ARTIFICIAL NEURAL NETWORK
HOMEWORK 2 REPORT

Jiahao Li
Department of Computer Science
Tsinghua University
lijiahao17@mails.tsinghua.edu.cn

1 Implementation Details

In this experiment, a multi-layer perceptron (MLP) and a convolutional neural network (CNN) are constructed for classification task on Cifar-10 dataset. The detailed architectures of MLP and CNN are shown in Table 1 and Table 2, respectively.

Layer	Type	In	Out
1	Linear	3072	256
2	BN	256	256
3	ReLU	256	256
4	Dropout	256	256
5	Linear	256	10

Table 1: Architecture of MLP model

Layer	Type	In	Out
1	Conv3x3	$32 \times 32 \times 3$	$30 \times 30 \times 32$
2	BN	$30 \times 30 \times 32$	$30 \times 30 \times 32$
3	ReLU	$30 \times 30 \times 32$	$30 \times 30 \times 32$
4	Dropout	$30 \times 30 \times 32$	$30 \times 30 \times 32$
5	MaxPool2x2	$30 \times 30 \times 32$	$15 \times 15 \times 32$
6	Conv3x3	$15 \times 15 \times 32$	$13 \times 13 \times 256$
7	BN	$13 \times 13 \times 256$	$13 \times 13 \times 256$
8	ReLU	$13 \times 13 \times 256$	$13 \times 13 \times 256$
9	Dropout	$13 \times 13 \times 256$	$13 \times 13 \times 256$
10	MaxPool2x2	$13 \times 13 \times 256$	$6 \times 6 \times 256$
11	Flatten	$6 \times 6 \times 256$	9216
12	Linear	9216	10

Table 2: Architecture of CNN model

Unless otherwise specified, both MLP and CNN are trained by Adam optimizer over 50 epochs with batch size of 100 and learning rate of 0.001. The results of the best epoch, which has the maximum validation accuracy, are reported as the final results.

2 Experiments on Cifar-10 Dataset

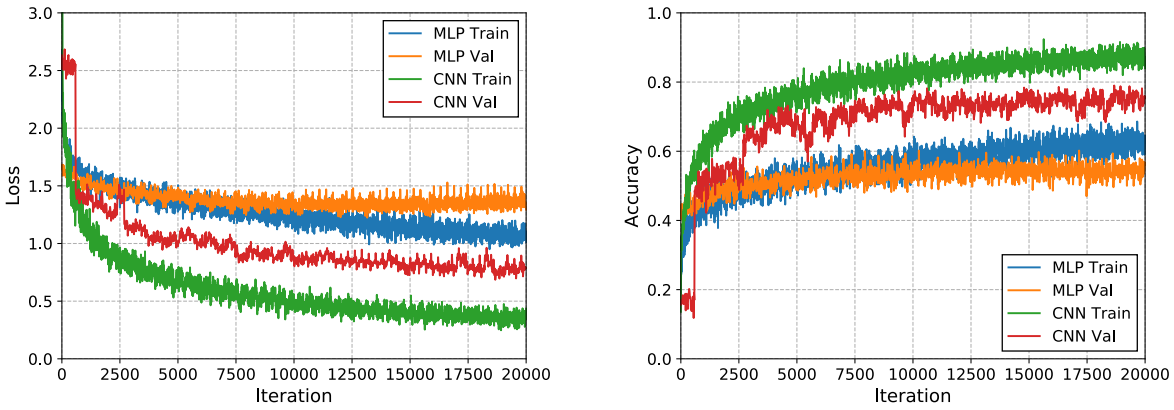
2.1 Argument Filling

For the training process, I fill the arguments of `model.forward` with `is_train=True` and `reuse=False`, while for the validation process, the arguments are filled with `is_train=False` and `reuse=True`.

The `reuse` flag is set for the validation and testing process to reuse the model parameters, which is already trained in training process. The `is_train` flag should be specified because the batch normalization layers and dropout layers behave differently in training and testing process. Specifically, for the batch normalization layer, the parameters μ and σ denote the mean value and standard deviation of the mini-batch in training process, respectively. These parameters of the entire training population are maintained during training, and reused in testing phase. For the dropout layer, it will randomly drop out its output nodes with probability p and scale the remaining nodes by $1/(1-p)$ in training time, while it will do nothing during testing.

2.2 Loss and Accuracy

The loss and accuracy graphs are shown in Figure 1. To display the results in pretty style, the loss and accuracy results are downsampled by 5 times before they are plotted in the following graphs.



(a) Loss against Iteration

(b) Accuracy against Iteration

Figure 1: Loss and accuracy against iteration

2.3 Experimental Results of MLP and CNN

The experimental results of MLP and CNN are shown in Table 3. From Figure 1 and Table 3, it can be observed that CNN achieves a superior performance over MLP, *i.e.* a higher accuracy and a lower loss in both train-set and test-set.

Model	Train Loss	Train Acc (%)	Val Loss	Val Acc (%)	Test Loss	Test Acc (%)
MLP	1.15	59.95	1.35	54.85	1.33	54.69
CNN	**0.36**	**87.00**	**0.76**	**75.98**	**0.76**	**75.66**

Table 3: Experimental results of MLP and CNN model on Cifar-10 dataset.

2.4 Ablation Study of Batch Normalization

The experimental results of both MLP and CNN with and without BN layers are shown in Table 4. Since the only variable of this experiment is the BN layers, it can be concluded that the batch normalization layer speeds up the convergence and improves the robustness, because it effectively prevents the vanishing gradients.

Model	Train Loss	Train Acc (%)	Val Loss	Val Acc (%)	Test Loss	Test Acc(%)
MLP without BN	**1.08**	**62.00**	1.42	53.86	1.40	53.58
MLP with BN	1.15	59.95	**1.35**	**54.85**	**1.33**	**54.69**
CNN without BN	0.43	84.81	0.80	74.95	0.81	73.93
CNN with BN	**0.36**	**87.00**	**0.76**	**75.98**	**0.76**	**75.66**

Table 4: Experimental results of MLP and CNN with and without BN layers

2.5 Dropout Rate Tuning

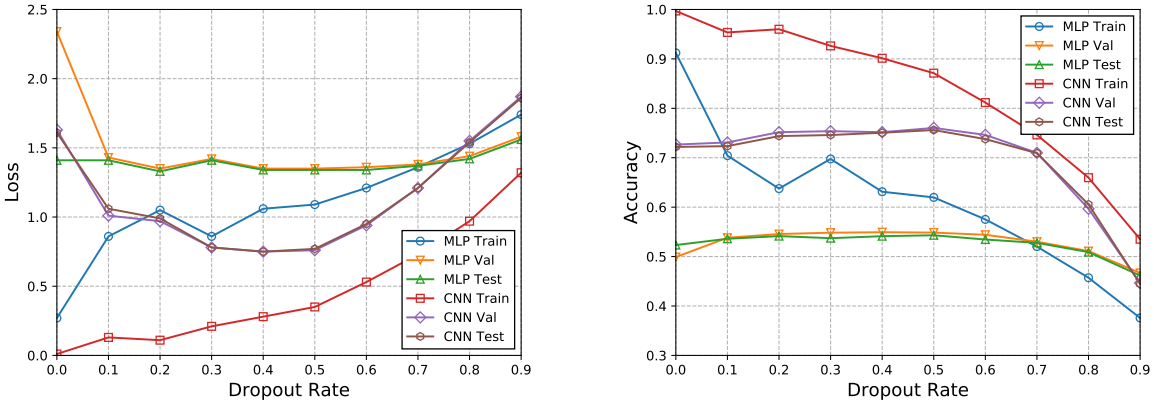
The dropout rate is adjusted from 0.0 to 0.9 by step of 0.1. The experimental results of both models are shown in Table 5 and Table 6. The final results with different dropout rates are visualized in Figure 2. Obviously, with a low dropout rate, the model will easily overfit on the train-set and perform poorly on the test-set. However, an excessively high dropout rate is also harmful to the model, because it damages the balance of the layers. A medium dropout rate, 0.5 for example, is beneficial to prevent overfitting and enhance the performance.

Dropout Rate	Train Loss	Train Acc (%)	Val Loss	Val Acc (%)	Test Loss	Test Acc(%)
0.0	**0.27**	**91.21**	2.34	49.90	1.41	52.33
0.1	0.86	70.42	1.43	53.81	1.41	53.60
0.2	1.05	63.75	**1.35**	54.56	**1.33**	54.14
0.3	0.86	69.75	1.42	54.83	1.41	53.72
0.4	1.06	63.12	**1.35**	**54.93**	1.34	54.12
0.5	1.09	61.97	**1.35**	54.85	1.34	**54.31**
0.6	1.21	57.52	1.36	54.41	1.34	53.45
0.7	1.36	52.02	1.38	53.01	1.37	52.77
0.8	1.53	45.71	1.44	51.11	1.42	50.90
0.9	1.74	37.58	1.58	46.69	1.56	46.15

Table 5: Experimental results of MLP with different dropout rates.

Dropout Rate	Train Loss	Train Acc (%)	Val Loss	Val Acc (%)	Test Loss	Test Acc(%)
0.0	**0.01**	**99.69**	1.63	72.66	1.61	72.18
0.1	0.13	95.35	1.01	73.09	1.06	72.34
0.2	0.11	96.00	0.97	75.17	0.99	74.38
0.3	0.21	92.63	0.78	75.37	0.78	74.59
0.4	0.28	90.13	**0.75**	75.17	**0.75**	75.06
0.5	0.35	87.11	0.76	**76.03**	0.77	**75.60**
0.6	0.53	81.16	0.94	74.62	0.95	73.77
0.7	0.73	74.62	1.21	70.99	1.21	70.92
0.8	0.97	65.96	1.55	59.66	1.54	60.56
0.9	1.32	53.51	1.87	44.65	1.86	44.45

Table 6: Experimental results of CNN with different dropout rates.



(a) Loss with different dropout rates

(b) Accuracy with different dropout rates

Figure 2: Loss and accuracy with different dropout rates

2.6 Difference between Training and Testing Loss

In most cases, the training loss is a little lower than the testing loss, because the model is usually over-parameterized and tends to overfit on train-set. Many methods have been proposed to tackle the issue of overfitting, including the reduction of total parameters, dropout layers, batch normalization, L2 regularization, data augmentation, early stopping, etc. In this experiment, the dropout layers and batch normalization layers are applied to avoid overfitting.