AI3607 Deep Learning: Assignment 3

Xiangyuan Xue (521030910387)
School of Electronic Information and Electrical Engineering

I. Dataset Construction

Download MNIST dataset from the source. The dataset contains 70000 28×28 greyscale images of hand-written digits, where 60000 images are used for training and 10000 images are used for testing.

TABLE I
MNIST COMPOSITION

#Train	#Test	#Total	#Class
60000	10000	70000	10

Some images in the dataset is shown in the following figure.

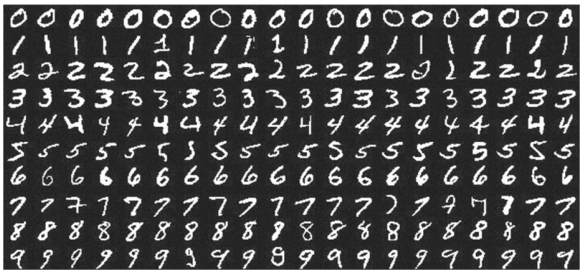

Fig. 1. MNIST dataset images

We can construct our customized dataset by overriding the `Dataset` class, which will be discussed in the following sections.

II. Neural Network

Admittedly, CNN-based models are most commonly used in visual classification tasks. But here we will try to solve this problem with RNN-based models.

A. Elman Network

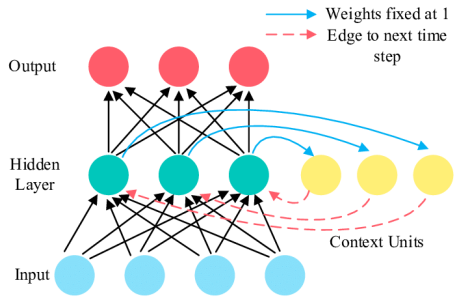

Fig. 2. Elman network model

Elman network is a simple RNN model dating back to the 1980s. It is a feedforward neural network with a recurrent connection, which can be used to model sequential data. [1]

For the image classification task, we regard a 28×28 image as 28 sequences of pixels. Each sequence contains 28 pixels, which is taken as the input of the network, which indicates that the input dimension is 28. The network has only 1 hidden layer, which contains 128 neurons and provides a 128-dimensional vector. With a fully connected layer, the output dimension is 10, corresponding to the number of classes.

B. LSTM Network

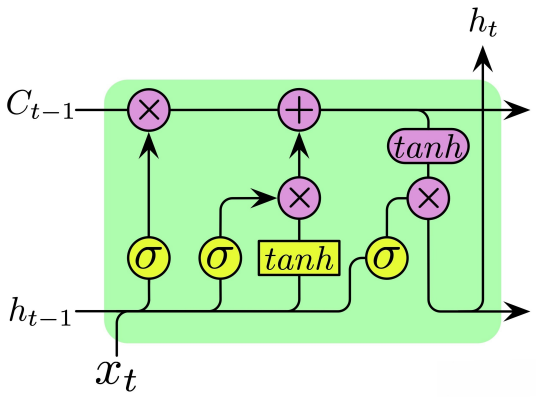

Fig. 3. LSTM network model

RNN is a powerful model for sequential data, but it suffers from the vanishing gradient problem and fails to learn long-term dependencies. LSTM is a variant of RNN, which can solve the problem by introducing a memory cell. Each LSTM cell contains four gates, controlling the flow of information. [2]

Despite the complexity of the model, the input and output dimensions are the same as the Elman network, so the training and testing processes are exactly the same. Since LSTM can learn long-term dependencies better, we expect a better performance compared to the Elman network.

III. Experiment Result

Run the experiment script and we train both Elman network and LSTM network on the original dataset. Since the MNIST dataset is relatively easy, both networks achieve high accuracy within a few epochs. The experiment result is reported in the following table.

We can see that both networks achieve high accuracy and converge quickly, where LSTM network performs slightly

TABLE II
EXPERIMENT RESULT

Network	Set	Loss	Accuracy
Elman	Train	0.6303	99.75%
	Test	0.7533	98.16%
LSTM	Train	0.0097	100.00%
	Test	0.6692	98.63%

better than Elman network. Note that LSTM network reaches 100% accuracy on the training set, indicating that overfitting problem happens. However, this problem is not as severe as in the CNN-based models since the accuracy is relatively close.

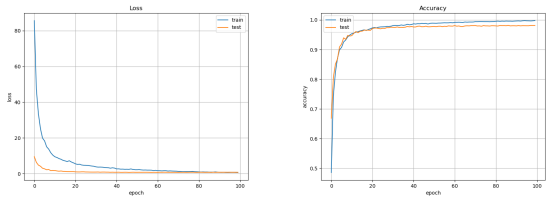

Fig. 4. Elman network training curve

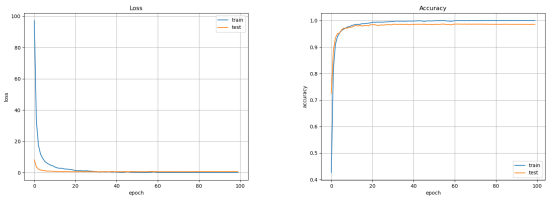

Fig. 5. LSTM network training curve

The training curve shows that Elman network converges after 80 epochs, while LSTM network converges after 20 epochs. Since LSTM network performs better in various aspects, we will apply it as the baseline model for the following experiments.

IV. Unbalanced Dataset

The unbalanced dataset is created with a little modification, where the images with label $(0, 1, 2, 3, 4)$ are removed by 90%, while the images with label $(5, 6, 7, 8, 9)$ remain unchanged. Hence, the size of the training set is reduced, which can lead to performance degradation.

TABLE III
UNBALANCED DATASET

Label	0	1	2	3	4
#Train	592	674	595	613	584
#Test	980	1135	1032	1010	982
Label	5	6	7	8	9
#Train	5421	5918	6265	5851	5949
#Test	892	958	1028	974	1009

Train the LSTM network on the unbalanced dataset and the result is reported in the following table. We can see that the accuracy drops especially on the classes $(0, 1, 2, 3, 4)$, which is caused by the imbalance of the dataset.

TABLE IV
UNBALANCED RESULT

Group	Set	Loss	Accuracy
Unbalanced	Train	0.0126	100.00%
	Test	1.6320	96.72%

The training curve is also shown in the following figure.

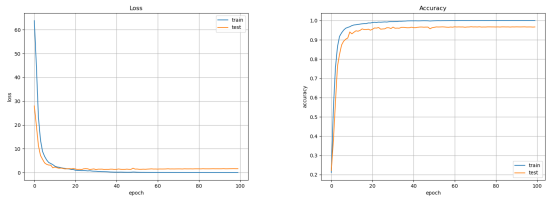

Fig. 6. reduced dataset training curve

We can see that the accuracy dropped by about 2% on the unbalanced dataset. However, the accuracy on the training set is still 100%, indicating that the overfitting problem is even more serious. This problem comes from the imbalance of training, where the model is trained too much on the classes $(5, 6, 7, 8, 9)$. Hence, the model works much worse on the classes $(0, 1, 2, 3, 4)$. We will try to handle this problem in the following sections.

V. Model Improvement

A. Data Augmentation

Data augmentation is a good choice to rebalance the dataset. Here we use a simple method to augment the training set, where Guassian noise is added into the images with label $(0, 1, 2, 3, 4)$ and the variances of the noise are respectively $0.1, 0.2, \ldots, 0.9$, namely

$$\boldsymbol{N} \leftarrow \mathcal{N}(0, \sigma^2) \tag{1}$$

$$\boldsymbol{X} \leftarrow \boldsymbol{X} + \boldsymbol{N} \tag{2}$$

Data augmentation restores the size of the training set and make the dataset balanced, which may improve the performance of the VGG model. [3]

B. Model Generalization

The imbalance is essentially an out-of-distribution generalization problem. Invariant risk minimization (IRM) is a popular method to generalize the model. [4] The idea is to add a penalty term into the loss function specified as

$$\min_{\Phi:\boldsymbol{X}\to\boldsymbol{y}} \sum_{e\in E} R^e(\Phi) + \lambda \cdot \left\|\nabla_{\omega|\omega=\omega_0} R^e(\omega\cdot\Phi)\right\|^2 \tag{3}$$

where Φ is a predictor, ω is a scalar and λ is a regularizer. It has been proved that IRM can effectively generalize the model. In addition, this algorithm is quite easy to implement using the automatic derivation function.

For simplicity, we use batch samples to compute the unbiased estimaton of the penalty term, which is specified as

$$\sum_{k=1}^{b} \left[\nabla_{\omega} L(\omega \boldsymbol{X}_i \Phi, \boldsymbol{y}_i) \cdot \nabla_{\omega} L(\omega \boldsymbol{X}_j \Phi, \boldsymbol{y}_j) \right]_{\omega=1.0} \tag{4}$$

where b is the batch size, $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$ are the i-th and j-th samples in the batch respectively, and L is the loss function.

VI. Experiment Result

We make a little modification to the dataset to support data augmentation. The training function is also updated to implement the IRM algorithm. Then an identical LSTM network is trained respectively on the augmented dataset, with the IRM algorithm, and with two methods combined together. The experiment result is reported in the following table.

TABLE V
IMPROVED RESULT

Group	Set	Loss	Accuracy
Augmented	Train	0.0041	100.00%
	Test	1.4249	97.53%
Generalized	Train	0.0134	100.00%
	Test	1.5357	97.07%
Combined	Train	0.0042	100.00%
	Test	1.3130	97.64%

We can see that all the methods improve the performance to some extent, where combining the two methods leads to the best result, which improves the accuracy by approximately 1%. The training curve is also shown in the following figure.

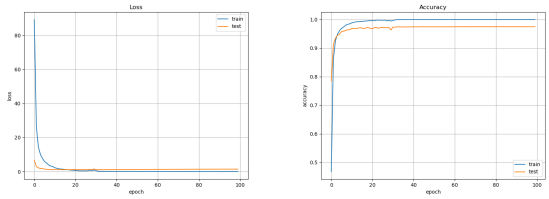

Fig. 7. data augmentation

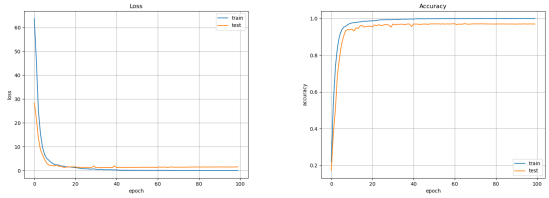

Fig. 8. model generalization

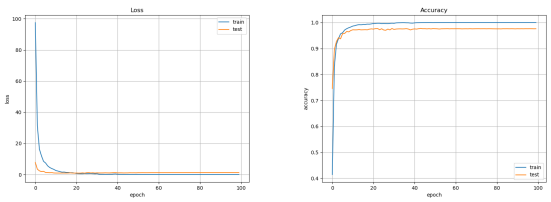

Fig. 9. method combination

From the figure we can conclude that data augmentation leads to faster convergence and promotes overfitting, while the IRM algorithm leads to slower convergence and suppresses overfitting. Combining the two methods makes a compromise and achieves the best result. The detailed accuracy of each class is shown in the following table.

TABLE VI
DETAILED ACCURACY

Group	Unbalanced	Augmented	Generalized	Combined
#0	97.45%	97.76%	97.65%	98.27%
#1	97.53%	99.03%	97.89%	98.50%
#2	94.67%	95.45%	95.16%	95.35%
#3	92.87%	95.64%	94.16%	95.94%
#4	92.16%	95.21%	92.67%	96.33%
#5	98.54%	98.54%	98.99%	98.32%
#6	98.75%	98.33%	99.06%	98.54%
#7	98.74%	98.54%	98.25%	98.35%
#8	98.05%	98.25%	98.67%	98.56%
#9	98.61%	98.51%	98.41%	98.32%
Total	96.72%	97.53%	97.07%	97.64%

We can see that the two methods effectively improve the model, where the images with label $(0, 1, 2, 3, 4)$ are predicted more accurately, which verifies our conclusion.

VII. Experiment Reflection

In this experiment, we reveal that RNN models can also solve the image classification tasks. In visual tasks, carefully designed CNN models tend to achieve the best performance. Compared with CNN models, RNN models are more universal for sequential data such as text and speech, which is a great advantage. However, RNN models are more difficult to train. Although RNN models apply strong parameter sharing, the recursive computation brings much higher time complexity. Nowadays, transformers with attention mechanism have become increasingly popular, which indicates that RNN models are not the only choice for universal tasks. [5]

Appendix
Training Parameter

All the datasets are loaded with 1024 batch size and random shuffling enabled. We use cross entropy loss function and Adam optimizer with learning rate 10^{-3}. All the RNN models are trained for 100 epochs and trained and tested on a single NVIDIA A100 80G Tensor Core GPU.

References

[1] Elman J L. Finding structure in time[J]. Cognitive science, 1990, 14(2): 179-211.

[2] Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.

[3] Vasconcelos C N, Vasconcelos B N. Convolutional neural network committees for melanoma classification with classical and expert knowledge based image transforms data augmentation[J]. arXiv preprint arXiv:1702.07025, 2017.

[4] Arjovsky M, Bottou L, Gulrajani I, et al. Invariant risk minimization[J]. arXiv preprint arXiv:1907.02893, 2019.

[5] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. Advances in neural information processing systems, 2017, 30.