MACHINE LEARNING CW4 : REPORT

Solang Kim
School of Computer Science Engineering
Ulsan National Institute of Science and Technology (UNIST)
solangii@unist.ac.kr

June, 2021

1 Task1

Random Forest Randomforest was run using sklearn. At this time, the hyperparameter of the random forest classifier is executed by changing the number of **n estimator** to 50.

MLP MLP is a simple MLP with three hidden layers. The number of nodes in the hidden layer is the same as 200, with input layer having 3072 and output layer having 10 or 100 layers. The hyperparameter configuration used in the experiment was set to **cross-entropy** for loss function, **SGD** for optimizer, **0.005** for learning rate, **100** for epochs, and **100** for batch size.

CNN CNN is a simple CNN with the most basic LeNet-5 structure. The hyperparameter configuration is the same as the MLP.

In the case of MLP and CNN, train data and val data are organized in a 1:1 ratio to proceed with learning. Run train, val per epoch. During the validation process, the test accuracy was measured by saving the weight of the highest value as a checkpoint.

And I experimented in a gpu environment to make calculations easier. (quadro 8000 * 6ea)

Table 1: simple result

Dataset	**Model**	**Accuracy**	**Total run time (sec)**
CIFAR-10	RF	42.19 %	32.48
	MLP	44.3 %	1046.74
	CNN	**64 %**	1084.46
CIFAR-100	RF	16.7 %	105.46
	MLP	2.45 %	1052.82
	CNN	**24 %**	1105.42

result Table[1] is the result of the experiment according to the hyperparameter configuration mentioned above. Based on the results, Random Forest was the best in terms of run time, and CNN was the best in terms of accuracy. We need to look at it a little more from a run-time perspective. The running time of the MLP and CNN shown in Table[1] is the total running time. This means that each epoch needs to be looked at in terms of running time as well. Because the running time varies greatly depending on the number of epochs.

If you look at Fig[1], the runtime per epoch of Neural networks is typically near 10. There is not much difference between each network. Due to the nature of neural networks, this case should be considered because the execution time varies significantly by epoch.

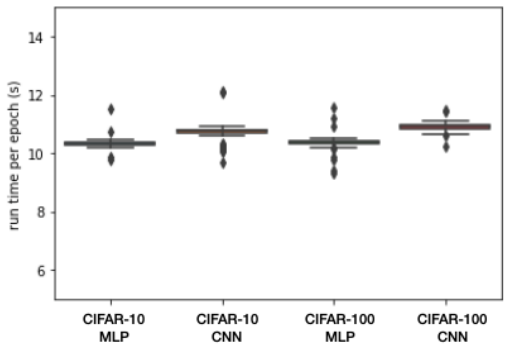

Figure 1: run time per epoch for NN

2 Task2

2.1 Random Forest

I did further experiments on the Random Forest.

n estimator To determine the effect of n estimator, which means the number of trees, on learning, I change the n estimator option to {25,50,75,100} to measure the accuracy and running time. The experiment was conducted on the CIFAR-10 dataset.

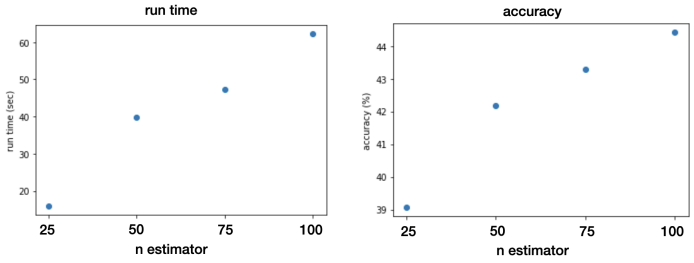

Figure 2: change n estimate

Fig[2] shows that as the number of trees increases(as the n estimator increases), the accuracy and execution time increase . This indicates that run time and accuracy are trade-off in randomforest.

number of train data To determine the effect of number of train data, on learning, I change the number of train data to {10000,20000,30000,40000, 50000} to measure the accuracy and running time. The experiment was conducted on the CIFAR-10 dataset.

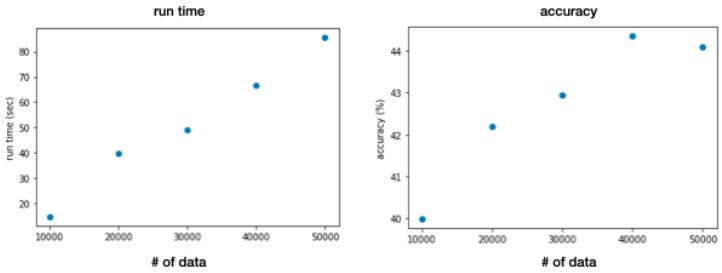

Figure 3: change # of data for Random Forest

Fig[3] shows that as the number of data, the run time increase. However, the accuracy dropped from 40000 to 50000. I think it's best to use 40,000.

2.2 Neural Network

I did further experiments on the Nerual Network.

learning rate To determine the effect of learning rate, on learning, I change the learning rate option to {0.1, 0.01, 0.005, 0.001} to measure the accuracy, loss and running time. The experiment was conducted on the cifar-10 dataset and CNN architecture(LeNet). Other hyperparameter configurations were used as mentioned in Task 1.

Table 2: Learning Rate Experiment Results

Dataset	Learning Rate	Test Accuracy
CIFAR-10	0.1	25 %
	0.01	61 %
	0.005	64 %
	0.001	55 %

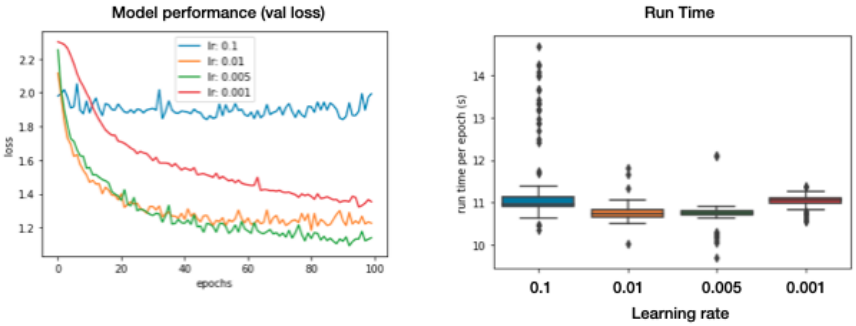

Figure 4: Learning rate

The left graph of Fig[4] is a plot of the loss mean value according to epoch, and the right graph is a boxplot of the run time for each setting. Table[2] shows the best performance at lr=0.005. The same result can be found in the loss graph. In the case of lr=0.1, where accuracy is the worst, Loss is also not converging. The accuracy and validation loss graph show the performance of the model. In the end, finding the appropriate learning rate allows the model's loss value to converge and achieve good accuracy. Also, the running time graph shows that learning rate and running time are not related.

number of train data To determine the effect of number of train data, on learning, I change the number of train data option to {10000, 20000, 30000, 40000, 50000} to measure the accuracy, loss and running time. The experiment was conducted on the cifar-10 dataset and CNN architecture(LeNet). Other hyperparameter configurations were used as mentioned in Task 1.

In each experiment, the number of train data was first determined as the above option. Half the amount of train data was used as actual train data and the remaining half as validation data. I measure test accuracy on the model that performs best in validation.

The results showed that the higher the number of train data, the better the performance of the model, given the accuracy(Table[3]) and loss graph(Fig[5]). However, as the number increases, so does the run time. The model performance and run time based on the number of data showed that the two factors were trade-off relations. If the gpu environment is sufficient, the more data is better.

other NN architecture LeNet[LeCun et al., 1998] is a structure developed by Yann LeCun, who first developed the concept of Convolution Neural Network. The CNN network I have used so far also follows the LeNet structure. After

Table 3: # of train data Results

Dataset	# of train data	Test Accuracy
CIFAR-10	10000	56 %
	20000	64 %
	30000	66 %
	40000	66 %
	50000	69 %

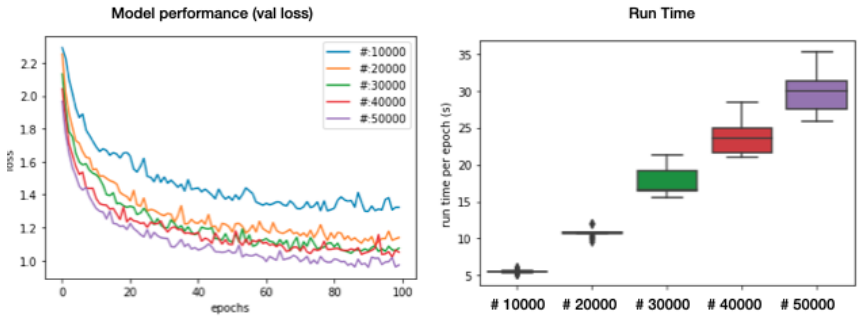

Figure 5: Number of train data

repeated convolution and subsampling, we perform classification with a fully connected multi-layer neural network at the end.

I wanted to implement other famous CNN architectures besides LeNet[LeCun et al., 1998] and see how different the performance is. Therefore, I conducted additional experiments on ResNet18[He et al., 2016], Vgg19[Simonyan and Zisserman, 2014], and GoogleNet[Szegedy et al., 2015].

The hyperparameter configuration used is the same as above1.

Table 4: Other CNN Results

Dataset	Model	Test Accuracy
CIFAR-10	LeNet[LeCun et al., 1998]	64 %
	ResNet-18[He et al., 2016]	74 %
	GoogLeNet[Szegedy et al., 2015]	74 %
	VGG-19[Simonyan and Zisserman, 2014]	**78 %**

A noticeable result was that convergence of all models except LeNet was much faster. Most of them also showed significant performance improvements of more than 10% and speed of convergence was also accelerated. Runtime, by comparison, has increased slightly, but not dramatically.

There are various reasons for making this possible for each model. It is common that they all have deeper layers than LeNet. But deep layers are not necessarily good. If the layer is deep, there must be a required computation or gradient vanishing problem. Thus, it has unique structures that deepens the model while addressing these problems.

In the case of GoogLeNet, there is a module called inception. A sparsely connected structure, such as a network that uses Dropout, is created to perform efficient operations. There are 22 layers in total.

For VGG, filters (3,3) were used on all layers. This is a minimum size filter, which shows efficiency in computation. I used VGG19 with 19 layers.

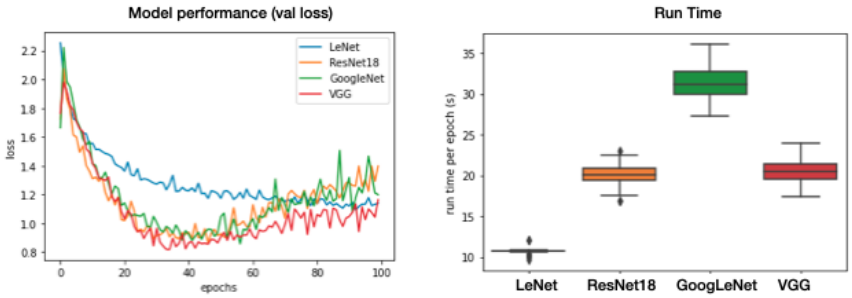

Figure 6: Other CNN

In the case of ResNet, there is a special module called Residual block. Due to the phenomenon that the deeper the layer becomes, the lower the performance of the layer, the more frequent learning using skip connection could be used to solve such problems. I used the 18th floor layer.

The total results showed the best results on VGG-19. At first, I thought ResNet would perform best, but it was an unexpected result. However, we expect to perform better than VGG-19 if we deepen the layer of ResNet.

Fasion MNIST Dataset Fasion MNIST is a dataset of 60,000 learning data, 10,000 test data, and images (28,28,1). I conducted an experiment on FasionMNIST to find out how it performs on other datasets.Other hyperparameter configurations were used as mentioned in Task 1.

Table 5: Other Dataset

Dataset	Model	Test Accuracy
CIFAR-10	LeNet-5 (base CNN)	64 %
CIFAR-100		24 %
FasionMNIST		85 %

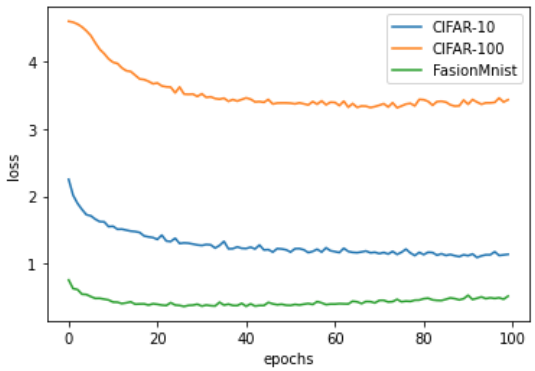

Figure 7: effect of dataset

Based on the results Table[5] and Fig[7],Experiments on datasets show that the more classes to match, the less performance. CIFAR10 and Fasion MNIST have 10 classes, whereas CIFAR100 is relatively difficult to learn. This requires deeper network layers or more training through hyperparameter manipulation.

Also, the number of channels seems to be affected. For Fasion MNIST, the number of channels is 1 greyscale image. CIFAR10, on the other hand, has an RGB value of 3. Therefore, if the number of channels is small, it seems to be able to perform better.(Easy to learn)

Ablation for LeNet Baseline LeNet had a relu function and max pooling. I was curious about the effect of **pooling** and the effect of **activation function**, so I changed the function and tested it.

Table 6: effect of pooling and activation function

Dataset	Model	Test Accuracy
CIFAR-10	LeNet-5 (base)	64 %
	LeNet-5, avg pool	59 %
	LeNet-5, avg pool + tanh	56 %

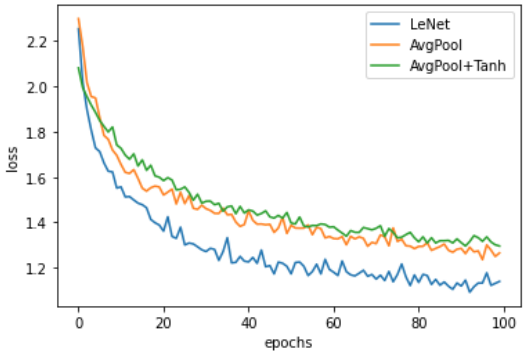

Figure 8: effect of pooling and activation function

Based on the results Table[6] and Fig[8], LeNet's results are better when using the ReLU function than Thanh, and for pooling, maxpooling is better than avg pooling.

References

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.