CS 584: Machine Learning Name: Muhao Chen
IIT ID Number: A20456889

Homework 4

1. IMDB Movie Review Sentiment Classification

1.1 Background Description

Nowadays, more and more people are commenting movies online, which reflects the quality of movies. However, different from scoring, only a small part of comments could not directly be used to judge the quality of movies, only if using algorithms to analyze the overall comments, we can classify the movies. Before that, we should firstly classify the comments sentiment into positive and negative classes.

1.2 Data Description

The data set is from keras, and is of 25,000 movies reviews from IMDB. All data is labeled by sentiment, including the classes of positive and negative. And we should input a parameter max_features, which means the most top words number, then return the matrix with the shape (25000,)

1.3 Problem

- The different length between different sentences. Using pad_sequences.
- The input should consider to convert the word to vector. Using Embedding.
- How to improve the performance of RNN. Using LSTM.

1.4 Data Preprocessing

We use the pad_sequences to ensure the length of the sentences, and we set the max_length of sentences is 70, which helps reshape the input to (25000,70). And we split the data into train_data and validation_data.
train_data_shape: (20000, 70)
validation_data_shape: (5000, 70)
test_data_shape: (25000, 70)

1.5 RNN Model

In this chapter, we use simple_rnn layer. Firstly, we should use embedding layer to make the words to a vector, and set the out_dim value of 120. Then add a simple_rnn layer with 120 out_dim. Finally, use dense layer to generate the two classes.

Model: "sequential"

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 70, 120)	2400000

simple_rnn (SimpleRNN)	(None, 120)	28920
dense (Dense)	(None, 1)	121

==========

Total params: 2,429,041
Trainable params: 2,429,041
Non-trainable params: 0

1.6 The Training Result of RNN Model

Train: We use adam optimizer, and binary_crossentropy to compile the model. And set the number of epochs to 10 and batch size to 256.

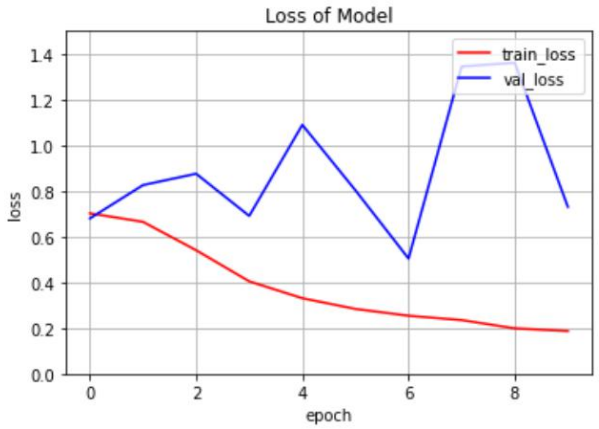
Result:

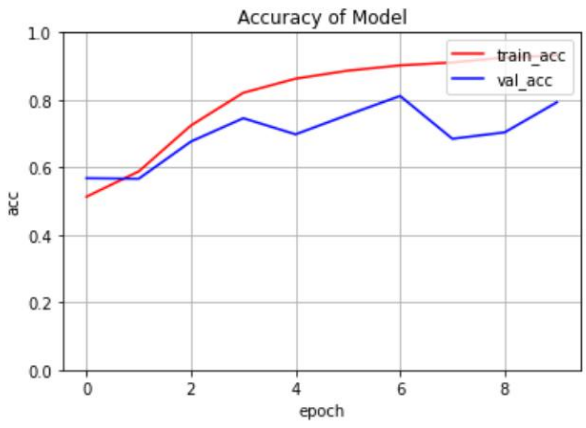
Test score of train dataset: 0.21158942580223083
Test accuracy of train dataset: 0.9236500263214111
Test score of validation dataset: 0.7303754687309265
Test accuracy of validation dataset: 0.7929999828338623

Analysis: It seems a not bad result.

1.7 The Loss, Accuracy Plot and Test Result.

Plot: It's obvious that the loss of training data and validation data is decreasing and the accuracy is increasing. And it also have a good performance on the test data.





Test score of test dataset: 0.7489978075027466
Test accuracy of test dataset: 0.7863600254058838

1.8 LSTM Model

In this chapter, we use LSTM layer. Firstly, we should use embedding layer to make the words to a vector, and set the out_dim value of 120. Then add a LSTM layer with

120 out_dim. Finally, use dense layer to generate the two classes.
Model: "sequential_1"

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 70, 120)	2400000
lstm (LSTM)	(None, 120)	115680
dense_1 (Dense)	(None, 1)	121

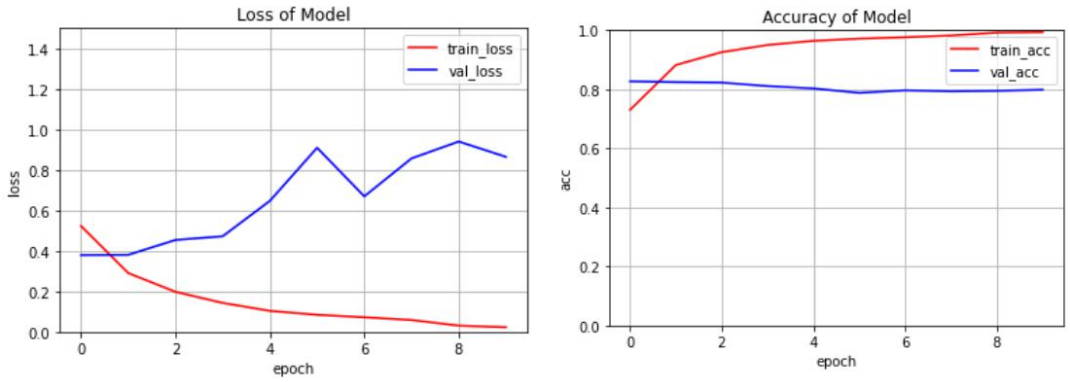
Total params: 2,515,801
Trainable params: 2,515,801
Non-trainable params: 0

1.9 The Training Result of LSTM Model

Train: We use adam optimizer, and binary_crossentropy to compile the model. And set the number of epochs to 10 and batch size to 256.
Result:
Test score of train dataset: 0.017682848498225212
Test accuracy of train dataset: 0.9960500001907349
Test score of validation dataset: 0.8644065260887146
Test accuracy of validation dataset: 0.7986000180244446
Analysis: It seems a better result.

1.10 The Loss, Accuracy Plot and Test Result.

Plot: It's obvious that the loss of training data data is decreasing and the accuracy is increasing. But the validation data performance is worse, which means it's over-fitting. And the test data performance is good.



Test score of test dataset: 0.8406696915626526
Test accuracy of test dataset: 0.7988399863243103

1.11 The Comparison Between RNN and LSTM

It's obvious that the performance of LSTM is better than than of RNN. And the speed of convergence of LSTM is quicker, which means more fast run to the peak performance. Because LSTM have the forget gate, which solve the problems of gradient vanishing.

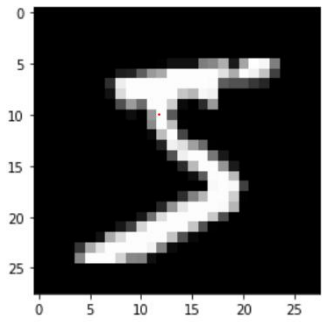
2. MNIST Classification

2.1 Background Description

Nowadays, digit recognition technology becomes more and more important in our daily life. For example, if we handle business in person, and need to write the SSN number by hand, and the bank or government could recognize the number without officers. In the same way, this technology could be applied in the bill, invoice and others.

2.2 Data Description

The data set is from keras, and is of 60,000 pictures from MNIST. All data is labeled by number, including the 0 to 9.



2.3 Problem

- The shape of one digit is (28,28), not word. Using input layer without embedding.
- How to visualize the loss and accuracy. Using callbacks to reconstruct return.
- How to improve the performance of RNN. Using LSTM and more dense layers.

2.4 Data Preprocessing

We split the training data to the training data and validation data. And divide the value of 255, which makes the digit range from 0-1.
train_data_shape: (50000, 28, 28)
validation_data_shape: (10000, 28, 28)
test_data_shape: (10000, 28, 28)

2.5 RNN Model

In this chapter, we use simple_rnn layer. Firstly, we should use input layer of (None,28,28). Then add a simple_rnn layer with 144 out_dim. Then add a dense layers with 72 out_dim. Finally, use dense layer to generate the 10 classes.

Model: "sequential_5"

Layer (type)	Output Shape	Param #
simple_rnn_4 (SimpleRNN)	(None, 144)	24912
dense_7 (Dense)	(None, 72)	10440
dense_8 (Dense)	(None, 10)	730

Total params: 36,082
Trainable params: 36,082
Non-trainable params: 0

2.6 The Training Result of RNN Model

Train: We use adam optimizer, and SparseCategoricalCrossentropy to compile the model. And set the number of epochs to 10 and batch size to 256.

Result:

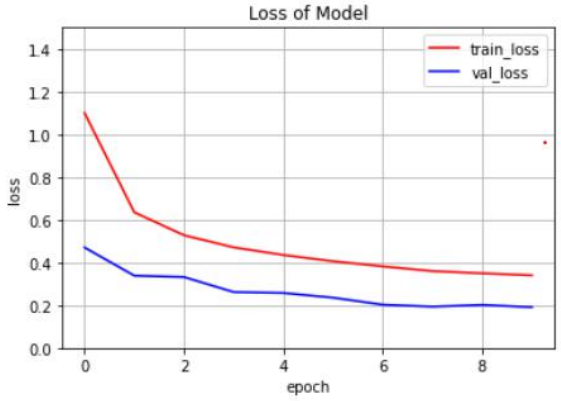
Test score of train dataset: 0.1809924989938736
Test accuracy of train dataset: 0.946399986743927
Test score of validation dataset: 0.19063450396060944
Test accuracy of validation dataset: 0.9455999732017517

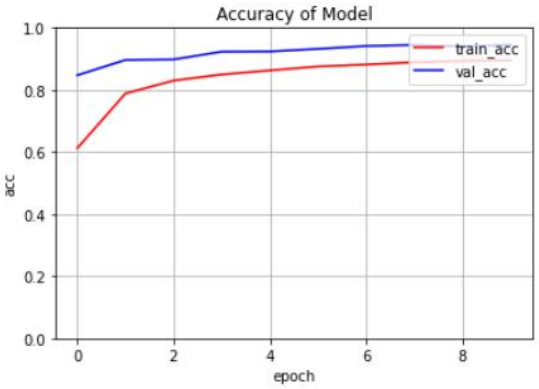
Analysis: It seems a not bad result.

2.7 The Loss, Accuracy Plot and Test Result.

Plot: It's obvious that the loss of training data and validation data is decreasing and the accuracy is increasing. And it also have a good performance on the test data.

Test score of test dataset: 0.17286671698093414
Test accuracy of test dataset: 0.9485999941825867





2.8 LSTM Model

In this chapter, we use LSTM layer. Firstly, we should use input layer of (None,28,28). Then add a LSTM layer with 144 out_dim. Then add a dense layers with 72 out_dim. Finally, use dense layer to generate the 10 classes.

Model: "sequential_6"

Layer (type)	Output Shape	Param #
lstm_1 (LSTM)	(None, 144)	99648
dense_9 (Dense)	(None, 72)	10440
dense_10 (Dense)	(None, 10)	730

Total params: 110,818
Trainable params: 110,818
Non-trainable params: 0

2.9 The Training Result of LSTM Model

Train: We use adam optimizer, and SparseCategoricalCrossentropy to compile the model. And set the number of epochs to 10 and batch size to 256.

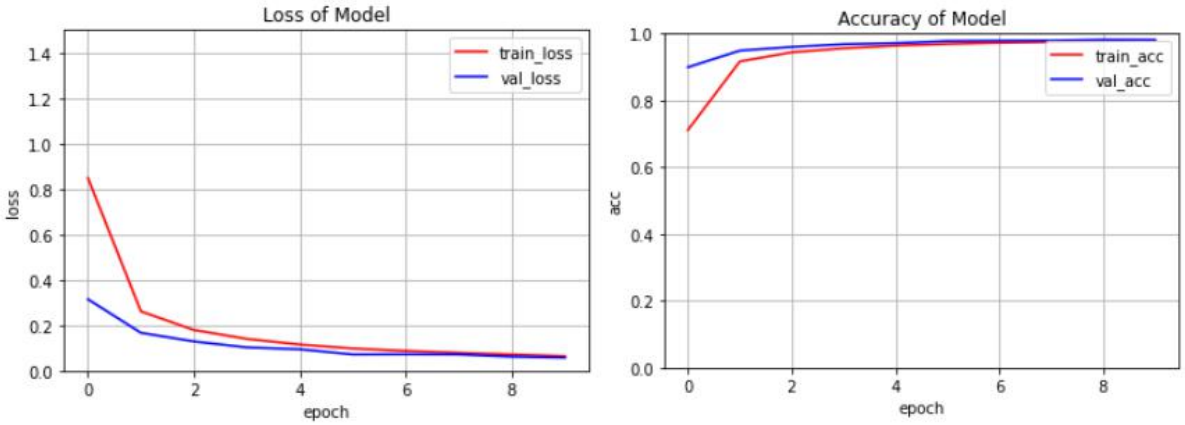
Result:

Test score of train dataset: 0.037766166031360626
Test accuracy of train dataset: 0.9883999824523926
Test score of validation dataset: 0.057117193937301636
Test accuracy of validation dataset: 0.9815999865531921

Analysis: It seems a good performance.

2.10 The Loss, Accuracy Plot and Test Result.

Plot: It's obvious that the loss of training data and validation data is decreasing and the accuracy is increasing. And it also have a good performance on the test data.
Test score of test dataset: 0.05263115093111992
Test accuracy of test dataset: 0.9828000068664551



2.11 The Comparison Between RNN and LSTM

It's obvious that LSTM have a better and most excellent performance. The accuracy of test dataset reach the 98.28%, which is larger than 94.86% while using RNN. The reason is that LSTM could think about the initial digit features, which is significant while judging the classification. For example, the LSTM could memorize the Part A, while RNN will nearly forget the Part A. And if the RNN only see the Part B and Part C. The digit of 5 also could be predicted by RNN to digit 2, which is incorrect.