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 and the image data is stored in 10 different folders, 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. Wrap the data, using Dataset in torch to wrap the data and labels, and vary the images. The images are transformed according 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. at this point the data is ready for training with the model.

B. Fully Connected Network

The input and output of the fully connected neural network are fixed 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.

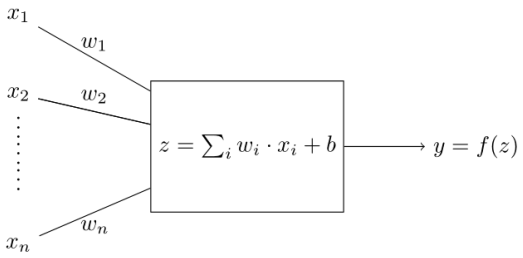

Fig. 1. function

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 512 in the first hidden layer and 256 in the second hidden layer. the number of neurons is set at 2^n as much as possible. The results of the network are

```
=========
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================
=========
Net                                      --                        --
├─Linear: 1-1                            [64, 512]                 66,355,712
├─ReLU: 1-2                              [64, 512]                 --
├─Linear: 1-3                            [64, 256]                 131,328
├─ReLU: 1-4                              [64, 256]                 --
├─Linear: 1-5                            [64, 10]                  2,570
==========================================================================
=========
Total params: 66,489,610
Trainable params: 66,489,610
Non-trainable params: 0
Total mult-adds (G): 4.26
```

Fig. 2. Structure of FCN

shown in the figure 2
The activation function of both layers is the Relu function. the ReLU function is characterized by an output equal to input x when input x is greater than 0 and an output of 0 when input x is less than or equal to 0. Therefore, the image of the ReLU function is a function consisting of a straight line with slope 1 and a horizontal line from the origin. the Sigmoid function is characterized by an output value close to 0 or 1 when input x is large or small, and an output value about 0.5 when input is close to 0. Therefore, the image of the Sigmoid function is an S-shaped curve taking the maximum value of 1 at the origin. The image of the Sigmoid function is therefore an S-shaped curve that takes the maximum value of 1 at the origin.

C. Convolutional Network

Convolution is a mathematical operation on two functions of real variables. Two types of neuron parameters are used. The first one, as shown in the figure 3 below.

```
=========
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================
=========
Net2                                     --                        --
├─Conv2d: 1-1                            [64, 64, 180, 180]        2,368
├─ReLU: 1-2                              [64, 64, 180, 180]        --
├─Conv2d: 1-3                            [64, 128, 180, 180]       73,856
├─ReLU: 1-4                              [64, 128, 180, 180]       --
├─MaxPool2d: 1-5                         [64, 128, 90, 90]         --
├─Conv2d: 1-6                            [64, 64, 90, 90]          73,792
├─ReLU: 1-7                              [64, 64, 90, 90]          --
├─Conv2d: 1-8                            [64, 32, 90, 90]          18,464
├─ReLU: 1-9                              [64, 32, 90, 90]          --
├─MaxPool2d: 1-10                        [64, 32, 45, 45]          --
├─Linear: 1-11                           [64, 256]                 16,589,056
├─Linear: 1-12                           [64, 10]                  2,570
==========================================================================
=========
Total params: 16,760,106
Trainable params: 16,760,106
Non-trainable params: 0
Total mult-adds (G): 206.95
```

Fig. 3. Structure of first CNN model

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 is able to pick the largest value in the kernel area for output. 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 4, changing the maximum pooling layer, and the number of neurons for convolution.

```
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
Net2_teach                               --                        --
├─Conv2d: 1-1                            [64, 32, 178, 178]        1,184
├─ReLU: 1-2                              [64, 32, 178, 178]        --
├─Conv2d: 1-3                            [64, 32, 176, 176]        9,248
├─ReLU: 1-4                              [64, 32, 176, 176]        --
├─MaxPool2d: 1-5                         [64, 32, 88, 88]          --
├─Conv2d: 1-6                            [64, 64, 86, 86]          18,496
├─ReLU: 1-7                              [64, 64, 86, 86]          --
├─Conv2d: 1-8                            [64, 64, 84, 84]          36,928
├─ReLU: 1-9                              [64, 64, 84, 84]          --
├─MaxPool2d: 1-10                        [64, 64, 42, 42]          --
├─Linear: 1-11                           [64, 256]                 28,901,632
├─Linear: 1-12                           [64, 10]                  2,570
==========================================================================================
Total params: 28,970,058
Trainable params: 28,970,058
Non-trainable params: 0
Total mult-adds (G): 48.02
```

Fig. 4. Structure of second CNN model

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 the overall loss can be optimized to a better local minimum, 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. When dealing with non-smooth objective functions, RMSprop can accelerate the convergence speed and improve the training effect.

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

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 5 is the accuracy curve of the training set. Figure 6 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.

Model	Loss	Accuracy	Epochs	Acc on test
Fc_RELU	2.0630	0.2989	50	0.2099
Fc_RELU	1.9417	0.2131	100	0.2399
Fc_Sigmoid	2.2987	0.3619	50	0.0299
Fc_Sigmoid	2.2870	0.2017	100	0.2299

The sigmoid causes the gradient to vanish. Neural network in the process of back propagation, the gradient calculation of each parameter layer will involve the value of the derivative of the activation function. For example, a three-layer $\hat{y} = F(F(X * w_1) * w_2) * w_3$. 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. The gradients of the parameters in different layers vary greatly in the calculation process, and this difference is a kind of cumulative effect, and this cumulative effect will lead to a part of the gradient of the linear layer parameters is too large and another part is too small, thus affecting the smooth training of the model. 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 7 also shows that the loss can hardly decrease using the sigmoid activation function.

Model	Loss	Accuracy	Epochs	Acc on test
CNN_MODEL1	2.2720	0.2689	50	0.0599
CNN_MODEL1	2.0717	0.4363	100	0.4099
CNN_MODEL2	2.2650	0.3619	50	0.3299
CNN_MODEL2	1.9410	0.4105	100	0.3999
CNN_Norm	1.2908	0.8226	50	0.6800
CNN_Norm	0.8245	0.9942	100	0.6899

Figure 8 and Figure 9 show, There is almost no difference between the two models for the number of neurons. With the addition of Normalization layer, there is a significant improvement in the accuracy rate. The reasons for the improvement are the following: 1. The current network structure is simple and does not handle matrices that are not Normalized well. There is too much interference. 2. By normalization, the neural network can converge faster because the inputs of each layer are tuned to have a similar range and distribution. In addition, Batch Normalization has some regularization effect because it reduces the covariate bias in the neural network. The application of the model

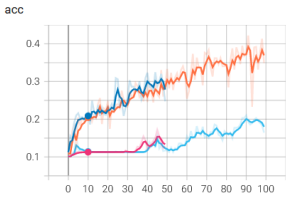

Fig. 5. Accuracy of two FCN

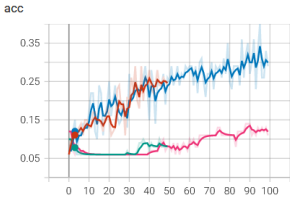

Fig. 6. Accuracy of test dataset

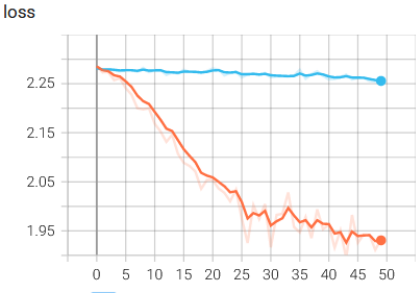

Fig. 7. Loss

and data is clearly not suitable for the use of RMSprop. During the training process, the loss of the model appears to be repeated and increased. This proves that the model may have experienced a gradient descent in the wrong direction. As the figure 10 and 11 below, The highest accuracy is the Batch normalization network and the lowest accuracy is the RMSprop network. The network using RMSprop only obtained 0.15 at 50 rounds of training and 0.22 at 100 rounds of training on the test dataset.

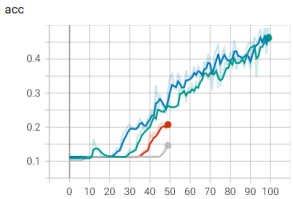

Fig. 8. CNN different models

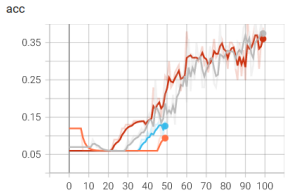

Fig. 9. CNN Accuracy test dataset

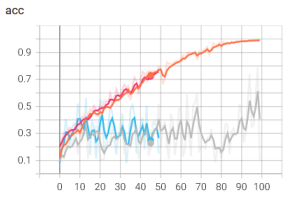

Fig. 10. BN and RMSprop train

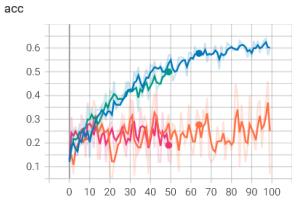

Fig. 11. BN and RMSprop test

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