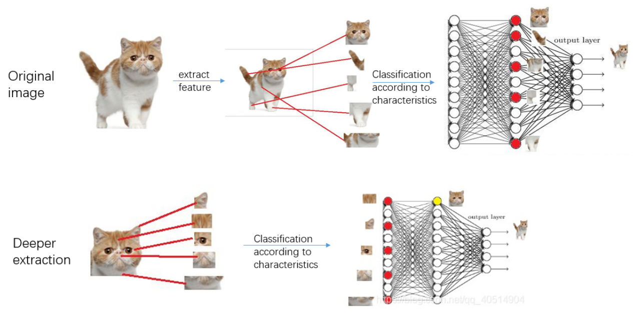
Compared to VGGNet, ZYZNet only has one fully connected layer. In VGGNet, A large number of parameters lead to more memory consumption, which leads to low efficiency and a waste of resources. The reason for a large number of parameters is to use fully connected layers many times. So in ZYZNet, we tried to reduce the use of a fully connected layer.

The role of the fully connected layer is classification. In VGGNet, the network needs to solve complex images, so VGGNet continuously extracts image features through the convolution layer, and then integrates features to recognize images through fully connected layers. However, in the HCCR task, Chinese characters do not subdivide the eigenvalues. Therefore, there is no need for multiple fully connected layers to gradually integrate features.

For example, as seen in Fig 9, using VGGNet recognizes a cat, We need to extract the features of each part, like head, tail, claws, etc. The features we find are the red dots in the picture. If the classification is based only on the head, tail, and other features, errors will easily occur due to too few eigenvalues. Then we need to extract the extracted eigenvalues and extract the head features. In this way, we use two fully connected layers for feature integration. One layer of fully connected layer represents extracting more fine features, which will be combined into the next fully connected layer. That is, small features will be aggregated into larger features, and finally, the image will be recognized.

As seen in Fig 10, however, in Chinese character recognition, there are not many eigenvalues to be extracted. Therefore, features can be aggregated with few fully connected layers. Therefore, the efficiency is greatly improved and the resources are saved.

After experimentation, it is best to use only one fully connected layer.



Original image
extract feature
Classification according to characteristics
output layer
Deeper extraction
Classification according to characteristics
output layer

Figure 11: Extracting cat eigenvalues. It is necessary to extract more detailed information and integrate features with multi-layer fully connected.

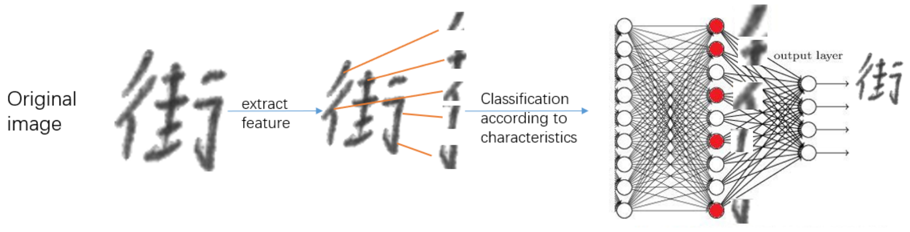


Figure 12: Extracting Chinese characters eigenvalues, There is no need to extract too much information in the image.

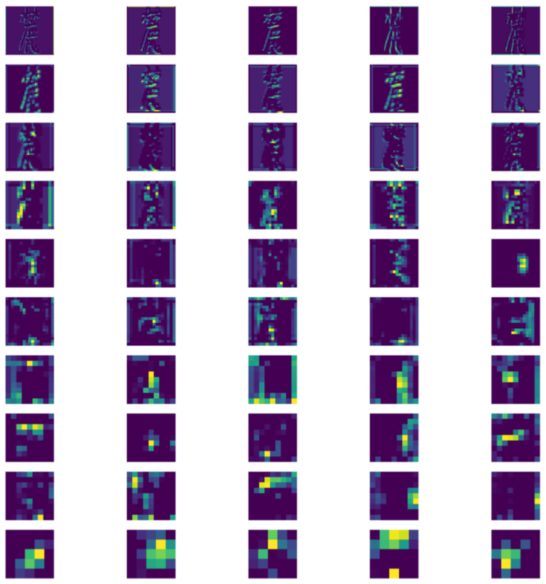


Figure 13: From top to bottom,Each convolution layer extracts the feature of the image

3.2 Make ZYZNet Robust

3.2.1 Data Augmentation

Data augmentation[10] is mainly to reduce the over-fitting phenomenon of the network. By transforming the training pictures, a network with stronger generalization ability can be obtained, which can better adapt to various application scenarios. We used the geometric transformation in supervised, single-sample data augmentation, which was to perform the geometric transformation on images, including various operations such as flipping vertically and horizontally, rotating in the range of -25 degrees to 25 degrees, and scaling. Several of these operations are shown below.

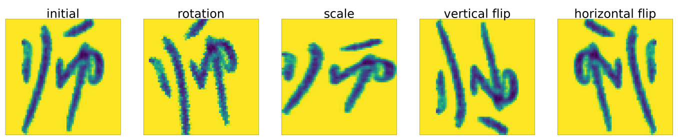


Figure 14: Geometric transformation of pictures

To achieve these transformations, we used some functions in the torchvision.transforms module, such as RandomRotation, RandomVerticalFlip, and RandomHorizontalFlip. But when it came to scaling, we wrote a function to randomly scale the images by the horizontal or vertical axis from 0.5 to 2. Then considering that all the input images should in the size of 64×64, we put the scaled image in the center and filled white around it, instead of scaling them again.

For the training dataset, we applied rotation and horizontal or vertical scaling to each image, to get more images different from each other, then flipped them vertically or horizontally or just not flip.

Concerning the testing dataset, we randomly selected one transformation from all the methods mentioned above for each image in it, to simulate the actual situation.

3.2.2 Adjusting Structure

Using the data augmentation method and the baseline model, we achieved 93% accuracy in the testing dataset with the robust model while 98% accuracy in the training dataset. Although 93% accuracy was high enough, we found that in the training dataset the model can achieve better performance. So it meant that the model still had potential. To achieve better performance, there were many methods like enlarging the datasets and using stronger data argumentation methods.

In this model, there were still too many parameters and too few training samples. It was easy to overfit. Finally, we added a dropout layer before the final fully connected layer, in which half of the parameters were randomly dropped.

Assuming there were N neurons in drop out layer. When training, every neuron had 0.5 possibilities to be dropped out. So there were 2^N kinds of real drop out layer. Using the drop out layer meant that we evaluated the output with 2^N kinds of models. After adding this layer, we achieved 96.4% accuracy in training datasets while 95.25% accuracy in testing datasets.

4 Model Architecture

4.1 Network Structure

The establishment of ZYZNet is based on VGGNet by changing input channel, depth and fully connected layer.

The whole convolution layer can be divided into four parts. Each part is composed of two convolution layers, which ensures the same receptive field as the 5×5 convolution filter. There are batchnorm2d layers and the ReLU layer behind each convolution layer, which ensures better feature extraction. At the end of each part, there is a pooling layer of max pooling2d. This structure ensures that there are fewer parameters, but it can extract feature information better.

There is a dropout layer between the convolution layer and the fully connected layer. Finally, the classification is carried out through a layer of fully connected.

Table 2 is the structure diagram of our network.

Table 2: ZYZNet body architecture

Type	in/out channels	filter size	stride	padding
Conv2d/s11	1/64	3	1	1
BatchNorm2d	64/64			
ReLU	64/64			
Conv2d/s12	64/64	3	1	1
BatchNorm2d	64/64			
ReLU	64/64			
max pooling2d	64/64	2		
Conv2d/s21	64/128	3	1	1
BatchNorm2d	128/128			
ReLU	128/128			
Conv2d/s22	128/128	3	1	1
BatchNorm2d	128/128			
ReLU	128/128			
max pooling2d	128/128	2		
Conv2d/s31	128/256	3	1	1
BatchNorm2d	256/256			
ReLU	256/256			
Conv2d/s32	256/256	3	1	1
BatchNorm2d	256/256			
ReLU	256/256			
max pooling2d	256/256	2		
Conv2d/s41	256/512	3	1	1
BatchNorm2d	512/512			
ReLU	512/512			
Conv2d/s42	512/512	3	1	1
BatchNorm2d	512/512			
ReLU	512/512			
max pooling2d	512/512	2		
dropout				
Linear	12800/3755			
log softmax	Classifier			

4.2 Activation Functions:ReLU

ReLU

$$f(x) = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases}$$

ReLU and its derivative curve As shown in Fig 13, it can be seen from the above figure that the effective derivative of ReLU is constant 1, which solves the problem of gradient vanishing in the deep network and makes the deep network trainable. At the same time, ReLU is also a nonlinear function. The so-called non-linearity means that the first derivative is not constant. For the derivation of ReLU, when the input value is positive and negative, the derivative is different, that is, the derivative of ReLU is not constant, so ReLU is nonlinear (its only difference from sigmoid and tanh, the non-linearity of ReLU is not smooth).

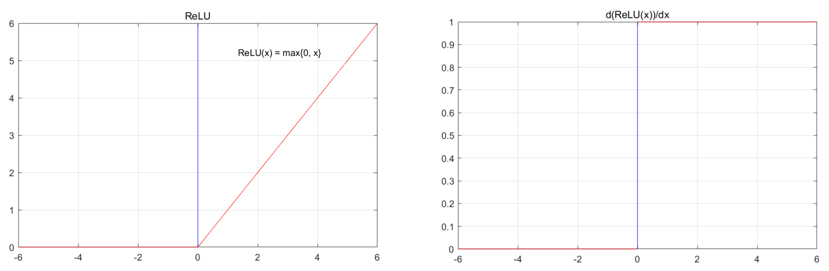


Figure 15: ReLU and its derivative curve

ReLU under x>0, derivative characteristics The advantage of constant derivative 1 is that the gradient will not disappear in the "chain reaction", but the intensity of the gradient decrease depends entirely on the product of weights, so the gradient explosion problem may occur.

ReLU under x<0, derivative characteristics

Characteristics of ReLU as activation function Compared with sigmoid and tanh, ReLU abandons the complex calculation and improves the convergence speed. The gradient vanishing problem is solved, and the convergence is faster than sigmoid and tanh functions, but the gradient explosion of ReLU should be prevented.

It is easy to get a better model, but it is also necessary to prevent the "dead" model in training.

4.3 BatchNorm2d

We use BatchNorm2d to solve the gradient explosion problem caused by the use of ReLU.

Principle Normalize the data. This prevents network performance from being unstable due to too much data before ReLU.

Formula:

$$y = \frac{x - E[x]}{\sqrt{Var[x] + \epsilon}} * \gamma + \beta$$

$E[x]$ is the mean and $Var[x]$ is the standard deviation, others are trainable variables. In the process of deep neural network training, usually, one batch of training data is used instead of all the data. The different distribution of each batch resulted in an internal Covariate shift problem – during the training process, the data distribution would change, which will bring difficulties to the learning of the next layer of the network. The Batch Normalization forces the data back to a positive distribution with the mean value of 0 and variance of 1, which keeps the data distribution consistent on the one hand and avoids gradient vanishing on the other.

4.4 Loss functions:Cross Entropy

Log softmax Z donates the output tensor. Softmax can transfer the value from 0 to 1. And the sum of the tensor is 1. So the elements of the tensor can be seen as the possibilities. And we can see that HCCR is a classification problem. Therefore, we need to evaluate the difference in possibility. So we can use the Kullback-Leibler Divergence to calculate the difference.

The only difference between softmax and log softmax is that for log softmax every element will go through the log to avoid numerical overflow.

$$\operatorname{softmax} Z = \frac{\exp Z}{||\exp Z||}$$

$$\text{logsoftmax}\, Z = \log(\frac{\exp Z}{||\exp Z||})$$

NLLLoss Actually, cross-entropy is a specific case of Kullback-Leibler Divergence. So we can use it as the loss function to evaluate the difference among many possibility distributions.

$$\text{loss}(p, x) = -\sum x * \log(p)$$

4.5 Dropout

In the machine learning model, if there are too many parameters and too few training samples, the trained model is likely to have the overfitting phenomenon and it will take a lot of time.

steps First, half of the hidden neurons in the network are randomly (temporarily) deleted, leaving the input and output neurons unchanged.

Then the input X is propagated forward through the modified network, and the loss result is propagated back through the modified network. After executing this process in a small batch of training samples, the corresponding parameters (W, b) were updated on the neurons that were not deleted according to the stochastic gradient descent method.

And then keep repeating the process.

4.6 Optimizer:Adam

Adam uses momentum. And unlike SGD, Adam uses an adaptive learning rate for each parameter to speed up convergence.

Exponentially Weighted Averages Exponentially weighted averages method is a simple but useful way to calculate the average of recent ranges. Let θ_t be a function, while β donated a super parameter of exponentially weighted averages.

$$v_t = \beta v_{t-1} + (1-\beta)\theta_t$$

v_t can be seen the average of t from $t - \frac{1}{1-\beta}$ to $t-1$

$$v_t \approx (1-\beta) \sum_{i=t-\frac{1}{1-\beta}}^{t-1} \theta_i$$

Momentum The momentum algorithm accelerates the stochastic gradient descent in the relevant direction and inhibits the oscillation. During parameter update, the direction of the previous update is retained to a certain extent, while the gradient of the current batch is used to fine-tune the final update direction, in short, the current gradient is accelerated by accumulating the previous momentum. In Adam, the momentum for the specific parameter is donated by m_t, while the gradient for the same parameter is g_t and the super parameter is β_1. Using exponentially weighted averages, m_t can be calculated in this way.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t$$

Adaptive Learning Rate Adam uses second-order momentum to adapt the learning rate for each parameter, in order to speed up convergence and get better performance. The second-order momentum is the average of g_t^2. Adam uses this method to speed up those parameters which update not so often and lower the speed for others. Let v_t be the second-order momentum and the super parameter is donated by β_2, then v_t can be calculated in this way.

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)g^2(t)$$

Momentum Correct In Adam, we usually set $v_0 = 0$, and $m_0 = 0$, which can cause some problems. So instead of using v and m, we use $\hat{v}$ and $\hat{m}$ to avoid the problems cause by the initial value.

$$\hat{v}_t = \frac{v_t}{1-\beta_1^t}, \hat{m}_t = \frac{m_t}{1-\beta_2^t}$$

Update Through momentum and adaptive learning rate, the update in each can be represented by the following formula, the parameter is donated by θ.

$$\theta_t = \theta_{t-1} - LearningRate * \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}}$$

5 Experiment

5.1 Dataset

We chose CASIA-HWDB1.1, an offline HCCR database constructed by the Institute of Automation, Chinese Academy of Science, as our dataset. The CASIA-HWDB1.1 set contains 3755 classes of level-1 Chinese characters in GB2312-80 (a Coded Chinese Character Set for Information Communication), and for each character, there are about 240 images of different handwritten versions sampled from 300 writers for training and 60 for testing. 897758 images are included in the training dataset, and 223991 in the testing dataset.

5.2 Adjusting Parameters

5.2.1 Learning Rate

The learning rate is the most important parameter in training. The learning rate determines the network parameters to a great extent. If we use a larger learning rate, it may lead to a very large range of parameter updating and unfitness. If we choose a smaller learning rate, it will lead to a very slow decline in network loss and lead to the local optimum, which wastes a lot of resources and is not efficient.

For the selection of the learning rate, we referred to Leslie N. Smith's method[11]. We did it iteratively. First of all, a relatively small learning rate was determined, and then the optimal learning rate was selected.

As the number of epoch increases, the learning rate can be changed to achieve the best effect. In the beginning, a large learning rate was used to accelerate the convergence and prevent the occurrence of a locally optimal solution. In the later stage, we used a smaller learning rate to approach the optimal solution.

Finally, the learning rate was 0.000042 in the baseline, while in the robust model, the learning rate was 1e-3 in the first 25 epoch,26 - 35 was 1e-5,36 - 40 was 1e-7. See Figure 12 for details.

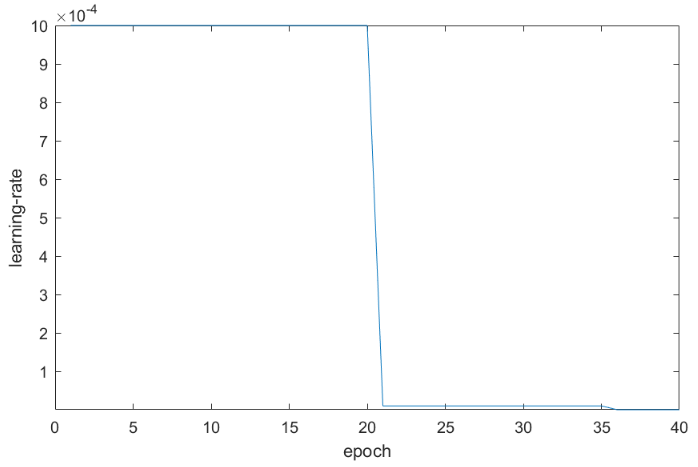


Figure 16: With the training going on, the change learning rate achieved the best effect

5.2.2 Convolution Layer Depth

There is not so much feature information in Chinese characters as in other images, so it does not need too many convolutional layers to extract the feature information, thus it is possible to greatly reduce the amount of computation by reducing the number of convolutional layers, and still has a good effect.

5.3 Result

5.3.1 Baseline

In the case of a batch size of 48, the training accuracy was increased to 97.8% after only 5 epochs training.and the test accuracy was increased to 93.9%. Loss decreased significantly. Train loss decreased to 0.0377, and test loss decreased to 0.25. See Figure 13 for details.

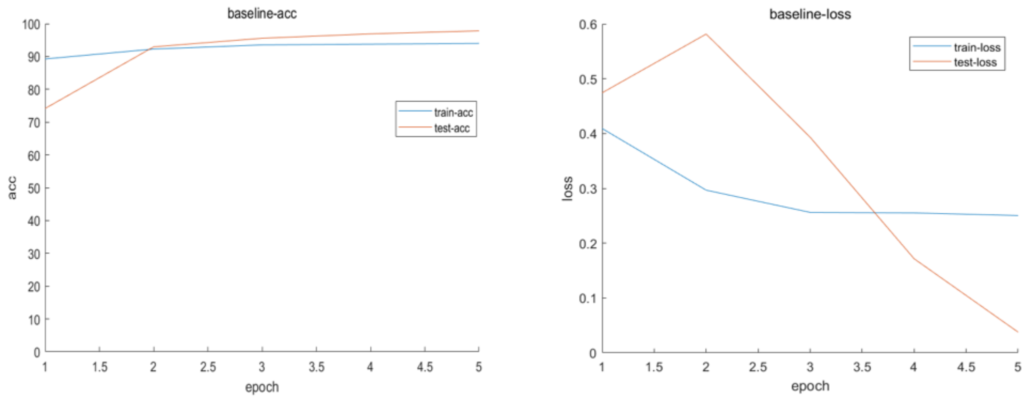


Figure 17: Baseline had good results after very few epochs

There was no unfit phenomenon, and there was space for improvement. It had a good effect on the baseline, which provided feasibility for continuing to make the model robust.

5.3.2 Robust model

Because this network performed very well in the baseline training, it continued to be used when we wanted to train a model that is more robust. With the batch size as 64, the network trained 40 epochs and accuracy showed an upward trend. Finally, train accuracy was stable at 96.4% and test accuracy was stable at 95.2%. Train loss fluctuated in a small number and test loss was stable at 0.195. See Figure 14 for details.

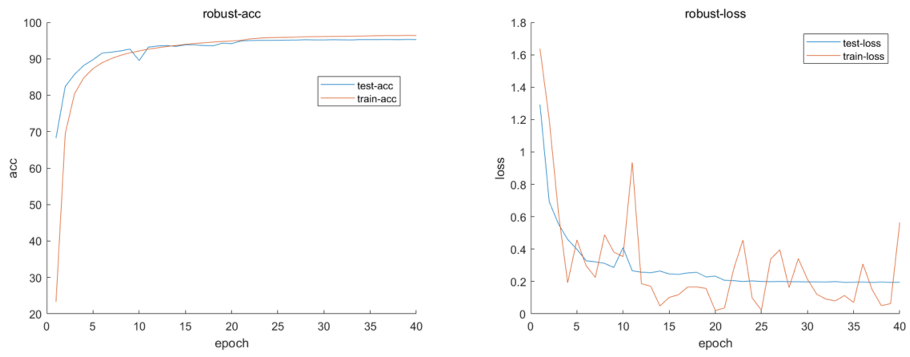


Figure 18: Robust result had good results after few epochs

From the above experimental data, we can see that ZYZNet has an excellent performance in solving HCCR tasks. While saving resources, ZYZNet can achieve higher accuracy and improve efficiency.

6 Conclusion

To solve the HCCR task, we considered using VGGNet but found that VGGNet was inefficient in solving this problem and wasted resources. Therefore, we designed the ZYZNet by changing the input image data specifications, changing the convolution filter rules, reducing the number of fully connected layers, etc. When used in solving the HCCR task, the network has very good results. Therefore, a solution that balances efficiency and accuracy has been found.

Acknowledgement

We would like to thank Prof. Chandrajit Bajaj from the University of Texas at Austin for his excellent teaching and patient explaining, and Prof. Jianlong Wu from Shandong University for all the suggestions and guidance he offered in this project. Without their supervision and dedication, we would not have done such amazing work as beginners of Deep Learning.

References

[1] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[2] Ruwei Dai, Chenglin Liu, and Baihua Xiao. Chinese character recognition: history, status and prospects. *Frontiers of Computer Science in China*, 1(2):126–136, 2007.

[3] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011.

[4] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[5] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems 25*, pages 1097–1105, 2012.

[7] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[8] He Kaiming, Zhang Xiangyu, Ren Shaoqing, and Sun Jian. Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[10] Khoshgoftaar T.M Shorten C. A survey on image data augmentation for deep learning. *Big Data*, 2019.

[11] Leslie N Smith. Cyclical learning rates for training neural networks. *Computer ence*, pages 464–472, 2015.