Image Classification of Fashion-MNIST Dataset Using Long Short-Term Memory Networks

Shuning Shen

Research School of Computer Science, Australian National University, Canberra
{Shuning Shen, U6342315}@anu.edu.com.au

Abstract. Image classification is a popular domain in machine learning in recent years which can be solved by deep learning methods. In this paper, I use the Long Short-Term Memory Networks (LSTMs) to build a model which can operate the image classification of Fashion-MNIST dataset [1]. The dataset includes 70000 grayscale images whose size is 28*28 [2]. The best accuracy of my model can reach 88.26% which is a little bit less than the result of another model [3]. The two main methods which are training pattern reduction and network pruning can influence the performances of models.

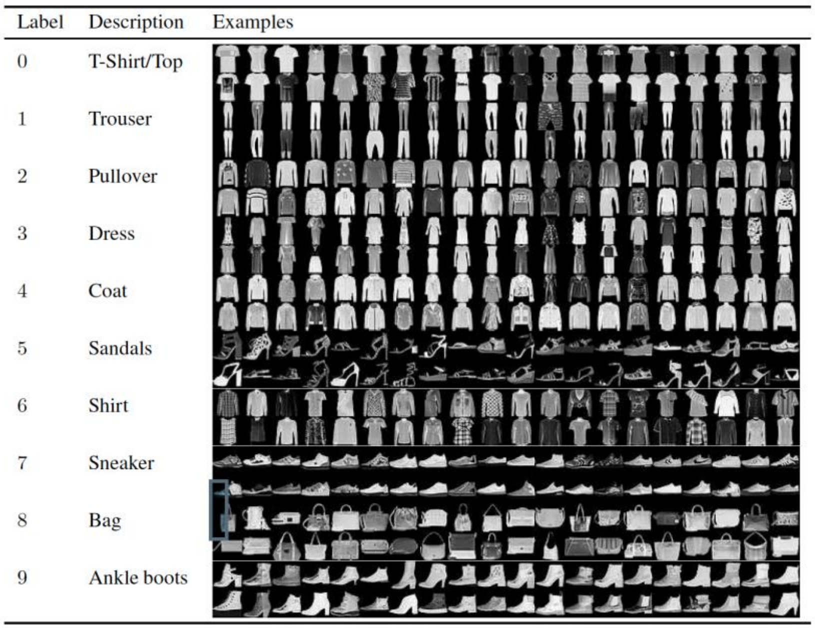
Keywords: Deep Learning, Image classification, Long Short-Term Memory Networks, Network Pruning, Training Pattern Reduction

1 Introduction

This paper indicates the work of Long Short-Term Memory Networks (LSTMs) on image classification of Fashion-MNIST dataset. Fashion-MNIST is a dataset recomposed from the product pictures of Zalando's website [1]. The dataset has 70000 grayscale images which divide into 60000 training patterns and 10000 testing patterns. The original size of pictures is 51*37 which have been resized to 28*28 [2]. The classes of Fashion-MNIST dataset are a little bit different from the MNIST dataset. There are 10 classes which are different types of clothes, bags and shoes. Figure 1 illustrates the class names and examples in Fashion-MNSIT dataset. I choose the Fashion-MNIST dataset instead of MNIST dataset because the Fashion-MNIST dataset is a relatively new dataset while MNIST dataset is applied by too many researchers. What's more, the Fashion-MNIST can be applied widely to discriminate different algorithms and classifiers [2].

In addition, it's known that image classification can be a difficult problem in computer vision. Recently, deep learning has scored tremendous achievements in computer vision, performing well on many image and speech recognition problems [3]. There are too many relevant works of image classification using the Conventional Neural Networks (CNNs) to obtain satisfied results, which inspired me to use the Long Short-Term Memory Networks (LSTMs) which is also a typical deep learning technique to build the Neural Networks model.

Table 1. Class Names and Examples in Fashion-MNIST Dataset [2]



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

2 Method

I use several methods aim to build a both accurate and less computational cost model. The methods are measured by testing accuracy, precision, recall and f1 score which are discussed in section 3.

2.1 LSTMs model

LSTMs are a special network of Recurrent Neural Networks (RNNs). The difference between traditional RNNs and LSTMs is that LSTMs can avoid the long-term dependency problem and work for either long or short time periods [4]. Besides, LSTMs can also reduce the problem of exploding gradient during backpropagation of traditional RNNs [4][5]. Figure 1 demonstrates the repeating module of LSTMs with 4 interacting layers. Between memory and past cells: c_t and c_{t-1}, LSTMs can present a linear dependence [4]. Moreover, LSTMs also has input, output and forget gates: i_t, o_t and f_t [4]. The calculate steps of LSTMs is below [4]:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1}) \quad (1)$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1}) \quad (2)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot tanh(W_{xc}x_t + W_{hc}h_{t-1}) \quad (3)$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}h_{t-1}) \quad (4)$$
$$h_t = o_t \odot tanh(c_t) \quad (5)$$

Thus, I choose the LSTMs to build both accurate and less expensive model. However, there are no time steps in the Fashion-MNIST dataset since it's not a time relevantly dataset. Therefore, I divide the size of pictures which make the height be time step and width be input. Besides, I use torchvision.transforms.ToTensor() function to normalize the input vector to range 0 to 1. The final LSTMs model use 6 layers which include 4 hidden layers. Additionally, the input layer has 28 neurons and the output layer has 10 neurons. The performances of LSTMs models are discussed in section 3.1.

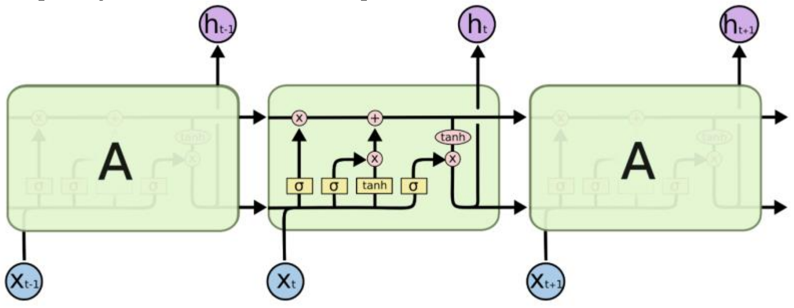


Fig. 1. The repeating module in a 4 layers LSTMs [5].

2.2 Optimizer and Activation Function

I select the torch.optim.Adam() be optimizer and torch.nn.LogSoftmax() be activation function to train the LSTMs model because they can perform better than other optimizer and activation functions in my model.

2.3 Training Pattern Reduction

According to the paper of Gedeon and Bowden, reducing the size of training pattern set can improve the performance of testing set [6]. Therefore, I randomly reduce the size of training pattern set from 60000 to 20000 to evaluate this technique with my model. The performances of different training pattern sets are discussed in section 3.2.

2.4 Network Pruning

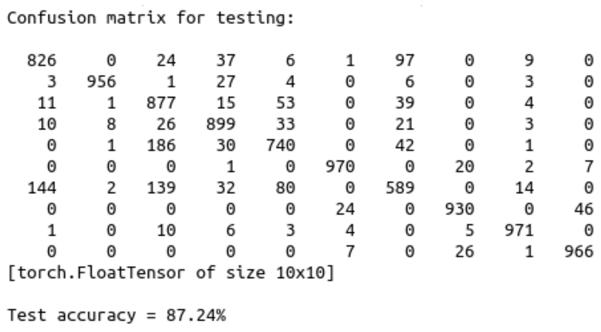
Network pruning can reduce computational cost while keeping the similar performance during training [7]. According to the paper of Rathi etc., the authors try to structure the pruning methodology compared to threshold [7]. They also briefly removing the connections between neurons whose weights less than threshold to reduce the computational cost of model [7]. Therefore, I set different value of threshold to find the weights of connections which are less than the value of threshold. Then train the model again and prune these connections temporarily by setting the weight be zero and not updating the weight of pruned connections. The results of different threshold network pruning model are discussed in section 3.3.

3 Result and Discussion

3.1 The First Training Result and Performances of LSTMs with Different Hidden Units

The test accuracy of my initial LSTMs model is 87.24%. Figure 2 illustrates the confusion matrix of the first model.

To determine the number of hidden layers and number of hidden neurons, I compare the performances of different number of hidden neurons and different number of hidden layers. Firstly, I change the parameter ‘num_layer’ in torch.nn.LSTM() be 1, 2 and 3 to choose the appropriate number of hidden layers. Besides, ‘num_layers = 1’ means 2 hidden layers, ‘num_layers = 2’ means 4 hidden layers, and ‘num_layers = 3’ means 6 hidden layers. Figure 3 demonstrates the confusion matrix of 4 hidden layer LSTMs. When the parameter ‘num_layers’ is 2, the LSTMs model perform best whose accuracy is 87.53% and the computational time is less than that of ‘num_layers = 3’. Secondly, I choose different size of hidden neurons and size of batch in each layer. When the size of hidden neurons is 64 and batch size is 100, the performance of LSTMs model is best. Therefore, the LSTMs model I use to evaluate the next methods has 4 hidden layers with 64 hidden neurons in each layer and whose batch size is 100.

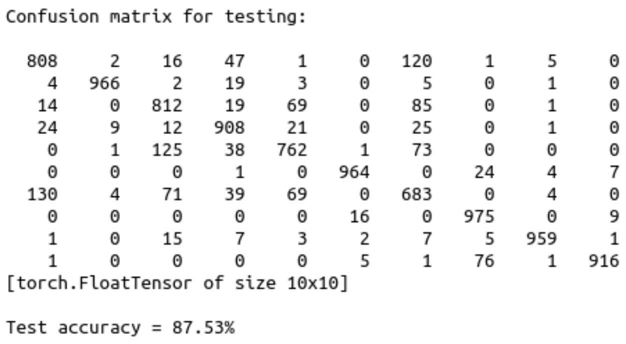


```
Confusion matrix for testing:

 826     0    24    37     6     1    97     0     9     0
   3   956     1    27     4     0     6     0     3     0
  11     1   877    15    53     0    39     0     4     0
  10     8    26   899    33     0    21     0     3     0
   0     1   186    30   740     0    42     0     1     0
   0     0     0     1     0   970     0    20     2     7
 144     2   139    32    80     0   589     0    14     0
   0     0     0     0     0    24     0   930     0    46
   1     0    10     6     3     4     0     5   971     0
   0     0     0     0     0     7     0    26     1   966
[torch.FloatTensor of size 10x10]

Test accuracy = 87.24%
```

Fig. 2. Confusion matrix of initial LSTMs model.



```
Confusion matrix for testing:

 808     2    16    47     1     0   120     1     5     0
   4   966     2    19     3     0     5     0     1     0
  14     0   812    19    69     0    85     0     1     0
  24     9    12   908    21     0    25     0     1     0
   0     1   125    38   762     1    73     0     0     0
   0     0     0     1     0   964     0    24     4     7
 130     4    71    39    69     0   683     0     4     0
   0     0     0     0     0    16     0   975     0     9
   1     0    15     7     3     2     7     5   959     1
   1     0     0     0     0     5     1    76     1   916
[torch.FloatTensor of size 10x10]

Test accuracy = 87.53%
```

Fig. 3. Confusion matrix of 4 hidden layers LSTMs model.

3.2 Performances of Different Size Training Pattern Sets

I randomly reduce the number of training patterns and select different numbers of epochs in order to evaluate the training pattern reduction method. According to figure 4, the value of loss is stable during number of batches increasing, so I choose the number of epochs be 10. Table 2 illustrates the accuracy of different size training pattern sets. According to the table, the performance of training pattern sets is worse when number of training patterns is reducing which means this method is lack of benefit. However, the difference of testing accuracy of different size training sets is not obvious. In my opinion, the worse performance of less size training pattern set may due to the structure of LSTMs, more training patterns can affect more on the error loss to get better result. Therefore, I choose the full training set which has 60000 training patterns to train the LSTMs model whose best testing accuracy is 88.26%.

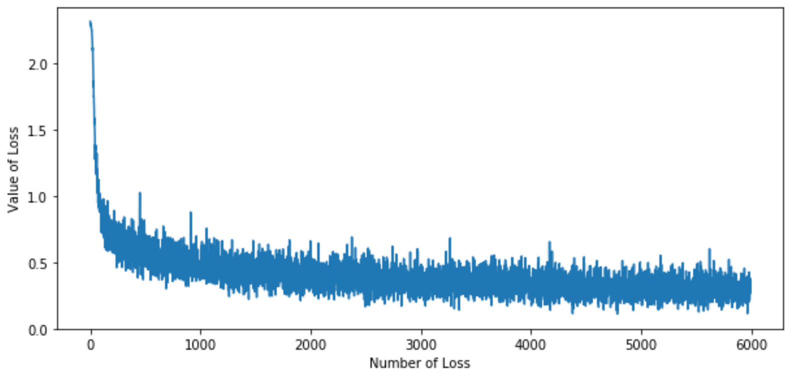


Fig. 4. Loss of 60000 Training Pattern Set.

Table 2. Accuracy of Different Size Training Pattern sets.

Number of Training Patterns	Testing Accuracy		
	1st	2nd	3rd
60000	87.53%	88.26%	88.12%
50000	86.73%	86.42%	86.54%
40000	86.21%	86.13%	86.27%
30000	85.08%	85.25%	85.13%
20000	84.40%	84.73%	84.52%

3.3 Performances of Network Pruning

To determine the standard threshold of pruning, I tried different value of thresholds from 0.5 to 0.05. Figure 5 illustrates the Kernel Density Estimate (KDE) plot of original weights and pruning weights when threshold is 0.05. It is noticed that the weights of connections which are less than 0.05 is temporarily removed in the retrain model. Table 3 indicates the testing accuracy and average updating time of weights of different pruning models. It should also be noticed that when the threshold more than 0.3, the accuracy of pruning networks can be much worse than the accuracy of original LSTMs. What' more, the average updating time of weights of pruning model is much more than that of original model. Although the accuracy of pruning model when threshold is 0.05 is 88.75% which is better than that of original model, the average updating time of pruning networks is 2 times more than that of normal LSTMs. As the aim of the pruning method is to reduce the computational cost of networks, my pruning algorithm didn't do very well. Thus, I choose to use the LSTMs model which is not pruned.

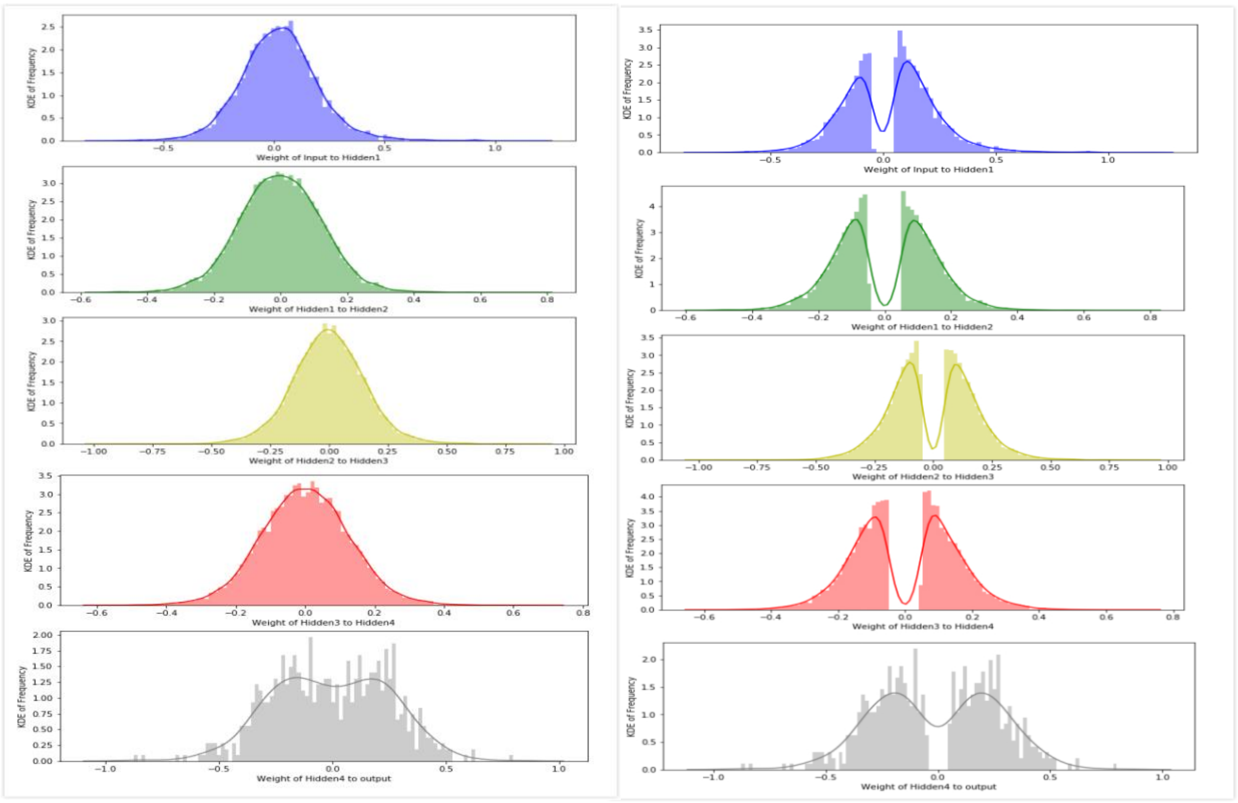


Fig. 5. The Kernel Density Estimate (KDE) plot of original weights and pruning weights (threshold = 0.05).

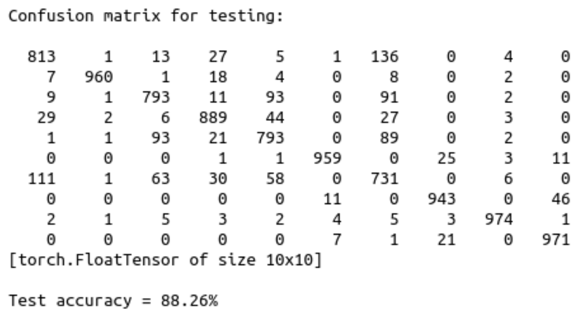
Table 3. Accuracy and average updating time of Different Pruning Threshold Model.

Threshold	Testing accuracy	Testing accuracy after pruning	Average updating time of weights(seconds)	Average updating time of weights after pruning(seconds)
0.05	87.35%	88.75%	0.0007	0.0018
0.1	87.84%	87.72%	0.0007	0.0026
0.2	87.74%	87.47%	0.0007	0.0029
0.3	87.56%	81.74%	0.0007	0.0030
0.4	87.96%	67.50%	0.0007	0.0030
0.5	88.17%	27.82%	0.0007	0.0031

3.4 Final Model and Comparison

According to section 3.1 to 3.3, the final LSTMs model is determined. There are 4 hidden layers in LSTMs model while each layer has 64 hidden neurons. The final LSTMs model also has 28 input neurons, 28 time steps and 10 output neurons. The batch size is 100 and number of epoch is 10. What's more, I choose the final model without network pruning because the computational cost is not reduced using my pruning algorithm. Therefore, the testing accuracy of my best performed model can reach 88.26%. Figure 6 and 7 demonstrate the confusion matrix, precision, recall and f1 score of my best performance model.

Table 3 illustrates the performance of Bhatnagar, Ghosal and Kolekar's model [3]. They use CNNs to solve the image classification problem. Compare to the authors' best performed model whose accuracy is 92.54%, the accuracy of my model is a little bit less than that of their model. According to figure 7 and table 4, the overall precision, recall and f1 score of my model is also less than those of their model. However, the precision of my model on classes trouser, sandals, sneaker, bag and ankle boots are as high as that of the authors' model. Therefore, in my opinion, my LSTMs model can make contribution to image classification problem with satisfied testing accuracy and classes precision.

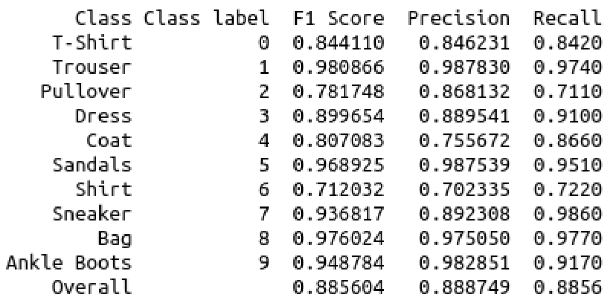


```
Confusion matrix for testing:

  813     1    13    27     5     1   136     0     4     0
    7   960     1    18     4     0     8     0     2     0
    9     1   793    11    93     0    91     0     2     0
   29     2     6   889    44     0    27     0     3     0
    1     1    93    21   793     0    89     0     2     0
    0     0     0     1     1   959     0    25     3    11
  111     1    63    30    58     0   731     0     6     0
    0     0     0     0     0    11     0   943     0    46
    2     1     5     3     2     4     5     3   974     1
    0     0     0     0     0     7     1    21     0   971
[torch.FloatTensor of size 10x10]

Test accuracy = 88.26%
```

Fig. 6. Confusion Matrix of best performed model.



```
      Class Class label  F1 Score  Precision  Recall
    T-Shirt           0  0.844110   0.846231  0.8420
    Trouser           1  0.980866   0.987830  0.9740
   Pullover           2  0.781748   0.868132  0.7110
      Dress           3  0.899654   0.889541  0.9100
       Coat           4  0.807083   0.755672  0.8660
    Sandals           5  0.968925   0.987539  0.9510
      Shirt           6  0.712032   0.702335  0.7220
    Sneaker           7  0.936817   0.892308  0.9860
        Bag           8  0.976024   0.975050  0.9770
Ankle Boots           9  0.948784   0.982851  0.9170
    Overall              0.885604   0.888749  0.8856
```

Fig. 7. Precision, recall and f1 score of best performed model.

Table 4. The result of Bhatnagar, Ghosal and Kolekar's model [3].

Class	Class Label	Precision	Recall	F1 Score
T-Shirt/Top	0	0.88	0.86	0.87
Trouser	1	0.99	0.98	0.99
Pullover	2	0.90	0.85	0.88
Dress	3	0.92	0.93	0.92
Coat	4	0.87	0.88	0.88
Sandals	5	0.99	0.99	0.99
Shirt	6	0.75	0.80	0.77
Sneaker	7	0.96	0.97	0.97
Bag	8	0.99	0.99	0.99
Ankle Boots	9	0.98	0.97	0.97
Overall	-	0.92	0.92	0.92

4 Conclusion and Future Work

In conclusion, using LSTMs to solve image classification is quite a challenge. After training and testing the dataset through different methods, my LSTMs model can fit the dataset whose best accuracy is 88.26%. However, the two main methods which are training pattern reduction and network pruning cannot contribute appropriately to enhance the performance of my model. Compare to the relevant paper, my model can still be improved to obtain better results. In the future, for one, I will improve my pruning algorithm to get both accurate and less computational cost model. For another, I will reduce the training pattern based on the error loss rather than randomly training reduction to choose more important training patterns.

References

1. A MNIST-like fashion product database, https://github.com/zalandoresearch/fashion-mnist
2. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. (2017)
3. Bhatnagar, S., Ghosal, D., Kolekar, M.H.: Classification of Fashion Article Images using Convolutional Neural Networks. In: Fourth IEEE International Conference on Image Information Processing (ICIIP), pp. 1--6. (2017)
4. Yao, K., Cohn, T., Vylomova, K., Duh, K., Dyer, C.: Depth-gated recurrent neural networks. (2015)
5. Understanding LSTM Networks, http://colah.github.io/posts/2015-08-Understanding-LSTMs/
6. Gedeon, T. D., Bowden, T. G.: Heuristic pattern reduction. In International Joint Conference on Neural Networks. vol. 2, pp. 449--453 (1992)
7. Rathi, N., Panda, P., Roy, K. STDP Based Pruning of Connections and Weight Quantization in Spiking Neural Networks for Energy-Efficient Recognition. In: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. (2018)