COMP6252 Deep Learning Technologies Coursework

Dong Wang, Student ID: 32798881 dw6u21@soton.ac.uk

I. Introduction

THIS report is related to four networks. The data results are presented in the following sections.

II. Description of The Methods

A. Data Preprocessing

In the data preprocessing process, the data has 10 categories. Firstly, the path address of the image, the type to which the image belongs is stored in the same dataframe. Then the images are encoded in the labels and the data are disrupted. The dataset is sliced according to 70% as training dataset, 20% as validation dataset and 10% as test dataset. The images are reshaped to (180,180). dataset binning data will return tensor type because the DataLoader function only accepts input of tensor type. The DataLoader is then used to specify the batch size of the dataset.

B. Fully Connected Network

The input and output of the fully connected neural network are assigned values, the input is 180 $\times$ 180 $\times$ 4, 4 is the number of channels of the transformed image. The output is all kinds of 10 kinds of images. The architecture of the fully connected neural network is a chain structure, where each layer is a function of the previous layer. Each layer is in turn composed of multiple nodes. The activation functions of this model are compared using sigmoid and relu functions trained at the same number of neurons, respectively. Accuracy will be used for evaluation. The number of neurons is 256 in the first hidden layer and 128 in the second hidden layer. the number of neurons is set at 2^n as much as possible. The optimizer chose adam. because SGD performed poorly in terms of accuracy as shown in the table below. adam showed a significant improvement in accuracy.The activation function of both layers is the Relu function.

C. Convolutional Network

Convolution is a mathematical operation on two functions of real variables. Two types of neuron parameters are used. The activation functions draw on the experience of fully connected network construction. The Relu function is used for all activation functions. The network consists of two convolutional layers, a maximum pooling layer, two convolutional layers, a maximum pooling layer, and two fully connected layers. The convolutional layer has a kernel size of 3x3, stride of 1, and padding of 1. The maximum pooling layer has a kernel size of 3x3, stride of 2, and padding of 1. The input channels of the CNN network will be the four channels of the image, and the output will be 10 to the classification. The maximum pooling layer can perform dimensionality reduction to reduce the computational effort and enhance the invariance of the image. The second one is shown below **??**, changing the maximum pooling layer, and the number of neurons for convolution.

D. Convolutional Network With BN

Batch Normalization is added based on the first CNN model, and what BN does is also to flatten and deflate the input data. The first stage is the Z-Score process, which shifts the mean of the input data to 0 and deflates the variance of the input data to 1. The second stage is the parametric shifting of the mean and parametric deflation of the variance of the data on this basis.

E. Convolutional Network Using RMSprop

Replacing the optimizer with RMSprop, SGD will update the parametric model according to the gradient of the current sample. The RMSprop formula is 1 η is the learning rate, g_t is the gradient, and ϵ is the hyperparameter, in order to keep the denominator from being zero. The hyperparameter α is used to do a weighted average of G_{t-1} and g_t^2, and δ_t for the gradient iterations is controlled by parameter B. Only when all the gradient values on all weights are relatively close, the model can have better learning ability and RMSprop adds hyperparameters for a better optimization algorithm. By using adaptive learning rate, RMSprop can better control the magnitude and direction of parameter updates, and is more stable and reliable.

$$\begin{aligned} G_t &= \alpha G_{t-1} + (1-\alpha) g_t^2 \\ \Delta_t &= \mathrm{B}\Delta_{t-1} - \frac{\eta}{\sqrt{G_t+\epsilon}} * g_t \\ w_t &= w_{t-1} + \Delta_t \end{aligned} \tag{1}$$

III. Evalution

According to the experimental data obtained from the table, by training and evaluating the model, the accuracy and loss reduction of relu is much better than the sigmoid activation function. Figure 2 is the accuracy curve of the training set. Figure 3 is the accuracy curve of the test set. The orange and blue colors in the figure are the results of using the relu function for 50 and 100 iterations, respectively, and the curve with lower accuracy this one uses the sigmoid function. In the model using Adam optimizer, the accuracy of Train set is high and the accuracy of test and validation set is overfitted.

the network with Sigmoid activation function is less effective. As shown in the Figure 1.

Network Model	Loss	Train Accuracy	Test Accuracy	Validation Accuracy
FC_RELU_ADAM_50	1.3514	0. 8125	0. 5199	0. 4849
FC_SIGMOID_SGD_50	2.2968	0. 1301	0. 0899	0. 0999
FC_RELU_ADAM_100	0.8629	0. 9971	0. 5099	0. 4599
FC_SIGMOID_SGD_100	2. 2925	0. 1888	0. 1499	0. 1649

Fig. 1. Evaluation Table one

The sigmoid causes the gradient to vanish. Neural network in the process of backpropagation, the gradient calculation of each parameter layer will involve the value of the derivative of the activation function. As the number of layers of the neural network continues to increase and the number of activation functions continues to increase, the number of activation function derivatives that need to be multiplied in the process of calculating the gradient for the first layer of parameters increases, while the number of activation function derivatives involved in the gradient calculation for the subsequent layers of parameters decreases step by step. When the gradient passes through multiple sigmoid functions, the derivative of each sigmoid function is less than 1, which causes the gradient value to gradually shrink and eventually disappear to 0. As shown in the table, the loss of the sigmoid function remains almost unchanged after 50 and 100 rounds of iterations. Figure 4 also shows that the loss can hardly decrease using the sigmoid activation function.

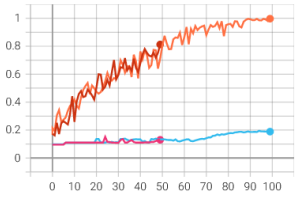

Fig. 2. Accuracy of two FCN

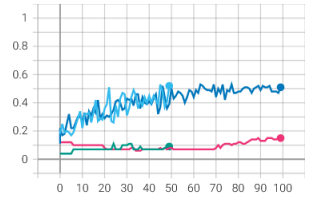

Fig. 3. Accuracy of test dataset

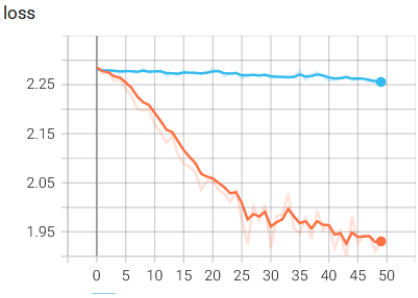

Fig. 4. Loss

Figure 6 and Figure 7 show, There is almost no difference between the two models for the number of neurons. The accuracy of both the validation set and the test set is around 0.5.With the addition of Normalization layer, there is a significant improvement in the accuracy rate. By normalization, the neural network can converge faster because the inputs of each layer are tuned to have a similar range and distribution. The

Network Model	Loss	Train Accuracy	Test Accuracy	Validation Accuracy
CNN_ADAM_50	0. 5721	1.0	0. 5299	0. 5450
CNN_teach_50	0. 3645	1.0	0. 560	0. 524
CNN_ADAM_100	0. 2747	1.0	0. 5699	0. 550
CNN_teach_100	0. 2033	1.0	0. 5999	0. 51999
BN_50	0. 1870	1.0	0. 680	0. 610
BN_100	0.0950	1. 0	0. 6599	0. 610
RMSPROP_50	0. 7924	1. 0	0. 620	0. 5949
RMSPROP_100	0. 3649	1. 0	0. 6499	0. 5949

Fig. 5. Evaluation CNN

use of RMSprop causes a decrease in accuracy. In the figure we can also see by the fluctuation of accuracy that the loss of RMSprop may suddenly become larger. This proves that the model may have experienced a gradient descent in the wrong direction. As the figure 8 and 9 below, The highest accuracy is the Batch normalization network and the lowest accuracy is the RMSprop network. The network using However, all models in CNN networks suffer from some overfitting.

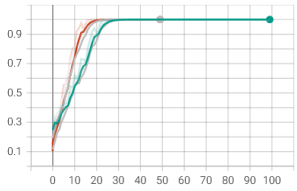

Fig. 6. CNN different models

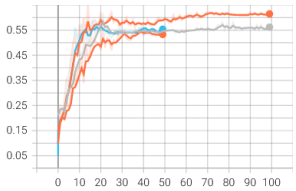

Fig. 7. CNN Accuracy test dataset

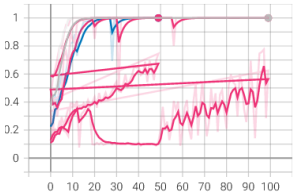

Fig. 8. BN and RMSprop train

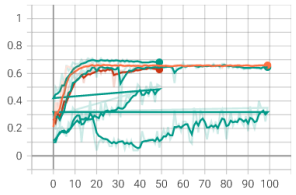

Fig. 9. BN and RMSprop test

IV. Conclusion

Based on the results of the experiments, the sigmoid function needs to be used with consideration of whether it causes gradient disappearance, especially after the number of training increases, and the depth and number of layers of the network can be increased in future work, because deeper models generalize better in task heavy [1].

References

[1] Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007b). Greedy layer-wise training of deep networks. In B. Schölkopf, J. Platt, and T. Hoffman, editors, Advances in Neural Information Processing Systems 19 (NIPS'06), pages 153–160. MIT Press. 173