COMP551 Mini project 3

Chaoyi Liu, Sandra Deng, Zheyu Liu

April 2020

1 Abstract

Image classification refers to classify images to appropriate category or the probability that the input image is a particular category. Multilayer perceptron(MLP) is artificial neural network which consists of at least three layers: an input layer, a hidden layer and an output layer. The purpose of convolution neural network(CNN) is to make the image input easier to be processed without losing features. In this project, we used both MLP and CNN for multiple class image classification. By keeping track of the cross entropy loss and the accuracy, we found that CNN had a better accuracy compared with MLP.

2 Introduction

In this project, we investigated the Cifar-10, a dataset which consists of 60000 32×32 colour images in 10 classes, with 6000 images per class. We implemented MLP and CNN for this multiclass dataset. For multi layer perceptron, we changed the number of hidden layers and the number of neurons in each hidden layer. We also tried sigmoid, relu, and leaky relu as the activation function of MLP. Then we tried gradient descent, stochastic gradient descent(SGD), and SGD with momentum of the back propogation of the MLP. For convolution neural network, we investigated the effect of the learning rate of SGD, the number of epoch and the activation function on the train and test accuracy.

3 Dataset

The CIFAR-10 dataset consists of 60,000 color images in 10 classes, where 50,000 are training images and 10,000 are test images. The training images are divided into 5 equal-size batches. The test batch contains 1000 randomly selected images for each class. Generally, the baseline test error is between 15% and 18% for convolutional neural network. [2]

Since the data downloaded is ndarray, data preprocessing is required in CNN. In CNN, we first convert the ndarray into tensor object, so we can use functions from pytorch library. We then normalize the data making all the features having 0.5 mean and 0.5 standard deviation. The normalization makes the data easier to learn.

4 Result

4.1 MLP

4.1.1 GD to SGD

We first implemented a two layer MLP(one input layer, one hidden layer and one output layer) by using sigmoid as the activation function, as well as gradient descent to update the weights and the bias. We found that the training time is extremely long (more than 6 hours), and the accuracy rate is 12.94%. As shown in figure 1, the cross entropy loss converges to around 2 as number of epochs increases to 5000.

To reduce the time needed for the training, we replaced gradient descent with stochastic gradient descent, and set the batch size to 200. The training time was reduced significantly to 136 seconds. As shown in figure

2, as number of epochs increased, the cross entropy loss is still higher than 2. The training accuracy rate was 12.67%, and the testing accuracy was 10.10%, which were still very low.

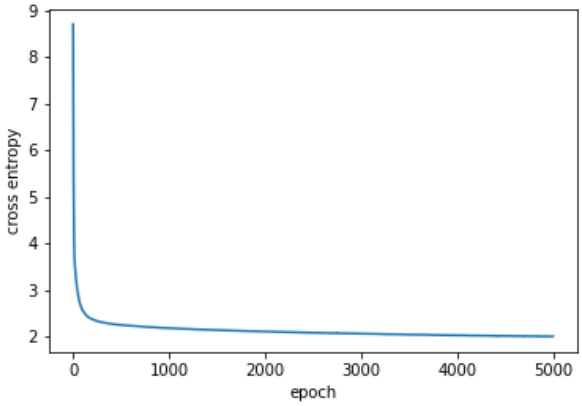


Figure 1: Cross entropy of 1-hidden-layer MLP with sigmoid and gradient descent

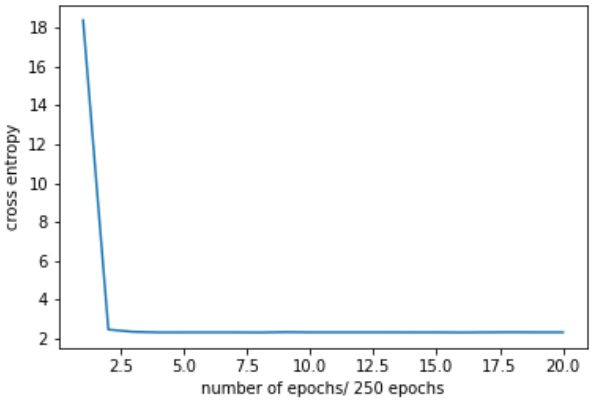


Figure 2: Cross entropy of 1-hidden-layer MLP with sigmoid and SGD

4.1.2 Activation function

From the above experiment, we found that the accuracy for both training and testing were low, which may due to the gradient vanishing in sigmoid function as the logit increased. Therefore, we then tried to replace the activation function of the hidden layer from sigmoid to ReLU and leaky ReLU. As shown in the figure 3, the cross entropy loss using ReLU as the activation function decreased to around 1.25. The accuracy rate of the training set was 46.63%, and the accuracy rate of the testing set was 44.54%. As shown in figure 4, the cross entropy loss using leaky ReLU as the activation function was also reduced compared to the model using sigmoid. The accuracy rate of the model using leaky ReLU had a higher accuracy on both training set (48.40%) and testing set (45.26%) compared with the model using sigmoid as the activation function.

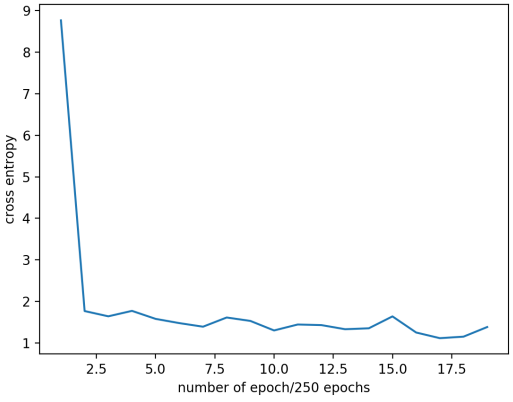


Figure 3: Cross entropy of 1-hidden-layer MLP with ReLU and SGD

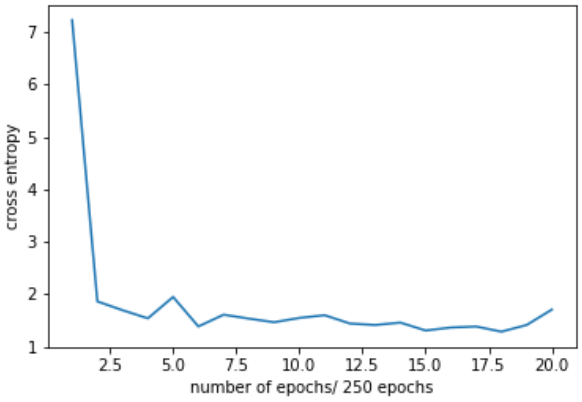


Figure 4: Cross entropy of 1-hidden-layer MLP with leaky ReLU and SGD

4.1.3 Momentum

As shown in figure 3 and figure 4, the cross entropy curve was not very smooth, so we add momentum into the stochastic gradient descent in our model. Figure 5 showed the cross entropy loss of the model using SGD with momentum, beta equals to 0.9. The activation function was ReLU, and other hyperparameters were the same as the model which generates the figure 3. Compared with figure 3, the curve was smoother, but the training accuracy and the testing accuracy was higher than the model using SGD (training accuracy was 47.54%, testing accuracy was 45.05%).

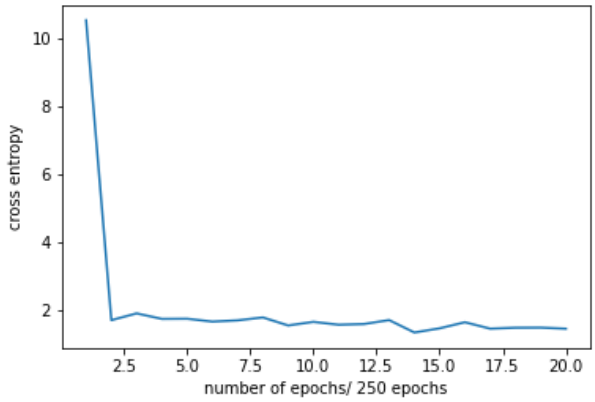


Figure 5: Cross entropy of 1-hidden-layer MLP with RELU and SGD(with momentum)

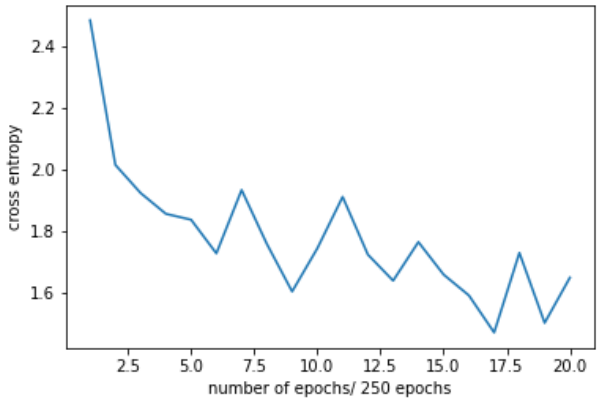


Figure 6: Cross entropy of 2-hidden-layer MLP with RELU and SGD(with momentum)

4.1.4 Number of hidden layers

To further improve the model, we tried to increase the hidden layers. Figure 6 shows the cross entropy loss of the model using 2 hidden layers. This 2-hidden-layer MLP used ReLU as the activation function, and SGD with momentum was also applied to this model. The training accuracy was 42.85%, and the testing accuracy was 42.58%. Compared with 1-hidden-layer MLP, the 2-hidden-layer MLP had lower training accuracy and testing accuracy.

	Training accuracy (%)	Test accuracy (%)
sigmoid	12.67	10.10
1 hidden layer ReLU	46.63	44.54
1 hidden layer ReLU + momentum	47.54	45.05
2 hidden layer ReLU + momentum	42.85	42.58
1 hidden layer leaky ReLU	48.40	45.26
1 hidden layer leaky ReLU + momentum	47.05	44.50

Table 1: Training and test accuracy rate for different SGD configurations

4.1.5 Number of units

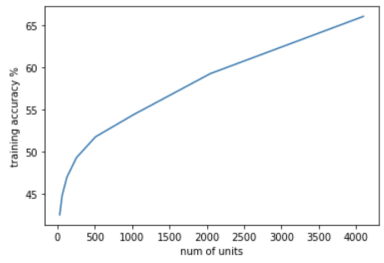
As we can see in table 1, using ReLU and leaky ReLU yields better performance. Thus, we tuned the number of units per layer based on 3 different configurations: 1-hidden-layer leaky ReLU, 1-hidden-layer ReLU and 2-hidden-layer ReLU. We selected 8 values as the number of units: 32, 64, 128, 256, 512, 1024, 2048 and 4096.

As in fig 7, the training accuracy increases as the number of units grows. However, this is not always the case for test accuracy. The increasing speed of test accuracy is extremely slow, even be negative, when

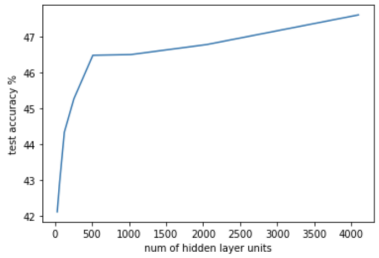
	1-hidden-layer leaky ReLU	1-hidden-layer ReLU	2-hidden-layer ReLU
best # of units (M)	4096	1024	2048
training accuracy for best M (%)	66.04	54.70	63.04
test accuracy for best M (%)	47.62	47.46	48.81

Table 2: Best unit value and the accuracy accordingly for 3 configurations

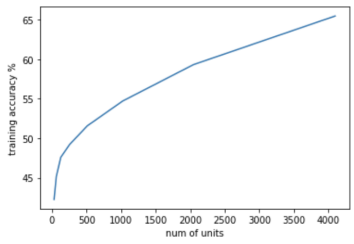
number of units exceeds certain value. Table 2 lists the best unit number value for each configuration. We can see that 2-hidden-layer ReLU gives the best test accuracy 48.81% using 2048 units per hidden layer.



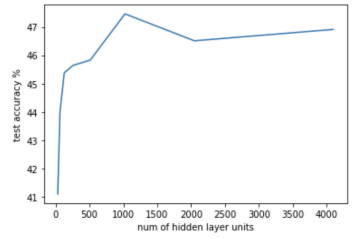
(a) training accuracy vs. num of units for 1 hidden layer leaky ReLU



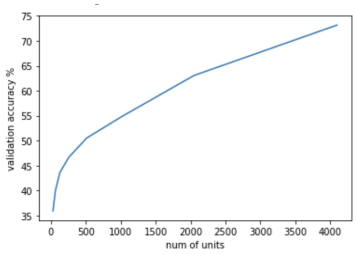
(b) test accuracy vs. num of units for 1 hidden layer leaky ReLU



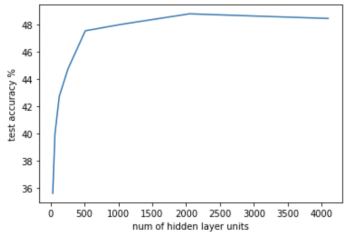
(c) training accuracy vs. num of units for 1 hidden layer ReLU



(d) test accuracy vs. num of units for 1 hidden layer ReLU



(e) training accuracy vs. num of units for 2 hidden layer ReLU



(f) test accuracy vs. num of units for 2 hidden layer ReLU

Figure 7: accuracy w.r.t number of units per layer for 3 configurations

4.2 CNN

For the CNN approach, we used the code from the pytorch website[1]. In general, we found that the CNN gave better result than the MLP. We did experiment on the learnign rate, the number of epoch and different choices of activation function. The best training accuracy was 68% and the test accuracy was 63%.

4.2.1 Learning Rate

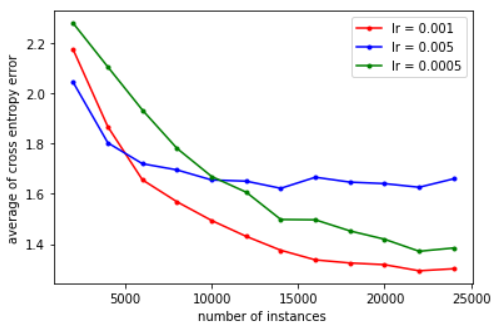
We did experiment with learning rate 0.005, 0.001 and 0.0005. The result is shown in figure 8a. We can see that when the learning rate is 0.005, the error stopped at around 1.5 and could not go lower as the number of training data increased. This suggests that the learning rate is too large and it overshot. When the learning rate is 0.0005, the error at 24000 is higher than that with learning rate 0.001. This suggests that the learning rate is too small and it takes longer for the model to reach the minimum. The line of learning rate 0.001 is stable at the end, meaning that it reaches the minimum. Overall, the learning rate 0.001 gives the best result in terms of the time reaching the minimum without overshoot.

4.2.2 Number of Epoch

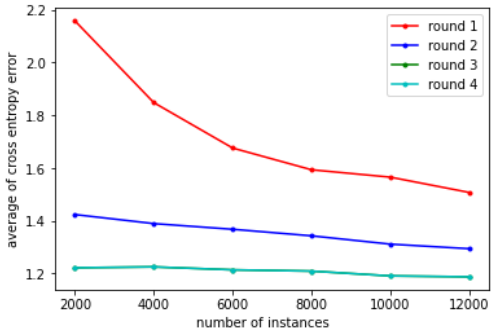
We then determined the epoch to train the model. The error decreases in round 1, 2 and 3. However round 4 overlaps with the round 3 in figure 8b, so we can know that we cannot reduce the training error to a great extent. Epoch equals to 3 giving the best result in terms of training time and accuracy.

4.2.3 Activation Function

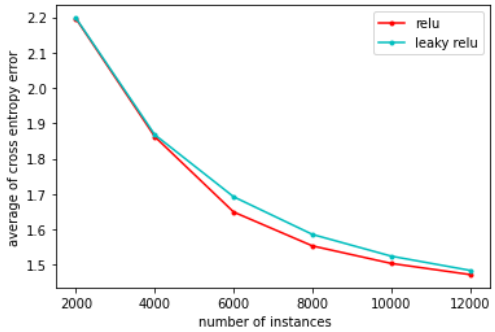
As in the MLP, we also tried different activation function in CNN. We tried relu and leaky relu. From figure 8c, the error in relu decreases faster than the leaky relu while the leaky relu has a smoother shape. However, they reached to a similar error at the end of 1 epoch. Therefore, both activation functions will give similar result at the end of epoch 3. The reason for we did not try sigmoid in CNN is that the gradient vanished as the logit increasing.



(a) CNN learning rate experiment



(b) CNN epoch experiment



(c) CNN activation function experiment

Figure 8: Tuning CNN

5 Conclusion

For MLP, we tried/tuned: 1) activation functions, 2) with or without momentum, 3) number of hidden layers, 4) number of units per hidden layer. Among all our experiments, 2-hidden-layer ReLU with momentum using 2048 units per hidden layer gives the best performance. For the CNN ,we did experiments on the learning rate of SGD, activation function and the number of epoch. We found that the best result was obtained when the learning rate is 0.001, activation function is ReLU and the epoch is 3. Based on the experiments we did, we found that both the training accuracy and the testing accuracy of CNN was higher than MLP.

References

[1] Training a classifier. https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html. Accessed on: April 17, 2020.

[2] Alex Krizhevsky. Learning multiple layers of features from tiny images, 2009.