LSTM: An Image Classification Model Based on Fashion-MNIST Dataset

Kexin Zhang,

Research School of Computer Science, Australian National University
Kexin Zhang, U6342657@anu.edu.au

Abstract. The application of Neural Network (NN) in image classification has received much attention in recent years. While most previous works focus on the application of Convolutional Neural Network (CNN), this research aims to develop and optimize a Recurrent Neural Network (RNN) model to finish the classification task on Fashion-MNIST dataset. The model is expected to be both relatively accurate and time-saving. I develop the model with Long-Short Term Memory (LSTM) technique to reduce the risk of gradient vanishing that traditional RNN faces. I use fine tuning and cross validation to optimize the model, and I also test Heuristic Pattern Reduction method and Network Pruning method. The model can achieve a high accuracy at more than 89% with acceptable time consumption. Compared with other models, this result is relatively good, but there is still room for improvement. PyTorch framework is used for experimentation.

Keywords: Long-Short Term Memory, Recurrent Neural Network, Network Pruning, Heuristic Pattern Reduction, image classification, machine learning

1 Introduction

1.1 Introduction of Research

In recent years, neural network has been proposed as an approach to develop high performance image classification model. Generally, Convolutional Neural Network (CNN) is considered as the first choice to do the image classification, but I test another Deep Learning method Recurrent Neural Network (RNN) in this research. To overcome the weakness of traditional RNN, I use the Long-Short Term Memory (LSTM) technique to build the model. I optimize the model by fine tuning, cross validation, Network Pruning and Heuristic Pattern Reduction method. Finally, the accuracy of LSTM model can reach 89.94% with acceptable time consumption.

2.1 Introduction of Fashion-MNIST Dataset

Fashion-MNIST (F-MNIST) is a relatively new dataset released by Zolanda Research (2017). It consists of 28 x 28 pixels grayscale images of 70,000 fashion products, and it has 10 categories with 7,000 images per category. The training set has 60,000 images and the test set has 10,000 images. **Table 1** presents the class names and their corresponding labels.

There are two reasons why I choose this dataset. The first one is that it has a good level of complexity, and some classifiers can hardly achieve a perfect score on it, so there is still lots of room for optimization. The second reason is that some researches based on it use relatively new methods, so we have a large number of modern results to compare with.

Table 1. Image Classes and labels

Class Name	Label
T-shirt/Top	0
Trouser	1
Pullover	2
Dress	3
Coat	4
Sandals	5
Shirt	6
Sneaker	7
Bag	8
Ankle boots	9

2 Methods

2.1 Model

Normally, Convolutional Neural Network (CNN) is considered to be the first choice on image classification, because it learns to recognize components of images firstly and then learn the larger structures, which can just meet the needs of image classification. Unlikely to CNN, RNN learns to recognize image features across time. Although RNN can be used for image classification theoretically, only a few researches about RNN image classifier can be found. Therefore, I choose to build RNN model in my research.

However, traditional RNN is sometimes unstable in practice, especially when backpropagating gradients through long time windows, which may cause gradient explosion or vanishing. Therefore, I try to use LSTM method to improve the model. It can avoid this problem by replacing the hidden units with memory cells. These memory cells can store information until it is relevant, so that the model dose not have to exploit long rang dependencies in the data. **Figure 1** presents the structure of LSTM memory cell (Graves, 2013). There are input, output and forget gates which controls information into or out of the memory cell.

At the beginning of the research, I build a traditional RNN model and an improved RNN model using LSTM technique. I test these two models respectively and try to find which one is better.

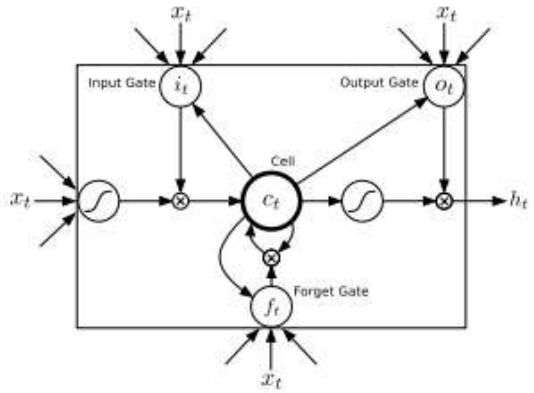


Figure 1. LSTM memory cell structure

2.2 Pre-processing

The built-in function transforms() in torchvision package is used to pre-process the input data. I use the toTensor() function to convert images into tensors, so they can be used as the input of the Neural Network model. I also normalize tensor images with mean and standard deviation both of which are set manually.

$$X' = \frac{X - Mean}{StandardDeviation}$$

2.3 Validation Set

I split the official training set into a new training set and a validation set. The validation set is used to detect overfitting and avoid it by stopping training early.

2.4 Fine Tuning

The validation set is also used in the adjustment of the model structure. I have tested models with different values of learning rate (from 0.001 to 0.1), different epoch numbers (from 5 to 50), different numbers of hidden layers (from 2 to 6) and hidden memory blocks (from 5 to 50). I use different loss functions and optimizers. These models are evaluated in terms of their training time consumption and predictive accuracy.

2.5 Heuristic Pattern Reduction

Gedeon and Bowden (1992) indicate that reducing the size of the training pattern set can sometimes improve the performance of classifier on the validation set. Normally, people tend to use more data for training, which may lead to a more accurate model. However, it is not a good choice to use too many data sometimes, because it may result in overtraining and overfitting. The overfitting model takes a long time for training but performs worse. Heuristic Pattern Reduction (HPR) method encourages us to find a better training set size through experiments.

In my previous paper, I focus on the influence of HPR on predictive accuracy. This time, I will evaluate models trained with different sizes (20%, 40%, 60%, 80%, 100%, split randomly) of training sets in terms of predictive accuracy and time consumption in order to build a relatively accurate and efficient model.

2.6 Network Pruning

In this part, I train the model and record weights of all connections. Then, I filter out connections of which the weight is smaller than the threshold which is set manually. The weights of these connections will not be updated in the retraining process. Therefore, the number of connections which contribute less is reduced, and the complexity of the model is also reduced. This method used here is similar with the method indicated by Louizos, Welling and P.kingma (2017) that the network can be pruned by encouraging weights to become exactly zero.

3 Results and Discussion

3.1 Model

I test the traditional RNN model and the LSTM model respectively. The result shows that LSTM model performs better than the RNN model with a higher accuracy at 80.20%, while the RNN model can only achieve a 74.52% accuracy. Therefore, I will use the LSTM for further experiment.

3.2 Pre-Processing

Pre-processing will not contribute to the model complexity and time consumption, but the input data quality can influence the predictive accuracy. Normalization can improve the accuracy of the LSTM model by 1% to 2% by providing high-quality input data.

3.3 Validation Set

I use the validation dataset to detect overfitting here. When the model is kept training, the decrease of predictive accuracy on validation set can be regarded as a sign of overfitting. I do not observe overfitting in my experiment. The reason is probably that the model is stopped training before being over-trained. However, this method can be useful in other researches.

3.4 Network Pruning

Firstly, I test some common optimizers, including Adam(), Adamax(), SGD() and RMSprop() and loss functions, including CrossEntroyLoss(), nll_loss() and margin_ranking_loss(). The results show that the accuracy of model using Adam () optimizer and CrossEntroyLoss() loss function is at least 2.5% higher than that of other models.

I test different values of epochs number ranging from 5 to 50 with an interval of 5. **Figure 2** presents the relation between the predictive accuracy on the validation set and the number of epochs. As can be seen in the figure, when the epoch number is larger than 20, the accuracy can hardly increase, but the model will be much more time-consuming. Therefore, I set epoch number to 20.

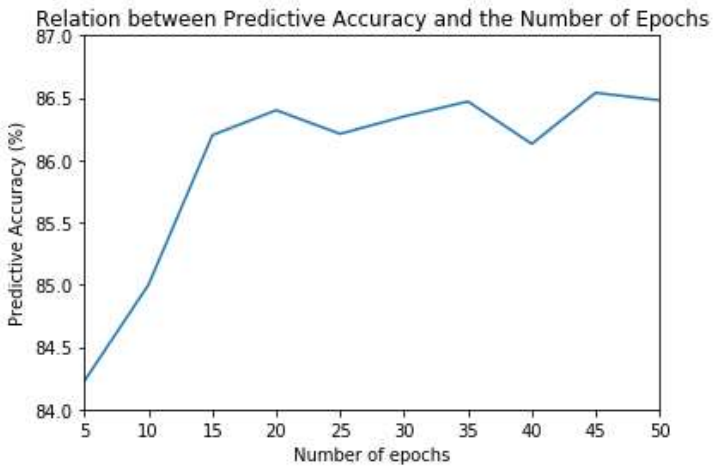


Figure 2. Relation between Predictive Accuracy and the Number of Epochs

I test different values of memory blocks number ranging from 5 to 50 with an interval of 5. **Figure 3** presents the relation between the predictive accuracy on the validation set and the number of memory blocks. The accuracy nearly stops increasing when the number of memory blocks is larger than 35, but the training time consumption of model with 50 blocks is nearly twice of that of model with 35 blocks. Therefore, I set the number of memory blocks to 35.

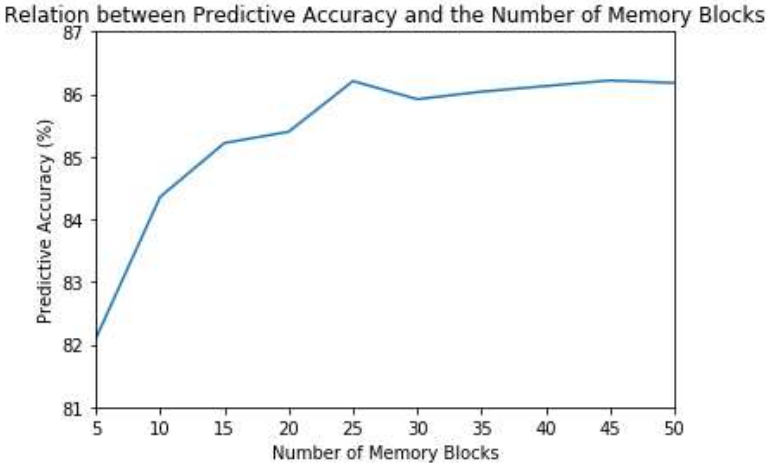


Figure 3. Relation between Accuracy and the Number of Memory Blocks

I use a two-hidden-layer LSTM model because it has similar performance with models with more hidden layers, which are much more complex and time-consuming. Four-hidden-layer LSTM takes a training time which is about 2.5 times as much as the two-hidden-layer one's. I choose a 0.001 learning rate because when the learning rate is larger than 0.001, the accuracy will decrease; when it is smaller than 0.001, the accuracy will no longer increase. Therefore, I set the learning rate at 0.001.

After fine tuning to parameters, the model has a high accuracy at about 87.61%. **Figure 4** presents the confusion matrix of the prediction result.

```
Confusion Matrix:
     0    1    2    3    4    5    6    7    8    9
0  857    1   12   48    5    1   67    1    8    0
1    5  966    3   22    1    0    2    0    1    0
2   18    2  854   22   38    1   60    0    5    0
3   23   10   11  917   17    0   20    0    2    0
4    2    1  152   61  729    1   54    0    0    0
5    0    1    0    0    0  959    0   25    1   14
6  168    3  109   47   53    0  613    0    7    0
7    0    0    0    0    0   13    0  914    0   73
8    3    0    3    8    2    4    5    4  971    0
9    0    0    1    0    0    3    0   15    0  981

Test set: Accuracy: 8761/10000 (87.61%)
```

Figure 4. Confusion Matrix of Prediction Result Given by the Model after Fine Tuning

3.5 Heuristic Pattern Reduction

Table 2 indicates the results of the application of Heuristic Pattern Reduction methods on the LSTM model. As can be seen in the **Table 2** that when use 60% training patterns to train the classifier, its predictive accuracy on the test set is close to that of classifier trained by all data. In additional to that, training with 60% patterns can save about 40% training time. Taking both the classification performance and time consumption into account, it would be a good choice to train the classifier with 60% to 80% training patterns. The model will be both relatively accurate and efficient.

As mentioned above, no sign of overfitting is detected by now, so it is unnecessary to reduce the training set size. However, taking the time consumption into account, training with 80% data can be a good choice.

Table 2. Heuristic Pattern Reduction Result

Percentage of Training Set	Training Pattern Number	Accuracy	Time Consumption
100%	60,000	87.57%	About 13.0 seconds for each epoch
80%	48,000	87.41%	About 10.1 seconds for each epoch
60%	36,000	87.14%	About 8.0 seconds for each epoch
40%	24,000	86.19%	About 5.2 seconds for each epoch
20%	12,000	84.10%	About 2.7 seconds for each epoch

* The time consumption data is related with the platform used for experimentation.
* I repeat each experiment for 5 times, and the data of accuracy and time consumption is the average of results from 5 experiments.

3.6 Network Pruning

Figure 5 presents the connection numbers of different weights between units of different layers. As can be seen in **Figure 5**, the number of connections of which weight is equal to or near to 0 is large. Therefore, we need to prune the network to reduce unnecessary connections.

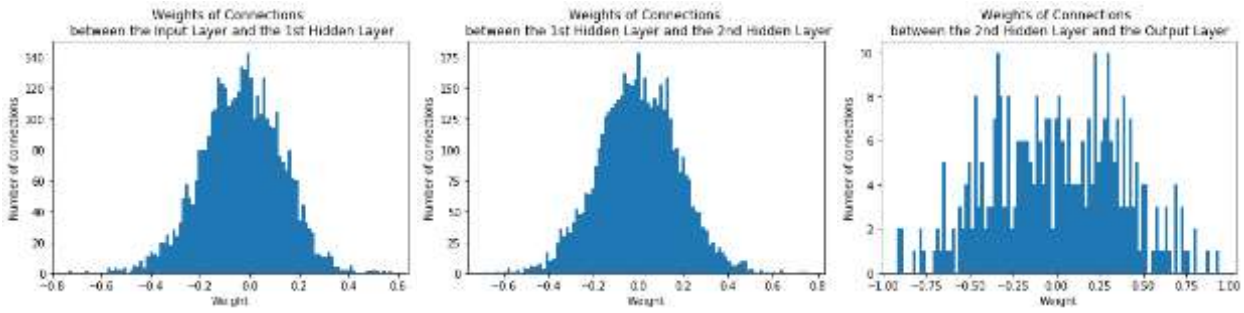


Figure 5. Weights of Connections between Different Layers

Figure 6 presents the connection numbers of different weights between units of different layers. Here I set the threshold to 0.05. It can be seen from the figure that connections of which weight is smaller than the threshold are set to zero. As a result, these connections will not contribute to the prediction any more.

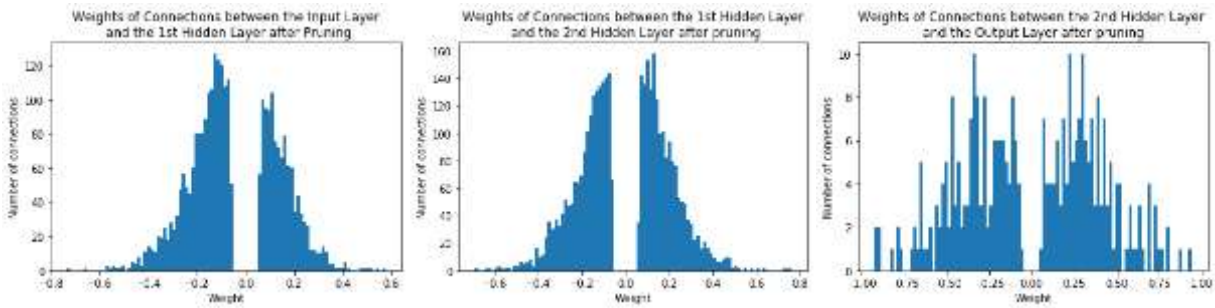


Figure 6. Weights of Connections between Different Layers after Pruning

Then, retrain the model. The weights of connections which are filtered out in last step will not be updated any more. However, this algorithm can only find out the connections of which weight is smaller than the threshold, set their weights to zero and stop updating them in the retraining process, but cannot cut the connection. This should be improved in future works.

Table 3 illustrates the predictive accuracy of model after pruning and model after retraining. It can be inferred that the model is likely to get a better performance when the threshold is relatively small.

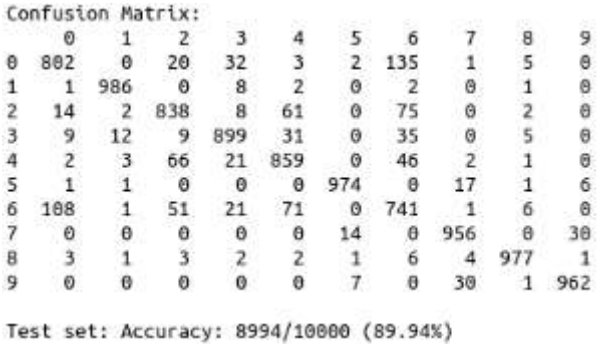
Table 3. Predictive Accuracy with Different Threshold Values

Threshold	Accuracy	
	Model after Pruning	Model after Pruning and Retraining
0.03	86.91%	88.46%
0.06	87.02%	88.24%
0.09	86.71%	88.35%
0.12	84.16%	87.44%
0.15	83.17%	86.57%
0.18	80.85%	84.23%

3.6 Final Model

I combine all methods mentioned above and build the final model. The final model is a two-hidden-layer LSTM model with 35 memory blocks. It uses Adam () optimizer and CrossEntroyLoss() loss function. It is trained with 48,000 training patterns for 20 epochs with a 0.001 learning rate. I applied Network Pruning on it with a 0.03 threshold. After retraining, its predictive accuracy on test set can reach 89.94%. And the average time consumption for one epoch is 10.2 seconds, which is acceptable. **Figure 7** presents the confusion matrix of the prediction result. It can be calculated from the confusion matrix that precision score of it is 89.74%.

However, the the standards of time consumption and accuracy are different in different cases. For example, a more complex model with four hidden layers can obtain 91% accuracy and 89.6% precision if a larger time consumption is acceptable.



Confusion Matrix:

	0	1	2	3	4	5	6	7	8	9
0	802	0	20	32	3	2	135	1	5	0
1	1	986	0	8	2	0	2	0	1	0
2	14	2	838	8	61	0	75	0	2	0
3	9	12	9	899	31	0	35	0	5	0
4	2	3	66	21	859	0	46	2	1	0
5	1	1	0	0	0	974	0	17	1	6
6	108	1	51	21	71	0	741	1	6	0
7	0	0	0	0	0	14	0	956	0	30
8	3	1	3	2	2	1	6	4	977	1
9	0	0	0	0	0	7	0	30	1	962

Test set: Accuracy: 8994/10000 (89.94%)

Figure 7. Confusion Matrix of Prediction Result Given by the Final Model

3.7 Comparison

Table 4 illustrates the test accuracy of different models. Deep Learning models generally perform better, and models using multiple methods are more likely to obtain a high accuracy. Unfortunately, few papers indicate the training time consumption of their models, so I cannot compare these models in term of it.

Table 4. Comparison with Other Models

Models (Methods)	Test Accuracy
Three-layer Neural Network	87.23%
Support Vector Classifier with rbf kernel	89.70%
Evolutionary Deep Learning Framework	90.60%
CNN using SVM activation function	90.72%
CNN using Softmax activation function	91.86%
CNN with Batchnor-malization	92.22%
CNN with Batchnor-malization and Residual skip	92.54%

* Data is from works of Bhatnagar, Ghosal and Kolekar, M. (2017) and Agarap (2017) except data of three-layer NN.
* Data of three-layer NN is from previous experiment.

4 Conclusion and Future Work

In this research, I develop a LSTM model for image classification on the F-MNIST Dataset. I test several methods to reduce the time consumption and increase the predictive accuracy of the model. Pre-processing can increase the score by providing high-quality input data. Cross Validation detects and prevents overfitting and the decrease of score caused by overfitting. Fine Tuning helps to improve the structure of the model, and it can help to increase the score as well as reduce training time consumption sometimes. Heuristic Pattern Reduction methods reduces the training time, and in some cases, it can also increase the score. Network Pruning is one of the most significant challenge in the experiment. Although the algorithm still needs further improvement, the scores of models after pruning and retraining are indeed increasing. In the end, I develop a final model which achieves relatively high accuracy and precision score, and it is also time-saving.

As I mentioned before, there is still a lot of space for improvement. Firstly, the HPR method can be improved by selecting training patterns depending on error loss or other measures rather than selecting randomly. Secondly, as I mentioned in Section 3.6, the pruning algorithm should be improved by completely cutting unnecessary connections which would help to reduce the model complexity. Last but not the least, the state-of-art results indicate that the combination of multiple methods is more likely to obtain high scores. Some other attempts can be made, like using CNN and RNN together.

Reference

1. Agarap, F. A. (2017). An Architecture Combining Convolutional Neural Network (CNN) and Support Vector Machine (SVM) for Image Classification. arXiv preprint arXiv:1712.03541.
2. Bhatnagar, S., Ghosal, D., Kolekar, M. (2017). Classification of fashion article images using convolutional neural networks. *2017 Fourth International Conference on Image Information Processing (ICIIP)*, 1-6.
3. Gedeon, T. D., Bowden, T. G. (1992). Heuristic pattern reduction. In International Joint Conference on Neural Networks, 2, 449—453.
4. Graves, A., (2013). Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850
5. Louizos, C., Welling, M., Kingma D.P. (2017). Learning sparse neural networks through L_0 regularization. arXiv preprint arXiv:1712.01312.
6. Xiao, H., Rasul, K., Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747.