INT305 - Assignment 2 - Report

1. Training, Tuning and Testing of the Framework

1.1 Environment

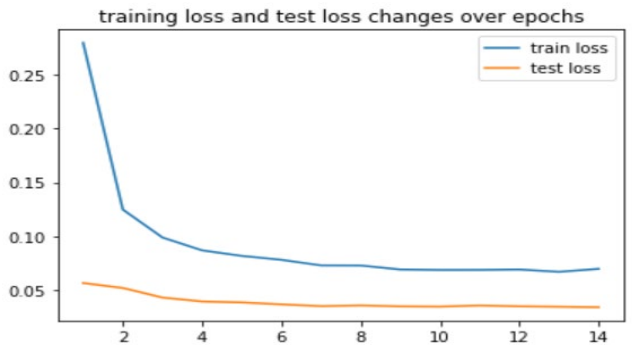
All the progress are done in the following environment to ensure the consistency and fairness.

CPU	GPU	System	PyTorch	CUDA	Python
Intel Core i7 – 10700F	NVIDIA RTX 2070	Windows 10	1.7.1	10.1	3.7.7

Data:

The handwritten digits dataset used is MNIST dataset from Yann Lecun's website.

1.2 Baseline Model



By train and test the sample model, the final performance of the sample model is 98.81%, which means the majority of the handwritten digits could be recognized correctly. By plotting the figure of how average training loss and average test loss changes over epochs, we could find out that the model performs well and is not overfitting.

Some typical well classified and misclassified images are picked below for comparison to find out the reason may cause misclassified:

Misclassified case:

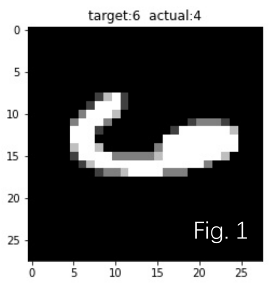


Fig. 1

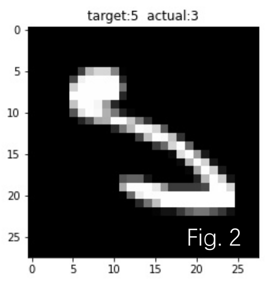


Fig. 2

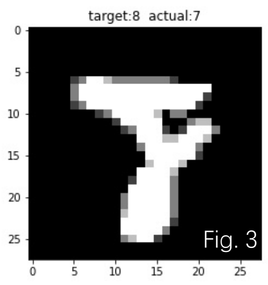


Fig. 3

Well classified case:

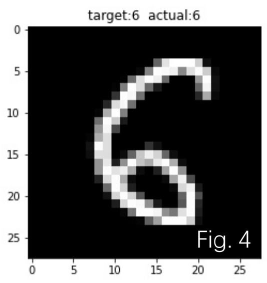


Fig. 4

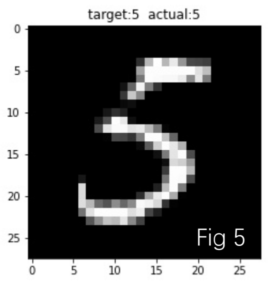


Fig 5

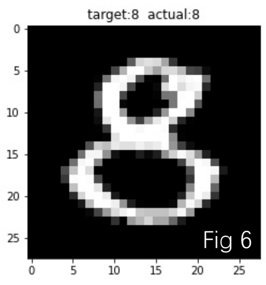


Fig 6

Confidence values of the example cases

No	0	1	2	3	4	5	6	7	8	9
Fig.1	0	0	0	0	0.9936	0.0001	0.0039	0	0.0001	0.0022
Fig.2	0	0.0022	0.0158	0.7797	0	0.1506	0.0140	0.0004	0.0373	0
Fig.3	0	0	0.0002	0.0031	0.0004	0.0002	0	0.8599	0.0013	0
Fig.4	0	0	0	0	0	0.0001	0.9986	0	0.0013	0
Fig.5	0	0	0	0.002	0	0.998	0	0	0	0
Fig.6	0	0	0	0.0017	0	0.0003	0	0	0.998	0

3 Groups of typical cases are picked. From the figure, we could find that the characters of the misclassified cases:

a) Having a wrong rotation of the digit, which may extract wrong feature.

b) Having a tight handwriting, which hard to extract the correct feature.

c) Lacking enough feature to recognize, even could be misclassified by human.

Besides, in the misclassified cases, the model is not that "confidence" as the well classified cases. For example, the highest probability of fig 2 is 77.97% while the fig 4 got 99.86%.

For the reasons a) and b), we could use have use some machine learning techniques to improve the model's performance. For c), it is the unavoidable error. The goal of the improvement is to achieve the Bayesian Optimal Error.

1.3 Hyperparameter Tuning

The four hyperparameters tuned in the sample model are:

Hyperparameter	Default Value	Description
Batch size	64	input batch size for training
Epochs	15	number of epochs to train
Learning rate	1	learning rate
Gamma	0.7	learning rate step gamma

The progress of hyperparameter tuning is shown below

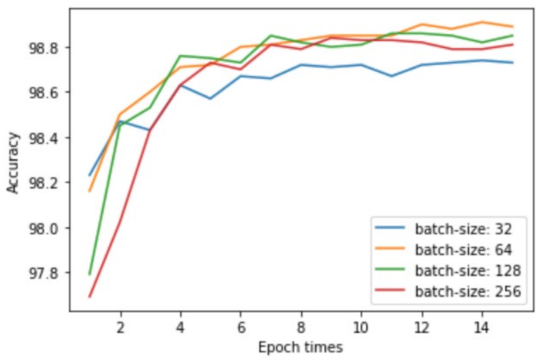
1. Determining the suitable range of the variable by change one and use default for others.
2. Using random search to find the combination of the variable, aiming to explore the variable space as wide as possible.

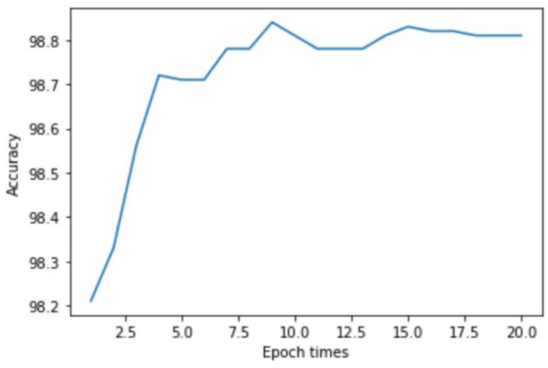
To ensure the reproducibility, the model use the same random seed.

2.3.1 Batch-size:

Batch size	32	64	128	256
Test Accuracy	98.73%	98.89%	98.85%	98.81%
Avg. Test Loss	0.0394	0.0358	0.0338	0.347

From the table and the figure, we found batch size matters the smoothness of the curve, but little on the accuracy, large batch size had a worse performance and lower cost. Therefore, when batch size went large, it might cause overfitting.





2.3.2 Epochs

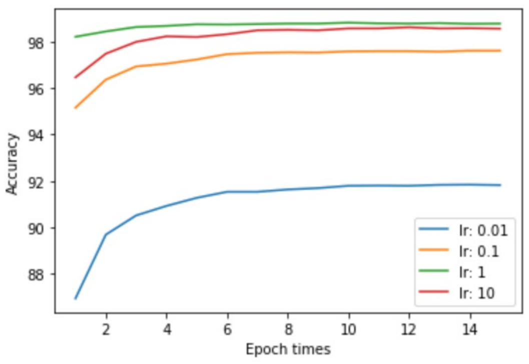
Epoch	10	15	20
Accuracy	98.81	98.83	98.81
Avg. Test Loss	0.0363	0.0359	0.0359

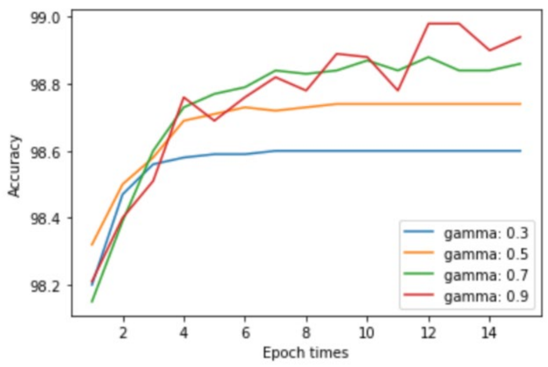
From the figure, we could find in the default setting, the learning epochs and accuracy have a positive correlation when epochs < 10, but when achieving 98.8% accuracy, it makes no contribution to the accuracy.

2.3.3 Learning rate:

Learning rate	0.01	0.1	1	10
Accuracy	91.82%	97.62%	98.79%	98.57%
Avg. Test Loss	0.2967	0.0766	0.0369	0.0472

The small learning rate like 0.01 and 0.1 performed bad because of the learning rate decay and the actual learning rate after several epochs is too small to descent. Besides, 1 and 10 have a similar final performance.





2.3.4 Gamma:

Gamma	0.3	0.5	0.7	0.9
Accuracy	98.6	98.74	98.86	98.94
Avg. Test Loss	0.0462	0.0406	0.0359	0.0367

The large gamma values have a good performance than the small one. In the later epochs, the small gamma value makes the learning rate too slow to descent. From the picture, we could see after 12 epochs, the model with 0.9 gamma still got a increase of performance.

2.3.5 Random Search

By the above experiments, we choose range of the hyperparameter and do random search for hyperparameters. The table records the highest score achieved and its hyperparameters.

Accuracy	Batch Size	Epochs	Learning Rate	Gamma
99.23%	64	35	1.5	0.95

2. Improvement

From the part 2 analysis and tuning, the two major commonplaces of the misclassified figures are the abnormal rotation of digit and the tight handwriting, which refers to dirty data and unapparent feature.

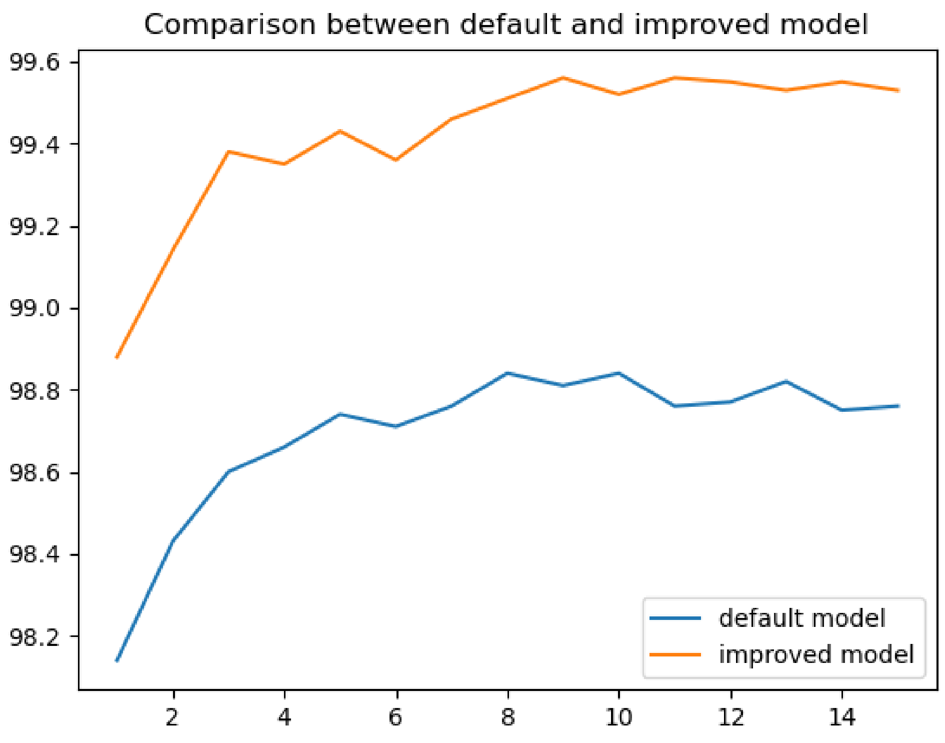
For the dirty data, we could do the data augment. For the unapparent feature, we could build a larger and deeper CNN, the larger first fewer layers could extract more features and the deeper layer could combine the primary features to obtain senior features.

Data Augment:

There are several data augment method we could use, add noise of the data, randomly rotate the image, randomly shift the image in a short distance, randomly magnify an area in the digit [2]. All these methods aiming to add more data that are had to extract features, so that the model could be more robust to handle the dirty data. Besides, there are more advanced methods like Mixup [3], and Augmix [4]. In this experiment, the rotate, shift, scale, shear will be used.

Change Model Structure

By adding layers to the model, the final model has 7 convolution layer and 1 full connected layer, max pooling, drop out and batch normalization was added to prevent



the overfitting [5]. Besides, to avoid vanishing gradients, ELU function was chosen as the activation function.
Using the default hyperparameter, the two model's performance is shown above. From the diagram, it is obvious that the improved model performed better than the default one and is more robust since the joggle extent is less when approaching the limit.

By doing hyperparameter tuning of the improved model, the final score and hyperparameters are shown as below:

Accuracy	Batch size	epochs	Learning rate	gamma
99.57%	128	21	0.05	0.95

4. Reference

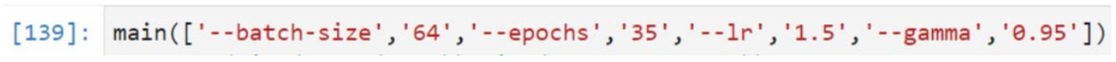
1. A. Ashiquzzaman, A. K. Tushar, A. Rahman, and F. Mohsin, “An Efficient Recognition Method for Handwritten Arabic Numerals Using CNN with Data Augmentation and Dropout: Proceedings of ICDMAI 2018, Volume 1,” 2019, pp. 299–309. doi: 10.1007/978-981-13-1402-5_23.

2. H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, “mixup: Beyond Empirical Risk Minimization,” arXiv:1710.09412 [cs, stat], Apr. 2018, Accessed: Nov. 30, 2021. [Online]. Available: http://arxiv.org/abs/1710.09412

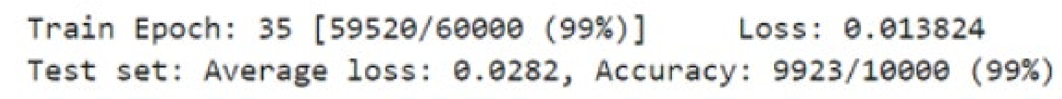
3. D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan, “AugMix: A Simple Data Processing Method to Improve Robustness and Uncertainty,” arXiv:1912.02781 [cs, stat], Feb. 2020, Accessed: Nov. 30, 2021. [Online]. Available: http://arxiv.org/abs/1912.02781

5.Appendix

5.1 Log Result for the Sample CNN after Hyperparameter tuning

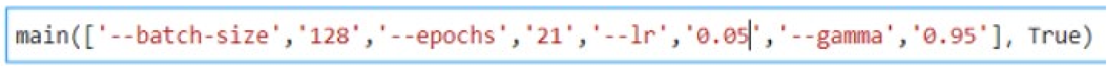


```
[139]: main(['--batch-size','64','--epochs','35','--lr','1.5','--gamma','0.95'])
```

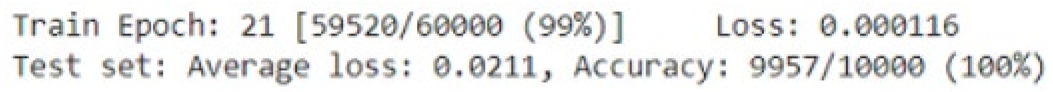


```
Train Epoch: 35 [59520/60000 (99%)]     Loss: 0.013824
Test set: Average loss: 0.0282, Accuracy: 9923/10000 (99%)
```

5.2 Lor Result for the Improved CNN



```
main(['--batch-size','128','--epochs','21','--lr','0.05','--gamma','0.95'], True)
```



```
Train Epoch: 21 [59520/60000 (99%)]     Loss: 0.000116
Test set: Average loss: 0.0211, Accuracy: 9957/10000 (100%)
```