ISSN 2616-5775 Vol. 5, Issue 8: 1-11, DOI: 10.25236/AJCIS.2022.050801

The Design and Realization of Intelligent Classification of Rock Samples Based on Convolutional Neural Network

Quan Hao, Nan Nie, Fuxi Zhu

School of Information Engineering, Wuhan College, Wuhan, 430212, China
quan.hao.apply@outlook.com

Abstract: *With substantial improvement of computer hardware performance and rapid development in artificial intelligence research, artificial intelligence recognition technology has been applied in many industries, such as medicine, education, industry, commerce, exploration and other professional fields. In geological exploration, the classification and identification of rock samples is an important link in geological analysis. However, the traditional manual classification faces high economic and time costs, and is vulnerable to subjective judgment which may affect the recognition result. In order to avoid these problems, research is carried out on the specific application of computer-based artificial intelligent recognition technology in recognition of rock images and the performance difference caused by different parameters. The sample data used in the experiment are lithic fragments and core samples capture by industrial cameras at the well logging site, with a total of 350 images which cover 7 types of rock samples. In this paper, the 350 rock images are divided into training set, validation set and test set based on certain proportion. Since the data set is too small and the quantity of each type of rocks varies, data augmentation technology is used to expand the original data before training. During the expansion, the quantity of different types of rock samples are balanced. At the training stage, the open-source deep learning framework Pytorch is used to construct a multi-layer convolutional neural network for learning and classification training on the training set. The best accuracy rate in the tests reaches 82.86%. However, the recognition rate is poor for stretched images, which required further optimization of the network structure. Based on the experimental results, the optimization orientations of neural network training are summarized, including improvement of data set quality, optimization of network structure, and adjustment of training strategies. In order to display the experimental data as intuitively as possible, data visualization tools such as PlotNeuralNet and Python matplotlib are used in the experiment for specific visualization work, after which the experimental data and model effect are analyzed from the perspective with combination of data and images.*

Keywords: *in-depth learning; convolutional neural network; recognition of rock sample images*

1. Introduction

1.1 Research background

For computers, pictures are stored in an unstructured form, where their information cannot be directly obtained. Therefore, the efficiency of information retrieval is significantly impeded. It is difficult to process hundreds of millions of images manually. In the era of big data, the speed of processing data is even more critical. Deep learning is one way to realize intelligent recognition. The growing amount of data and model size is making deep learning technology increasingly reliable. At the same time, the development of modern Convolutional Neural Networks (CNN) for object recognition also significantly improves the efficiency of computer image recognition.

Rock image recognition is a field of geological research where image recognition technology is applied. The recognition of the types of rock samples is of great significance in oil and gas exploration. However, there are great drawbacks in traditional artificial image recognition of rock samples, including the following:

(1) Low efficiency and high cost;

(2) The difficulties in extracting deeper and more complex abstract features;

(3) Vulnerability to subjective judgment with regard to the recognition results.

Programs based on computer artificial intelligence recognition technology can efficiently extract

ISSN 2616-5775 Vol. 5, Issue 8: 1-11, DOI: 10.25236/AJCIS.2022.050801

features from images, effectively avoid the negative impact of subjective recognition on classification results, and reduce various costs of manual recognition, providing a feasible new approach for rock sample image recognition.

2. Design of Neural Network Structure

2.1 Design of convolutional layer

The kernel function of the CNN convolutional layer can a given image spatially to detect features such as edges and shapes. In essence, these kernel functions learn to capture spatial features from images based on weight learned through backpropagation. Furthermore, the stacked convolutional layers can be used to detect complex spatial shapes from spatial features at each subsequent level. Hence, they can extract highly abstract features of images that are imperceptible to humans.

Please refer to (equation 1) for the calculation formula of convolutional layer in Pytorch. The input format of convolutional layer is: (N, C_{in}, H, W); the output format is: $(N, C_{out}, H_{out}, W_{out})$. Where, N is batch size; C is the number of channels; H and W are the length and width of data whose unit is pixel.

$$\mathrm{out}(N_i, C_{\mathrm{out}_j}) = \mathrm{bias}(C_{\mathrm{out}_j}) + \sum_{k=0}^{C_{\mathrm{in}}-1} \mathrm{weight}(C_{\mathrm{out}_j}, k) \star \mathrm{input}(N_i, k) \quad (1)$$

In order to better extract the abstract features, 4 convolutional layers are provided for the neural network in the experiment. The parameters are as follows:

Conv2d(1, 32, kernel_size=5, stride=2, padding=2)

Conv2d(32, 32, kernel_size=4, stride=1, padding=2)

Conv2d(32, 64, kernel_size=3, stride=1, padding=2)

Conv2d(64, 64, kernel_size=3, stride=1, padding=2)

The first parameter of the function means the number of input channels, namely, C_{in} . Since each convolution needs to receive input from the neurons in the upper layer, the number of input channels in each layer is related to the output of the upper layer. The second parameter of the function is the number of output channels, namely, C_{out}. The third parameter is the size of kernel function. Initially, a larger size is used to facilitate extraction of more global features. Then, a smaller size kernel function is used for subsequent convolutional layers, so as to recognize small local features. The fourth parameter is the number of strides of kernel function and used to reduce the size of output image. Initially, a larger number of strides are used. Then, subsequent strides are set to 1. The fifth parameter is the padding size. Padding is usually used to add columns and rows of zeros to keep the spatial size unchanged after convolution, so that the performance may be improved due to preservation of information at the boundaries. In order to ensure the correct input and output of the model, the padding size is set to 2 uniformly.

2.2 Design of pooling layer

The pooling layer is used to reduce the size of the output of the cross-correlation operation, which reduces the amount of computation in the network and the parameters that need to be updated. The pooling layer summarizes the features in the feature map output by the cross-correlation operation.

Pooling behaves like a kernel function. The pooling operation consists of sliding a two-dimensional matrix over each channel in the output of the cross-correlation operation and summarizing certain features within the area covered by the matrix. For a pooling layer of size 2×2, it reduces the number of pixels or values in each feature map to a quarter.

There are two commonly used pooling operations, namely average pooling and maximum pooling. This paper uses maximum pooling.

2.2.1 Max pooling

Max pooling is used to compute the maximum value among the specified amount of data in each feature map. The max pooling highlights the most significant features of the feature map after sampling

ISSN 2616-5775 Vol. 5, Issue 8: 1-11, DOI: 10.25236/AJCIS.2022.050801

or pooling. Max pooling is more suitable for extracting salient features, while average pooling sometimes fails to extract good features because it takes all values into account and produces an average value, which may be less effective for tasks such as image classification. In addition, as average pooling takes all values into account and passes them to the next layer as output, all values are used for feature mapping and output, resulting in lowered computational inefficiency. If it is not necessary to consider all the inputs from the convolutional layers, then average pooling is not very suitable.

In conclusion, max pooling generally outperforms average pooling in classification. Therefore, it is preferably used as the pooling layer. In the code, each convolutional layer is followed by a pooling layer, and they all have the same form: nn.MaxPool2d(kernel_size=2, stride=2). The size of the pooling layer is designated as 2×2, while the number of strides is 2. The concept of the number of strides is the same as that in the convolutional layer.

2.3 Activation function

In order to effectively train a multi-layer neural network, a nonlinear function similar to a linear function is needed to serve as the activation function. What's more, it must also be more careful about the activation and input, and inhibit saturation. The node or unit that implements this activation function is called a rectified linear activation unit, or ReLU for short. Please refer to (equation 2) for its definition.

$$f(x) = \max(0, x) \tag{2}$$

For values greater than zero, ReLU exhibits the characteristics of a linear function. When backpropagation is used to train the neural network, it demonstrates many ideal properties of linear activation function. Nonetheless, it is a non-linear function, and the output is 0 if the input value is negative. Compared with traditional machine learning activation functions, ReLU has some merits which are suitable for deep neural network training, including:

(1) Better backpropagation: Because the derivative is 0 when the input is negative, the neuron is not activated and will not get smaller and smaller, it is less likely that there will be vanished gradient.

(2) Higher computational efficiency: Due to simplicity of definition of the function, ReLU only includes comparison, addition and multiplication. Therefore, the compiler can optimize the calculation through displacement operations. At the same time, since only some neurons are activated, the computational efficiency of ReLU is much higher than that of sigmoid and tanh.

(3) Acceleration of gradient decline: Due to the linearity and non-saturation of ReLU, the neural network can accelerate gradient decline toward the convergence of global minimum of loss function.

In view of these advantages, ReLU is preferentially used as the activation function for neural network training in the experiment.

2.4 Batch normalization

Batch normalization is a method of coordinating the adjustment parameters of multiple layers in a neural network. It can readjust the output from the previous layer, and normalize the mean and variance of each unit to stabilize the learning efficiency. The normalization of output of the previous layer in the neural network will prevent significant change of output distribution of subsequent layers when their parameters are adjusted, which improves the stability and the multi-layer model training. What's more, batch normalization can also optimize performance, reduce the number of trainings, and smooth the corresponding optimization.

In Pytorch, torch.nn.BatchNorm2d() function is used to achieve batch normalization. Please refer to (equation 3) for its definition. γ and β are parameter vectors with a size of C which can be learned. If it's not specifically designated, γ is set as 1, β is set as 0. $\hat{x}$ means the data after standardization. The standard deviation is calculated through torch.var(input, unbiased=False). The first parameter of BatchNorm2d() is the number of output channels, namely, C_{out}. It is dependent on the number of output convolutional layers.

$$y = \gamma\hat{x} + \beta \tag{3}$$

ISSN 2616-5775 Vol. 5, Issue 8: 1-11, DOI: 10.25236/AJCIS.2022.050801

2.5 Design of fully connected layer

The fully connected layer generally constitutes the last layers of network, with its input being the output of flattened final pooling layer or convolutional layer. The output of a convolutional layer represents abstract features of data. Although the output can be directly flattened and connected to the output layer, the adoption of fully connected layer makes it easier for non-linear combination to learn these features. The fully connected layer will map the output to the sample space and complete the final classification.

In Pytorch, fully connected layer is achieved through use of torch.nn.Linear() function, where input data undergoes linear transformation. The first parameter of the function represents the size of the input samples, and the second parameter represents the size of each output sample. In order to improve the execution efficiency, double-layer full connection is used in the experiment.

2.6 Visualization of network structure

PlotNeuralNet is an open-source visualization tool for network structure. The network structure generated with it is shown in Figure 1.(Figure1 is only a schematic, the scale of the figure shown by it is not the actual scale of the data) The input of the first layer of the network is an 80×80 image with 1 channel. After the convolution operation of the first layer, a maximum pooling is performed. The kernel size of each pooling layer is 2 and the number of strides is 2. After pooling, batch normalization is performed again, and the final window shape is 40×40. The change of window size can be observed from the image. The output of the first layer is used as the input of the second layer, and so on. After two full connections, the result is mapped to the number of classes 7.

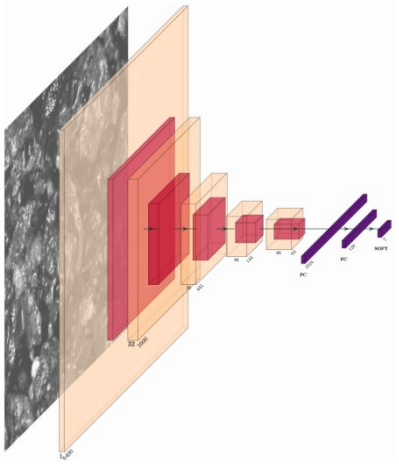

Figure 1: Visualization of network structure

2.7 Loss function

In dealing with machine learning or deep learning, loss function is an evaluation algorithm which can calculate the effectiveness of model on the data set. Its output is called Loss. Generally speaking, in the field of intelligent recognition, the smaller the loss, the better the fitting effect of the neural network on the data set, while the larger the Loss, the poorer the prediction effect of the model on the data set. In improving the model, loss can be well reflective of whether the improvement is effective. The loss function is related to model accuracy, being a key component of the parameters of neural networks adjustment model. For each prediction by the model, the loss function explicitly gives the absolute difference between the neural network's prediction and the actual value.

Cross entropy is one of the most popular loss functions. It is a measurement method from information theory. Usually, it calculates the difference between two probability distributions. Since it minimizes the distance between the predicted and actual probability distributions, it demonstrates excellent performance in classification. In training the neural network, the model will adjust various parameters in the network based on the value of loss, so as to seek a smaller cross entropy loss. Please refer to (equation 4) for the definition of cross-entropy. Where, n means the number of classes; t_i means the actual situation; p_i means the probability of i^{th} class.

ISSN 2616-5775 Vol. 5, Issue 8: 1-11, DOI: 10.25236/AJCIS.2022.050801

$$L_{CE} = -\sum_{i=1}^{n} t_i \log(p_i) \tag{4}$$

In Pytorch, cross-entropy is achieved through torch.nn.CrossEntropyLoss() function. During backpropagation, the neural network updates the parameters of the model through loss.

2.8 Optimizers

An Optimizer is an algorithm used to adjust model parameters (such as weights and learning rate) to reduce loss. It reduces the output of loss function by adjusting the model, serving as a bridge between the loss function and the model parameters. The loss function is an optimization compass and tells the optimizer whether to optimize the weight in the appropriate direction. If the adjustment direction is wrong, it also gives feedback to the optimizer. Initially, it is impossible to know what parameters of the model are appropriate in the first place. Nonetheless, with the feedback of loss function, a parameter combination with lower Loss can be found finally.

There are many commonly used optimizers, including the most basic gradient descent, Stochastic Gradient Descent (SGD) as one of the gradient descent variants, AdaGrad, RMSProp, and Adam etc. Although the ideas of these algorithms vary, they are all designed to minimize Loss. In view of this, whichever is easy to debug may be adopted. In this experiment, Adam is adopted as the optimization algorithm for neural network training.

Adam [1] is an algorithm for gradient descent optimization and is suitable for circumstances of huge amount of data and complex parameters. It is one of the algorithms for optimization of adaptive learning rate. Compared with other optimization algorithm, it requires less memory but achieved higher efficiency. The pseudo-code of the Adam's calculation process algorithm is as follows:

for $t = 1$ **to** ... **do**
  $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
  if $\lambda \neq 0$
    $g_t \leftarrow g_t + \lambda\theta_{t-1}$
  $m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$
  $v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^2$
  $\widehat{m_t} \leftarrow m_t / (1-\beta_1^t)$
  $\widehat{v_t} \leftarrow v_t / (1-\beta_2^t)$
  if $amsgrad$
    $\widehat{v_t}^{max} \leftarrow \max(\widehat{v_t}^{max}, \widehat{v_t})$
    $\theta_t \leftarrow \theta_{t-1} - \gamma \widehat{m_t} / (\sqrt{\widehat{v_t}^{max}} + \epsilon)$
  else
    $\theta_t \leftarrow \theta_{t-1} - \gamma \widehat{m_t} / (\sqrt{\widehat{v_t}} + \epsilon)$

return θ_t

Adam requires 6 main parameters, namely the number of strides γ, that is, the learning rate; the exponential decay rates β_1 and β_2 of moment estimation; the small constant ϵ used for numerical stability; the initial parameter θ; The weight decay coefficient λ. Where, β_1 and β_2 are 0.9 and 0.999 by default; ϵ is 1e-8 by default; λ is 0 by default.

In the initialization of the algorithm, β_1 and β_2 are initialized to 0, representing the first momentum and the second momentum), respectively. The term "momentum" comes from physics and it is defined as mass times velocity. In neural networks, the first momentum, also known as standard momentum, is used to speed up training mainly for dealing with gradients with high curvature. The second momentum uses Nesterov [2] momentum which a variant of first momentum that adds a correction factor to standard momentum.

In the algorithm flow, the gradient of the relevant variable is first calculated. If λ is not 0, then the gradient calculation is involved. Afterwards, the first momentum and the second momentum are updated respectively, with deviations corrected. In updating parameters, amsgrad refers to whether the variant of

ISSN 2616-5775 Vol. 5, Issue 8: 1-11, DOI: 10.25236/AJCIS.2022.050801

Adam [3] is used, but this variant is not necessarily better and is usually not used. In computing updates, ϵ is added to the denominator, so as to improve numerical stability.

3. Network Training Process

3.1 Training achievements

Due to the uncertainty of hyperparameters of neural network, a large number of hyperparameter combinations are used in the experiment. Some data is shown in Table 1. Where, AOV means to the accuracy of the validation set; AOT means to the accuracy of the test set; and Dropout_p means the probability of Dropout. For a data item numbered n, it is called data item n or parameter group n. Among them, the highest accuracy rate of the validation set is 93.9%, and the highest accuracy rate of the test set is 82.86%, suggesting satisfactory generality.

Table 1: Experimental data

S.N.	LR	Bath Size	Epoch	Dropout_p	AOV (%)	AOT (%)
1	0.0001	128	50	0.0	62.62	54.29
2	0.00025	128	50	0.0	66.62	55.29
3	0.0001	128	100	0.0	61.95	57.14
4	0.0001	128	200	0.0	62.81	54.29
5	0.0025	128	100	0.2	88.81	54.29
6	0.001	128	200	0.2	92.43	68.57
7	0.005	128	200	0.2	87.67	65.71
8	0.001	128	300	0.2	93.90	65.71
9	0.0025	128	300	0.2	90.57	68.57
10	0.0005	256	200	0.2	91.76	62.86
11	0.0001	256	300	0.5	63.33	68.57
12	0.00025	256	300	0.5	73.48	68.57
13	0.0005	256	300	0.5	80.67	65.71
14	0.0001	256	121	0.2	77.57	77.14
15	0.0001	256	348	0.5	65.24	71.43
16	0.00001	128	143	0.2	62.24	71.43
17	0.0025	256	25	0.5	60.19	71.43
18	0.00025	128	20	0.2	66.38	71.43
19	0.0025	256	20	0.2	72.05	74.29
20	0.00001	256	400	0.2	66.57	71.43
21	0.025	256	189	0.5	65.38	80.0
22	0.0025	128	67	0.2	79.52	80.0
23	0.0025	256	81	0.2	87.57	82.86
24	0.0005	256	117	0.5	70.71	80.0
25	0.0001	256	141	0.2	78.81	80.0

4. Comparative Analysis of Results

4.1 The influence of parameters to training

After obtainment of the experimental data, the actual impact of different parameters on the model training effect is observed. Then, the law behind is summarized, with an attempt to explain the reasons.

4.1.1 Learning rate

In the experiment, the specific performance of 5 different learning rates is taken into account. The experimental data is shown in Table 2.

ISSN 2616-5775 Vol. 5, Issue 8: 1-11, DOI: 10.25236/AJCIS.2022.050801

Table 2: Experimental data

S.N.	LR	Bath Size	Epoch	Dropout_p	AOV (%)	AOT (%)
1	0.1	128	200	0.2	14.29	8.57
2	0.01	128	200	0.2	89.81	60.0
3	0.001	128	200	0.2	86.43	54.29
4	0.0001	128	200	0.2	84.86	65.71
5	0.00001	128	200	0.2	63.81	65.71

Figure 2 shows the variation of decline of loss with 5 different LR values. It can be seen from the figure that, where Adam optimizer is used, the adoption of a particularly large LR (0.1) will cause loss fail to converge at all; while a particularly small LR (0.00001) will cause quite slow decline of Loss.

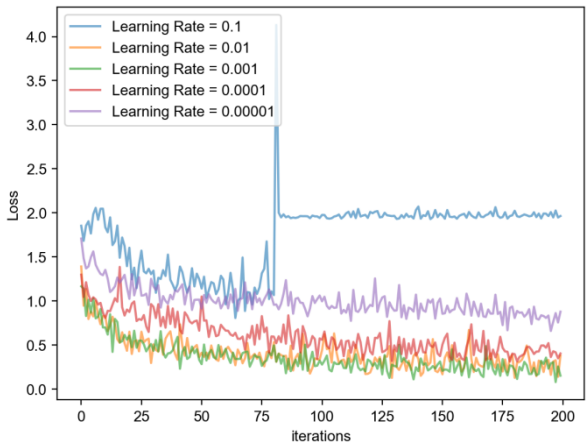

Figure 2: Variation of Loss with different LR values

The learning rate determines the training speed of neural network model, and its changes can lead to significant differences in the accuracy of the model. Figure 3 shows the variation of their validation set accuracy. It can be observed that the smaller the LR, the smoother the rise, and the larger the LR is, the steeper the rise.

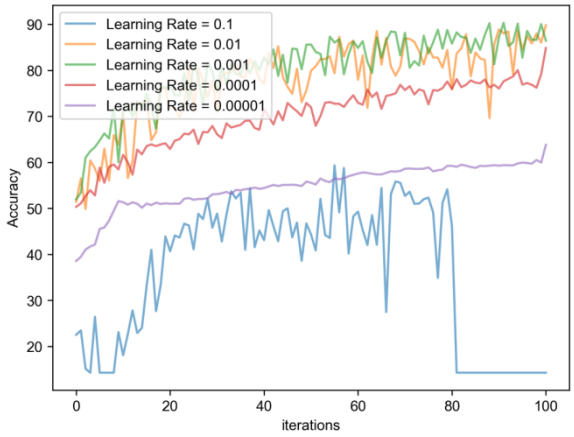

Figure 3: Variation of accuracy with different LR values

Note that when the LR is 0.1, the trend of the loss image is special. It does not show an overall downward trend like that of other parameters. Instead, it does not change much after reaching close to a specific value. Nonetheless, the accuracy of its validation set even no longer changes after 80 iterations, indicating that the output of the model is always fixed regardless of the input. Figure 4 shows some of the internal parameter values of the model obtained with different LR values. It can be seen that when the LR is 0.1, the internal parameter values of the model are considerably large compared to other models. Usually, in these circumstances, it can be considered that the loss has been trapped in some local optimal solutions during the training of neural network.

Although the learning rate based on the Adam optimizer can be adaptive, the neural network may be still trapped in local optimal solutions if excessively large learning rate is set at the beginning. Generally

ISSN 2616-5775 Vol. 5, Issue 8: 1-11, DOI: 10.25236/AJCIS.2022.050801

speaking, adoption of a smaller learning rate can avoid this circumstance.

LR = 0.1

```
Parameter containing:
tensor([[[[ 2.5554e+00, -1.6689e+00, -5.0392e-01, -8.9879e+01, -6.4340e-01],
          [ 1.8878e+00, -4.4330e+00, -8.6627e-01,  9.2810e+00,  7.4673e+00],
          [ 1.7810e+00,  5.5084e+00, -4.6183e+00, -1.0437e+00, -4.4517e+01],
          [ 3.6232e+00, -6.5945e+00, -5.5476e+00,  7.9669e+00,  3.5062e+00],
          [ 3.2668e+00,  3.8261e+00,  3.3876e+00, -2.6132e+00,  8.3545e+00]]],
```

LR = 0.01

```
Parameter containing:
tensor([[[[ 5.6227e-02,  8.5790e-02,  3.1453e-02,  5.5246e-02,  6.6818e-02],
          [ 1.0392e-01,  6.4525e-02,  4.5259e-02,  5.5455e-02,  3.0687e-02],
          [ 4.0720e-02,  1.4878e-02,  4.3192e-02,  2.8184e-02,  3.8877e-02],
          [-5.9350e-01, -1.1149e+00,  1.7512e-02,  7.4849e-02,  5.7196e-02],
          [ 3.8742e-02, -6.6172e-02,  5.0166e-02,  3.4816e-02,  3.4501e-02]]],
```

LR = 0.001

```
Parameter containing:
tensor([[[[ 1.1970e-01,  2.9219e-01,  3.3396e-01,  1.7131e-01,  1.1897e-01],
          [ 4.5691e-01,  1.0750e-01,  1.8751e-01, -3.6499e-01,  8.8689e-02],
          [ 7.9242e-02,  3.2702e-01,  5.3790e-01, -1.4912e+01,  3.2144e-01],
          [ 2.6423e-01,  2.8455e-01,  4.1871e-01,  6.7888e-01,  4.4491e-01],
          [ 1.3846e-01,  7.2433e-02,  3.2246e-01,  3.3196e-01,  1.5842e-01]]],
```

Figure 4: Partial parameters within the model with different LR values

In training a model, it is not easy to seek the global optimal solution of the neural network, as it may require a lot of experience and practice. However, some local optimal solutions are actually quite close to the global optimal solution, with insignificant difference. However, when the learning rate is 0.1, loss is finally around 2.0, which is much larger than the loss at other learning rates. Therefore, this result is still quite far from the global optimal solution.

4.1.2 Batch size

Batch size mainly affects the computation time spent by the model on each Epoch and the smoothness of loss decline. Figure 5 shows the variation of Loss under different batch sizes. It can be observed from the figure that the larger the batch size, the smoother the loss decline. However, excessively large Batch Size (4096) may also cause slow convergence of the model. Nonetheless, the smaller the batch Size, the more significant the oscillation when the model's loss changes, which is also not favorable for convergence.

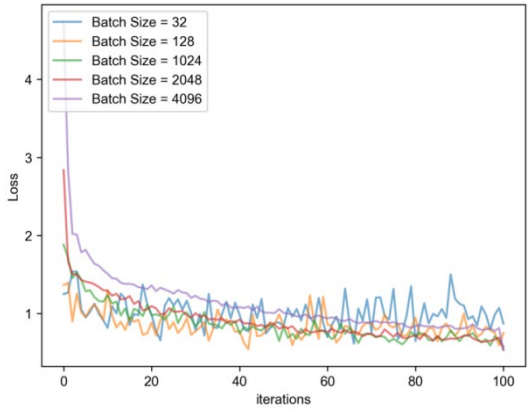

Figure 5: Variation of loss with different batch sizes

When the batch size is relatively small, overfitting is also more likely to occur, as shown in Figure 6. When the Batch Size is 32, the accuracy of the model's validation set is no longer improved after 50 iterations.

ISSN 2616-5775 Vol. 5, Issue 8: 1-11, DOI: 10.25236/AJCIS.2022.050801

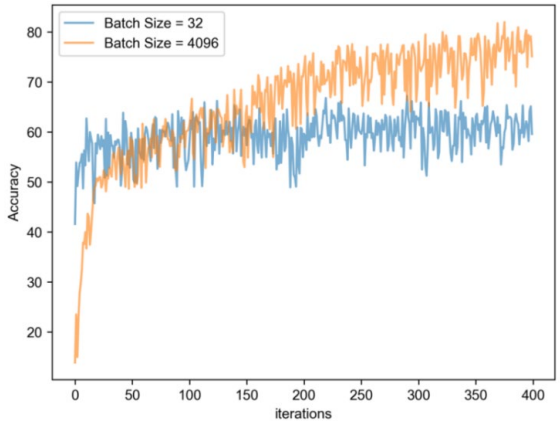

Figure 6: Variation of accuracy of verification set with different batch size

4.1.3 Dropout

As a regularization technique, Dropout can resist overfitting.[4] Figure 7 shows the variation of the accuracy of the model's validation set and test set under different dropout probabilities. Among the parameters of these models, only Epoch is different, where LR is 0.0001 and Batch Size is 128. It can be seen from the figure that, the model fast completed fitting when Dropout is not introduced. However, with the increased number of iterations, the accuracy of the validation set and test set failed to demonstrate a significant upward trend. When the Dropout probability is set to 0.2, it can be seen that the model develops certain ability to resist overfitting. After more than 50 iterations, the accuracy of test set is no longer improved with that of validation set, indicating the model is still trapped in over-fitting. When the Dropout probability is set to 0.5, although slow model training is caused, overfitting is well inhibited, which helps to achieve a model with better generality. However, when the Dropout probability is set to 0.8, the model training is too slow, suggesting a probability of 0.8 is obviously excessively high in this case.

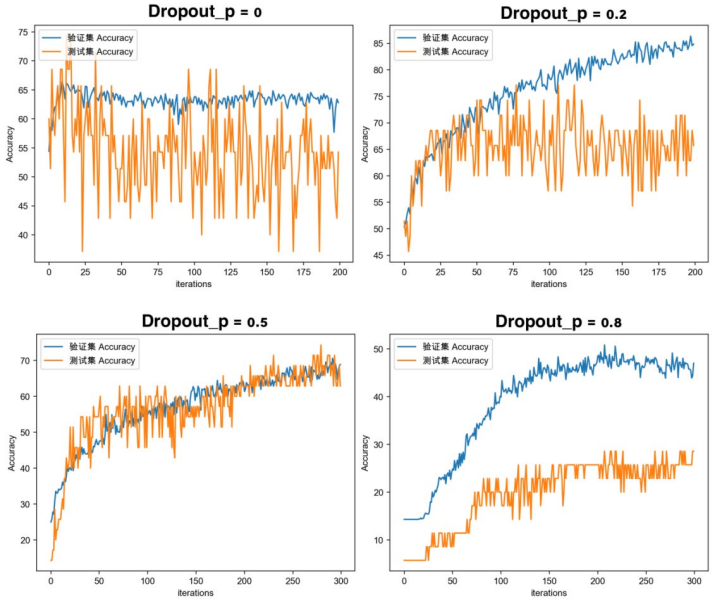

Figure 7: Variation of accuracy with different Dropout probabilities

4.2 Optimization orientation

4.2.1 Improvement of data set quality

The main problems with the data set are too scarce high-quality data and excessively single samples. In despite of quite uneven distribution proportion of rocks, the rocks vary significantly under different

ISSN 2616-5775 Vol. 5, Issue 8: 1-11, DOI: 10.25236/AJCIS.2022.050801

environmental conditions such as lighting and humidity, etc. The data under specific environment is also scarce. If the raw data set can be directly expanded before data enhancement, better effect will be achieved compared with existing data set.

There may be incorrectly labeled data in the dataset, which results in large noise interference in the data set. In this case, the data set should be re-calibrated by professionals. All these factors are one cause that limits the further improvement of accuracy.

In this experiment, the data enhancement methods are relatively few and are mainly on cropping and rotation. Therefore, other image generation techniques such as liquefaction, blurring, sharpening, blooming, etc. may be used to create samples with richer dataset.

4.2.2 Optimization of network structure

The convolutional layer of convolutional neural network can achieve feature extraction of images. For CNN, the more layers, the better the fitting effect. At the same time, it also easily leads to over-fitting, with significantly increased calculation time. Therefore, simplification of the network structure may reduce overfitting. The specific approach is to reduce the number of network layers and neurons to limit the fitting of network.

In addition, multiple models can also be combined, and the most frequent predication of these models is adopted as the output. For instance, a model may be used for large-class classification of images. Then, other models are used to make predication for each sub-class.

4.2.3 Adjustment of training strategies

As to inhibition of overfitting, more regularization methods can be added in network training, including addition of more Dropout layers, batch normalization layers, and use of L1/L2 regularization, etc. Furthermore, the early stopping strategy is also used in the experiment, so that the model training will be stopped at an appropriate time. Reasonable combination of these regularization methods is also an orientation of optimization.

5. Conclusion

The artificial intelligence recognition technology based on deep learning is one of the current research hotspots. In this paper, a study is carried out on the application of convolutional neural network in rock sample image recognition, where the bests test accuracy rate of model reaches 82.86%. In the experiment, regularization techniques such as Dropout, batch normalization, and early stopping are used to effectively inhibit overfitting. This paper also gives discussions on the influence of learning rate, Batch Size and Dropout probability on model training, after which some laws are summarized and the model trainings under different parameters are analyzed. Based on existing result, some orientations of optimization are proposed, including improvement of the data set quality, optimization of network structure, and adjustment of the training strategies.

There are also some deficiencies in the experiment, including:

(1) The upper limit of the accuracy can be hardly further improved: Although the regularization techniques inhibit overfitting, accuracy cannot reach a higher upper limit.

(2) There is lack of rational analysis: a shortcoming of research is that the analysis is not extended to the field of mathematics. Most neural network training strategies in the experiment are based on experience and lack mathematical proofs.

(3) There is lack of in-depth mining of phenomenon: Although some circumstances are analyzed in the training, the analysis remains superficial and not deep enough.

In conclusion, there should be in-depth understanding of artificial intelligence recognition technology in the future study, so as to improve mathematical analysis, and accumulate more experience in model training.

References

[1] Kingma D P, Ba J. Adam. A method for stochastic optimization [J]. arXiv preprint arXiv: 1412.6980, 2014.
[2] Timothy D. Incorporating nesterov momentum into adam [J]. Natural Hazards, 2016, 3(2): 437-453.

ISSN 2616-5775 Vol. 5, Issue 8: 1-11, DOI: 10.25236/AJCIS.2022.050801

[3] Reddi S J, Kale S, Kumar S. On the convergence of adam and beyond [J]. arXiv preprint arXiv: 1904. 09237, 2019.
[4] Srivastava N, Hinton G, Krizhevsky. A et al. Dropout: a simple way to prevent neural networks from overfitting [J]. The journal of machine learning research, 2014, 15(1): 1929-1958.
[5] Li Yan. Rock Image Recognition based on Deep Learning [D]. Beijing Forestry University, 2020.
[6] Cheng G, Guo W. Rock images classification by using deep convolution neural network[C]. //Journal of Physics: Conference Series. IOP Publishing, 2017, 887(1): 012089.