Report for Artificial Neural Network HW3

Jiayi Weng
Department of Computer Science
Tsinghua University
wengjy16@mails.tsinghua.edu.cn

1 Network Architecture and Hyperparameter Setting

According to the provided code and guideline, I keep the other hyperparameters the same as before. I only tune the hidden layer size of network, whether it has BN layer or not, and the drop rate.

The interface of MLP is different from CNN. It's "drop_rate" in CNN, but "keep_prob" in MLP. From my perspective, it should be a typo due to different interpretation (drop_rate + keep_prob = 1), so I modify the "main.py" in MLP with changing "keep_prob" into "drop_rate".

1.1 MLP

It's the default setting provided by original code. Detailed network architecture is in Table 1.

Table 1: MLP Default Network Architecture

Layer	Type	Input Size	Output Size	Init	Others
1	Linear	784	50	Xavier	
2	BN	50	50		
3	ReLU	50	50		
4	Dropout	50	50		$p = 0.5$
5	Linear	50	10	Xavier	

1.2 CNN

It's the default setting provided by original code. Detailed network architecture is in Table 2.

Table 2: CNN Default Network Architecture (the size is presented as "Channel $\times$ Height $\times$ Width")

Layer	Type	Input Size	Output Size	Kernel	Pad	Init	Others
1	Conv2D	$1 \times 28 \times 28$	$256 \times 28 \times 28$	3×3	1	Xavier	
2	BN	$256 \times 28 \times 28$	$256 \times 28 \times 28$				
3	ReLU	$256 \times 28 \times 28$	$256 \times 28 \times 28$				
4	Dropout	$256 \times 28 \times 28$	$256 \times 28 \times 28$				$p = 0.5$
5	MaxPool2D	$256 \times 28 \times 28$	$256 \times 14 \times 14$	2×2	0		
6	Conv2D	$256 \times 14 \times 14$	$128 \times 14 \times 14$	3×3	1	Xavier	
7	BN	$128 \times 14 \times 14$	$128 \times 14 \times 14$				
8	ReLU	$128 \times 14 \times 14$	$128 \times 14 \times 14$				
9	Dropout	$128 \times 14 \times 14$	$128 \times 14 \times 14$				$p = 0.5$
10	MaxPool2D	$128 \times 14 \times 14$	$128 \times 7 \times 7$	2×2	0		
11	Linear	6272	10			Xavier	

2 Experiment & Question Answering

2.1 Experiment Setting

All of these experiments are conducted on a Linux server. The CPU infomation is "Intel(R) Xeon(R) CPU E5-2683 v3 @ 2.00GHz", with 4 NVIDIA GeForce GTX 1080 Ti.

2.2 Question 1

I write the argument in the following way: the first line is for training process, thus, I fill it with "is_train=True, reuse=False"; for the second line, it is for the validation process, and the parameter should be the same as above. So its argument is "is_train=False, reuse=True" in order to reuse the parameter used in training process.

2.3 Question 2

Both training and validation loss against each iteration during training is shown in Figure 1.

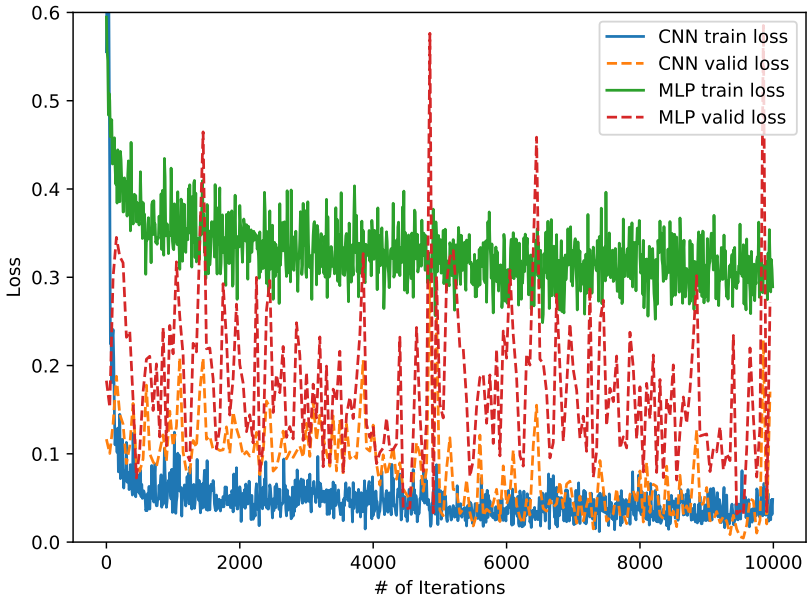


Figure 1: Training and validation loss for each iteration.

2.4 Question 3

The experiment result is shown in Figure 1. From Figure 1, CNN converges better than MLP. More specifically, in MLP experiment, the validation loss is lower than training, but it is less stable. I think it is caused by the "Dropout" layer and the overparameterize characteristic of MLP. The training loss of CNN is lower than its validation loss, but the gap isn't large.

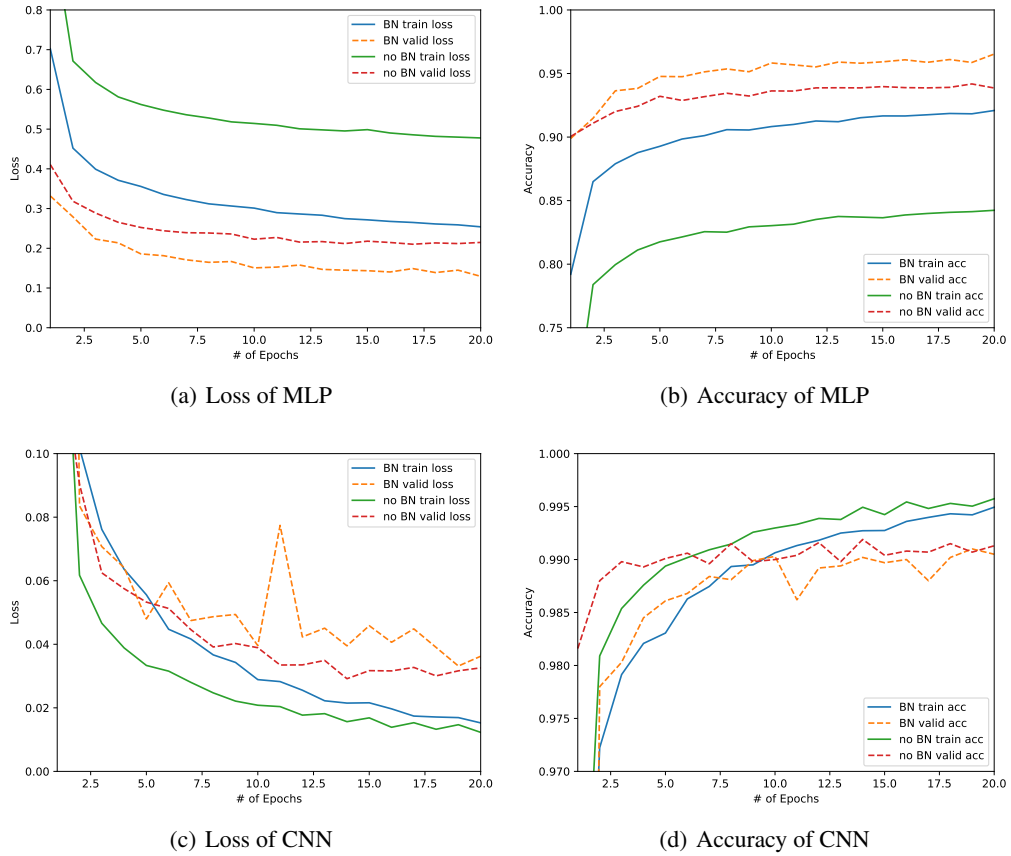
Table 3: Experiment Result of Default Setting

Model	Train Loss	Train Acc.	Val Loss	Val Acc.	Test Loss	Test Acc.
MLP	0.254016	92.090%	0.129405	96.530%	0.140109	95.950%
CNN	**0.015257**	**99.494%**	**0.036231**	**99.050%**	**0.029973**	**99.200%**

Table 3 illustrates the numeric result of this experiment. From Table 3, CNN greatly outperforms than MLP in training set, validation set and test set, both Loss and Accuracy metrics.

2.5 Question 4

With/Without "Batch Normalization" layer, the result of each epoch can be found in Figure 2.



(a) Loss of MLP

(b) Accuracy of MLP

(c) Loss of CNN

(d) Accuracy of CNN

Figure 2: Comparison of several Batch Normalization experiments.

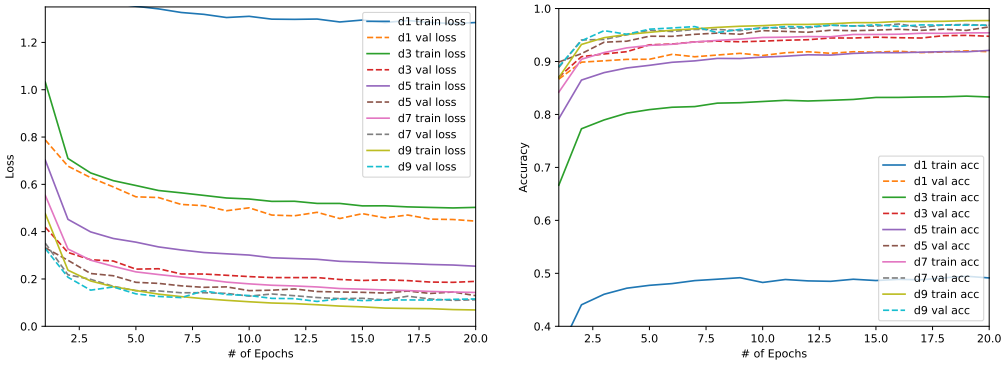
Without BN layer, the performance of MLP drop down a lot, in both metric and both training and validation dataset. From PAC, we know the VC dimension of MLP is very large. The BN layer could help regularize the model's parameter and lower the VC-dim.

However, the result of CNN is totally different from MLP. I think it can be explained from PAC. The CNN's VC-dim is much smaller than MLP's, thus in the same condition it doesn't need too much regularizer to enhance its performance. BN is helpful in extremely large neural network, which VC-dim is almost equal to a simple MLP, but could not help this kind of simple CNN with small VC-dim.

2.6 Question 5

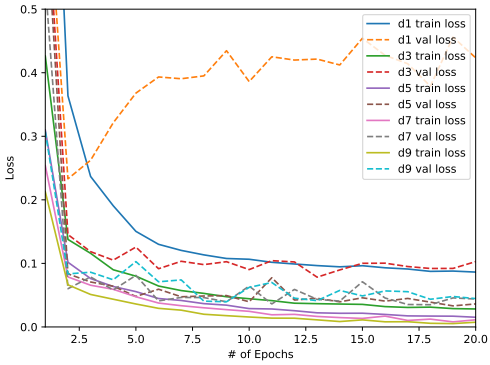
I have changed the drop rate by $\{0.1, 0.3, 0.5, 0.7, 0.9\}$ in each Dropout layer of the model, namely "d1", "d3", "d5", "d7", "d9", where "d5" is the default setting of previous experiments. The experiment result can be found in Figure 3.

From Figure 3, we can clearly see that the Dropout layer plays a regularizer role in the training procedure. The smaller the drop rate, the more effect of regularization it adds to. Take CNN as an example: when the drop rate is equal to 0.9, almost no dropout layer, the training loss is smaller than validation loss, thus it can not generalize well. When turning down the drop rate a little, such as 0.7 or 0.5, though the training loss becomes larger, the accuracy almost has no change. It indicates the model generalize well than the model without Dropout layer. But if we continue to turn down

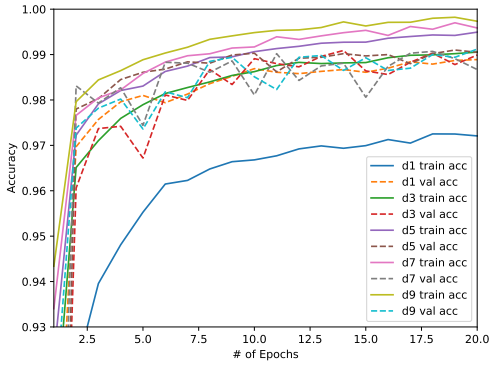


(a) Loss of MLP

(b) Accuracy of MLP



(c) Loss of CNN



(d) Accuracy of CNN

Figure 3: Comparison of several Dropout experiments.

the drop rate to nearly zero, the network cannot converge well enough. Thus a suitable drop rate is curious of the training process.

2.7 Question 6

Admittedly, there is a gap between training loss and validation loss. From the perspective of PAC, it gives a probabilistic upper bound estimation of this gap, which is:

$$\mathrm{P}\left(\text{test error} \leq \text{training error} + \sqrt{\frac{1}{N}\left[D\left(\log\frac{2N}{D}+1\right)-\log\frac{\eta}{4}\right]}\right) = 1-\eta$$

where D is the VC dimension of the model, $0 \leq \eta \leq 1$ is the learning rate, and N is the size of training set. When D is larger, the test error (or validation error, not used in the training process) may be much higher than training error. This is due to the overfitting.

When we look at the validation loss, it gives a rough estimation of overall data, if the data is i.i.d.. Thus we can use the validation data to pick a model. Also, the gap between training and validation loss/accuracy cannot be too high, otherwise the model cannot generalize well and would poorly perform in other dataset of same distribution.