REPORT FOR ARTIFICIAL NEURAL NETWORK ASSIGNMENT 2

Xiang Zhang
Department of Computer Science and Technology
Tsinghua University
xiang-zh17@mails.tsinghua.edu.cn

October 18, 2019

ABSTRACT

In this assignment, two types of neural network, i.e. MLP and CNN are implemented to perform classification task on CIFAR-10 dataset. Later, the effects of batch normalization and different drop rates are examined in order to help better understand the roles they play in training process. Moreover, a naïve VGG model is implemented as a supplementary and baseline as well.

1 Network Architecture

All the architectures conform to the requirements, only kernel size or initializers are tuned since they are not specified previously.

1.1 MLP

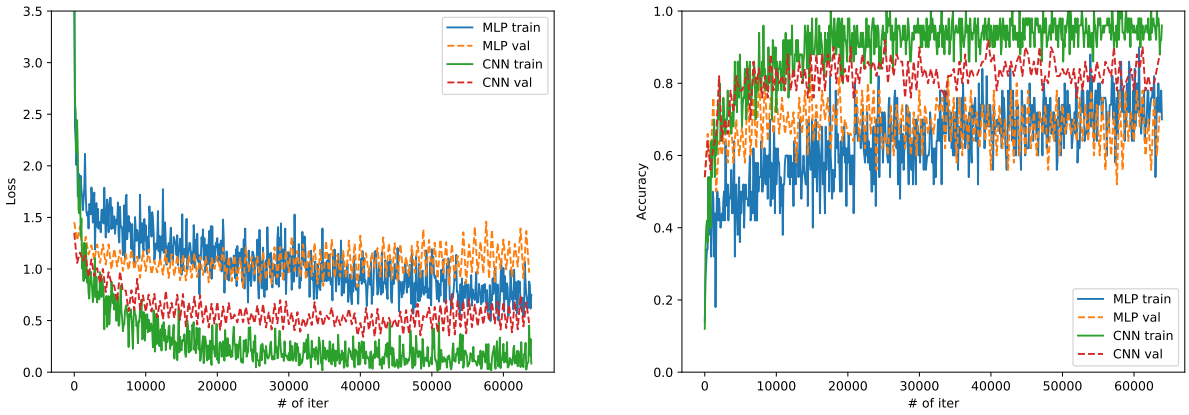
Layer	Type	Input Dim	Output Dim	Initializer
1	Linear	3072	512	truncated normal ($\sigma = 0.1$)
2	BN	512	512	
3	ReLU	512	512	
4	Dropout	512	512	
5	Linear	512	10	truncated normal ($\sigma = 0.1$)

1.2 CNN

Layer	Type	Input Dim	Output Dim	Initializer
1	Conv2D	$3 \times 32 \times 32$	$128 \times 32 \times 32$	he_uniform
2	BN	$128 \times 32 \times 32$	$128 \times 32 \times 32$	
3	ReLU	$128 \times 32 \times 32$	$128 \times 32 \times 32$	
4	Dropout	$128 \times 32 \times 32$	$128 \times 32 \times 32$	
5	MaxPool2D	$128 \times 32 \times 32$	$128 \times 16 \times 16$	
6	Conv2D	$128 \times 16 \times 16$	$256 \times 16 \times 16$	he_uniform
7	BN	$256 \times 16 \times 16$	$256 \times 16 \times 16$	
8	ReLU	$256 \times 16 \times 16$	$256 \times 16 \times 16$	
9	Dropout	$256 \times 16 \times 16$	$256 \times 16 \times 16$	
10	MaxPool2D	$256 \times 16 \times 16$	$256 \times 16 \times 16$	
11	Linear	65536	10	he_uniform

2 Experiment Results

2.1 Loss and Accuracy



(a) Loss (b) Accuracy

Figure 1: Training and validation accuracy along with loss against each iteration

The `drop_rate` parameter is default to 0.5 in this section. Batch size is set to 50 while number of epochs is 80.

3 Answers to the Questions (with Analysis)

3.1 Arguments of `model.forward`

For training process, `is_train` and `reuse` are set to true, whereas they are both set to false during test process.

In the implementation, the argument `is_train` is passed over to batch normalization and dropout function, which enables them to act properly in different scenarios. For training, dropout is enabled and the output of previous layers is scaled, while simultaneously the variance and mean value of samples are calculated and updated. However, for model testing, dropout needs to be disabled and the variance and mean value for normalization must be acquired from training process, since we are unable to derive proper values when the samples are tested one by one. This is why and how `is_train` is configured with respect to the type of execution.

In terms of `reuse`, as the name suggests, it controls whether the variables are reused when executing. Conspicuously, `reuse` should only be enabled when we are testing, for all the model weights are loaded from file.

3.2 Comparsion of MLP and CNN

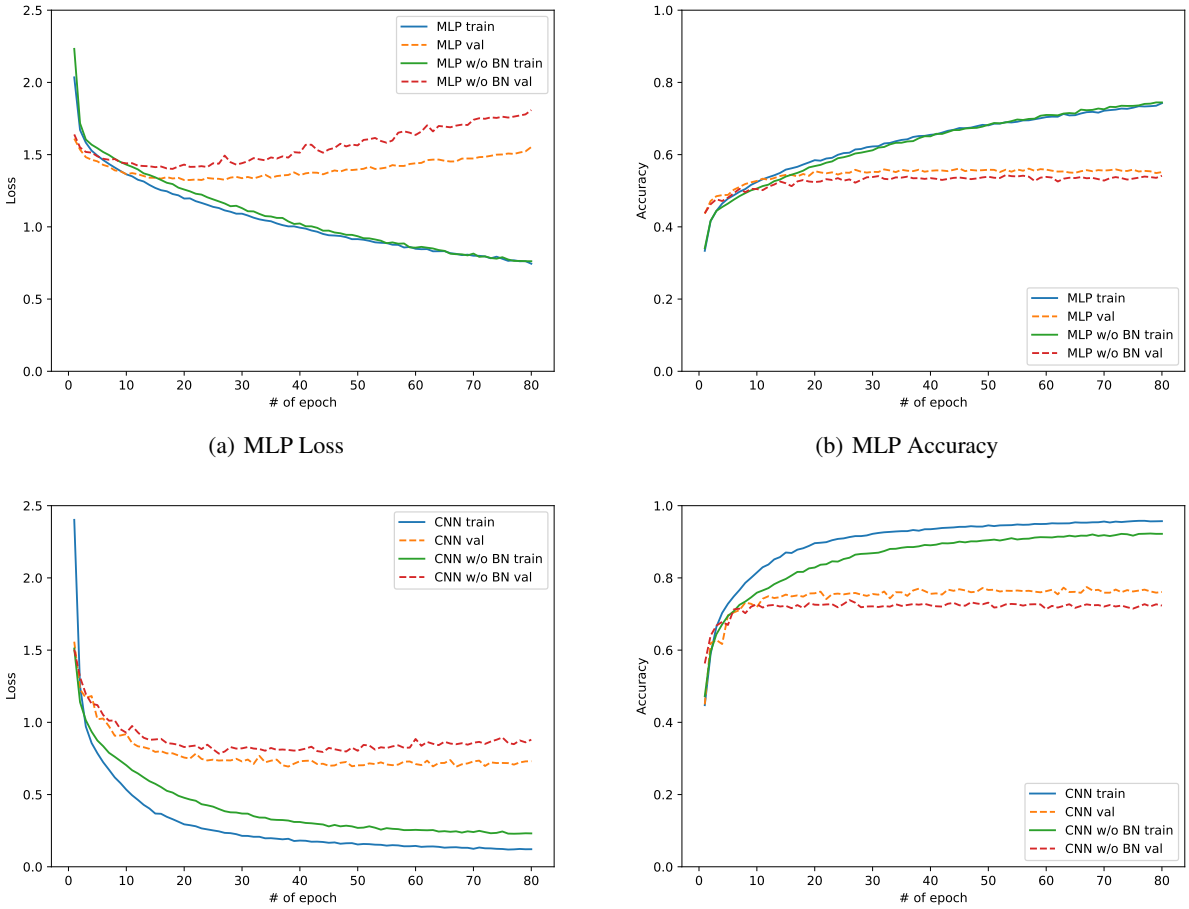
Table 3: Numerical model performance of MLP and CNN (from best epoch)

Model	Train loss	Train acc	Val loss	Val acc	Test acc
MLP	0.9418	0.6736	1.3777	0.5617	0.5539
CNN	**0.1354**	**0.9530**	**0.6923**	**0.7747**	**0.7616**

From **Table 3** and **Fig. 1**, it is obvious that CNN outperforms MLP in every aspect. Not only does it converges better, yielding a much higher accuracy than MLP, but it also tends to overfit less. After iteration 20000, the loss of MLP network begins to rise up and its variance increases simultaneously, rendering the value unstable, whereas CNN is continually lowering its loss during the process of training.

No wonder CNN performs better when it comes to image data, since it can properly utilize the 2D spatial information within the data through convolution, while MLP requires flattening of images before training, which erases the spatial correlation of pixels. Moreover, pure MLP network is prone to overfitting, yielding a worsened performance after many iterations.

3.3 The Effect of Batch Normalization



(a) MLP Loss

(b) MLP Accuracy

(c) CNN Loss

(d) CNN Accuracy

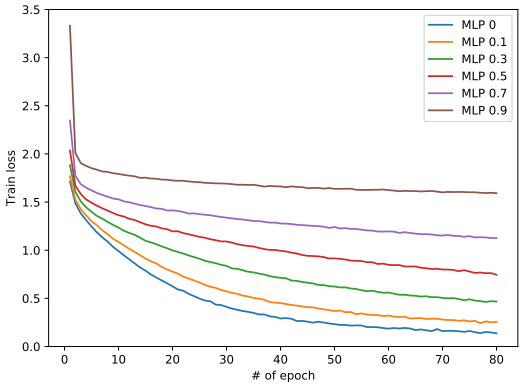
Figure 2: Comparison of MLP/CNN performance with or without batch normalization

From **Fig. 2**, batch normalization boosts the performance of both MLP and CNN but to different extents. It has little impacts on the training process of MLP, whereas it can increase the accuracy dramatically during the training of CNN.

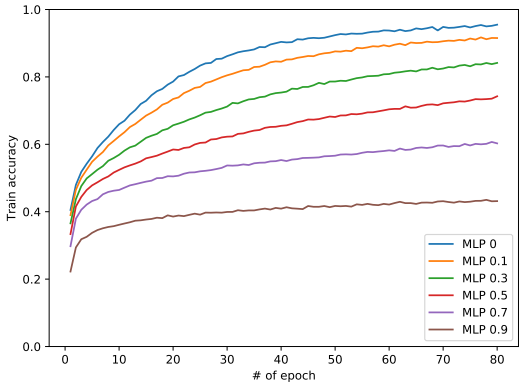
The batch normalization technique mitigates covariance shift problem that might be introduced with training data, especially for dataset (CIFAR-10) used in this task. Through normalizing the output of previous layers, it helps to accelerate the training process. In terms of the different behaviors when applied to MLP and CNN, it can partially be explained by CNN's smaller number of parameters, which is subject to high variance of training data.

3.4 Droprate Tuning

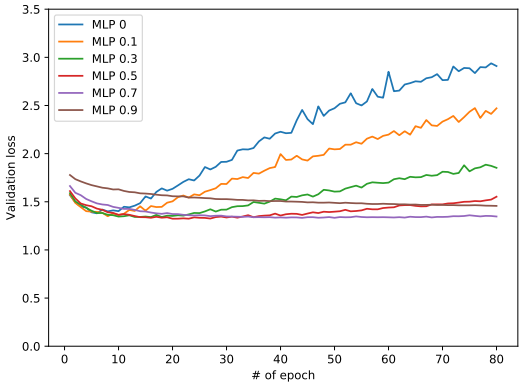
In the following experiments, each model is evaluated using droprate set $\{0,\ 0.1,\ 0.3,\ 0.5,\ 0.7,\ 0.9\}$, where 0 means there is no dropout and 0.5 is the default setting of previous experiments. The results are shown in **Fig. 3**.



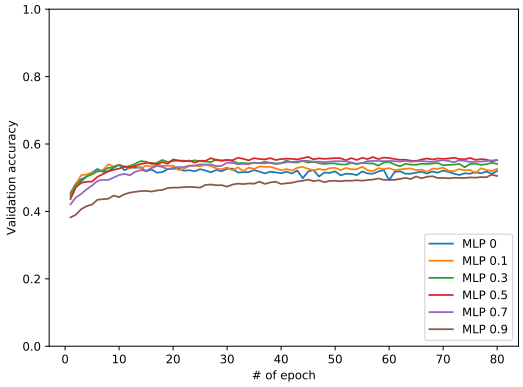
(a) MLP Train Loss



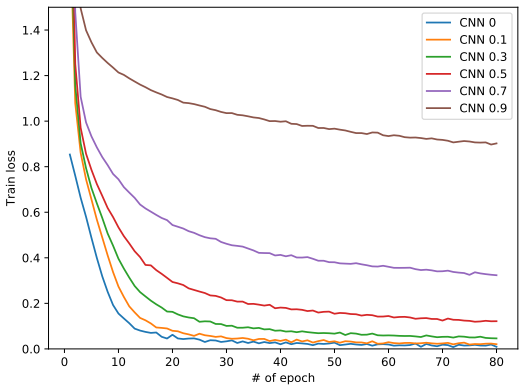
(b) MLP Train Accuracy



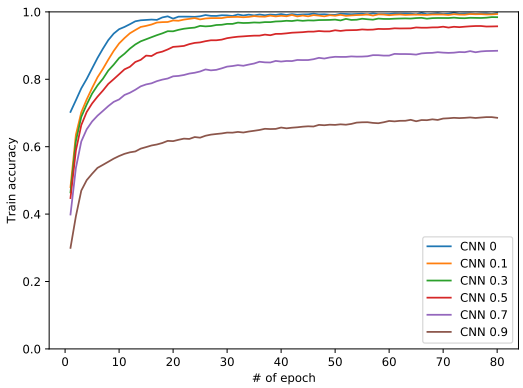
(c) MLP Validation Loss



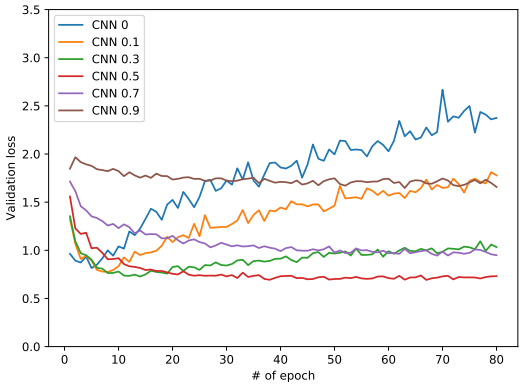
(d) MLP Validation Accuracy



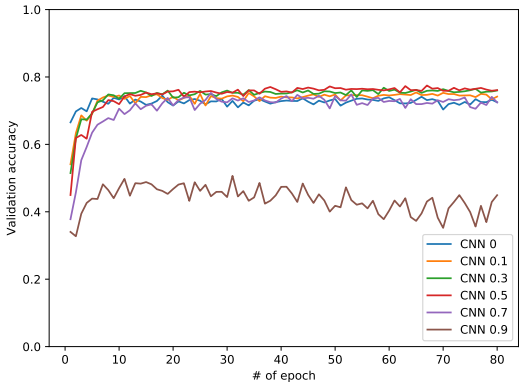
(e) CNN Train Loss



(f) CNN Train Accuracy



(g) CNN Validation Loss



(h) CNN Validation Accuracy

Figure 3: Comparison of MLP/CNN performance in different droprates

From **Fig. 3**, the presence of dropout layer exhibits distinct impact on performance in terms of type of the model used. During the training phase, the speed of learning decreases monotonically as the drop rate increases. For MLP network, it is subject to excessive overfitting when drop rate ≤ 0.3, in which case, the validation loss exceptionally rises after 10 epochs. In terms of CNN, it suffers less from overfitting when drop rate $= 0.3$. (Yet for drop rate lower than this sample, overfitting can still take over.) These phenomena corroborate that dropout layer can forestall overfitting of the model to some extent, however, improperly high drop rate can impair the performance by dropping out excessive amount of information, which can be illustrated in MLP and CNN with drop rate 0.9, and the latter is affected more, yielding a validation accuracy approximately half of other samples.

For our settings (the network architecture and dataset), best performance occurs when drop rate $\sim 0.5 - 0.7$ for MLP and ~ 0.5 for CNN.

3.5 Training and Validation Loss

There are several factors contributing to the difference present in training and validation loss, and in this experiment, the distribution of dataset, batch normalization and dropout layer. The latter two behave distinctively during training and validating, in which case, BN calculates the mean and variance during training and reuses them in validating, whereas, unlike training process, dropout is disabled during validating. When BN is not present, training loss can resemble that of validating, which is exemplified in **Fig. 4**.

The training and validation loss can help us to select the best model for the specific task. If the gap between them is too high, the model would not generalize well, i.e. train loss $\gg$ validation loss indicates underfitting, while the contrast signals overfitting. When their values are appropriate, we can utilize the validation loss to determine the best epoch during training.

4 Implementation of Supplementary Baseline

Since the network architectures provided in requirements perform badly on CIFAR-10 datsaset, I tried to implement a supplementary baseline based on VGG model, and here is the basic architecture:

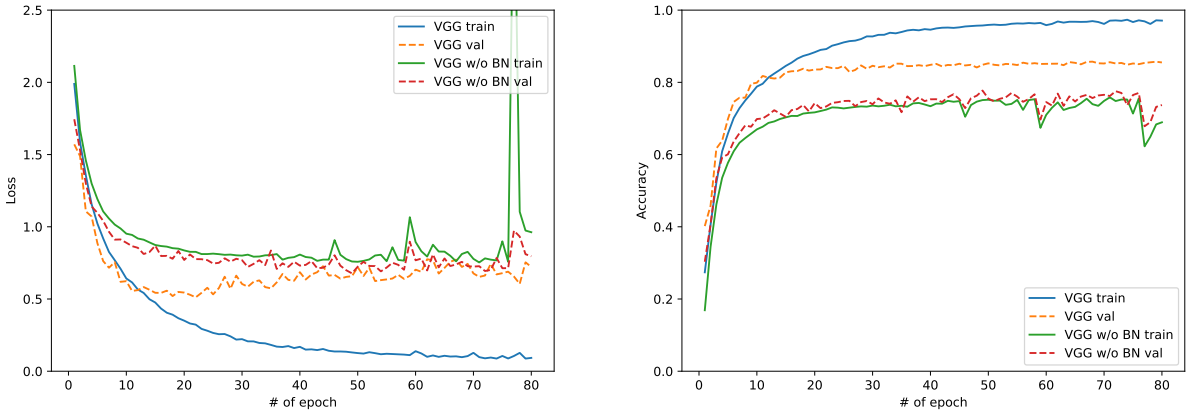
Table 5: The architecture of quasi-VGG model

Layer	Type	Activation	Params
1, 2	Conv2D	ReLU	filters=32
3	MaxPool2D		size=2, stride=2
4	Dropout		droprate=0.3
5, 6	Conv2D	ReLU	filters=64
7	MaxPool2D		size=2, stride=2
8	Dropout		droprate=0.3
9, 10	Conv2D	ReLU	filters=128
11	MaxPool2D		size=2, stride=2
12	Dropout		droprate=0.4
15, 16, 17	Conv2D	ReLU	filters=128
18	MaxPool2D		size=2, stride=2
19	Dropout		droprate=0.4
20	Linear		hidden_unit=4096
21	Dropout		droprate=0.4
22	Linear		hidden_unit=512
23	Dropout		droprate=0.4
24	Linear		

4.1 Performance

Table 7: Numerical model performance of quasi-VGG

Model	Train loss	Train acc	Val loss	Val acc	Test acc
quasi-VGG	**0.0972**	**0.9696**	**0.7219**	**0.8573**	**0.8496**
quasi-VGG (w/o BN)	0.7597	0.7509	0.6775	0.7772	0.7721



(a) Loss (b) Accuracy

Figure 4: Training and validation loss/accuracy of quasi-VGG with or without BN

Without batch normalization, the performance of quasi-VGG degrades drastically since deep layers of convolution can suffer from high covariance shift more. Note in this case the training/validation accuracy and loss closely resemble each other, while they differ much with the presence of batch normalization, as shown in **Fig. 4**.