DL hw2. 姓名:許庭恩 學號:111321514

IMDB:

USE PyTorch

Use Early-Stop to avoid overfitting and keep the best model. (patience=10)

lr: Learning rate、opt: Optimizer、seq_len: sequence length

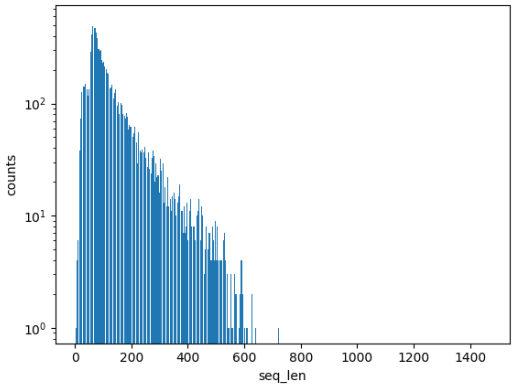
- **Dataset analysis:**

  IMDB has 50000 reviews, which labels only contain positive and negative, their counts are 25000 for each.

  So, we split data into train (70%), validation (10%), and test (20%). We rebuild data in other to make the model train more balance. Each of their positive/negative ratios is 1:1. That is the train (Pos:17500, Neg:17500), validation (Pos:2500, Neg:2500), and test (Pos:5000, Neg:5000). Of course the indexes were shuffled.

- **Dataset preprocess and feature engineering:**

  To remove unnecessary words, punctuation marks, and emojis. We write the function to clean these data. After the data was cleaned, we calculate some information about these processed reviews.




```
count    40000.000000
mean       125.416450
std         94.631719
min          4.000000
25%         67.000000
50%         93.000000
75%        153.000000
max       1465.000000
Name: len, dtype: float64
```

Max length	1465
Mean length	126
Median length	313.5

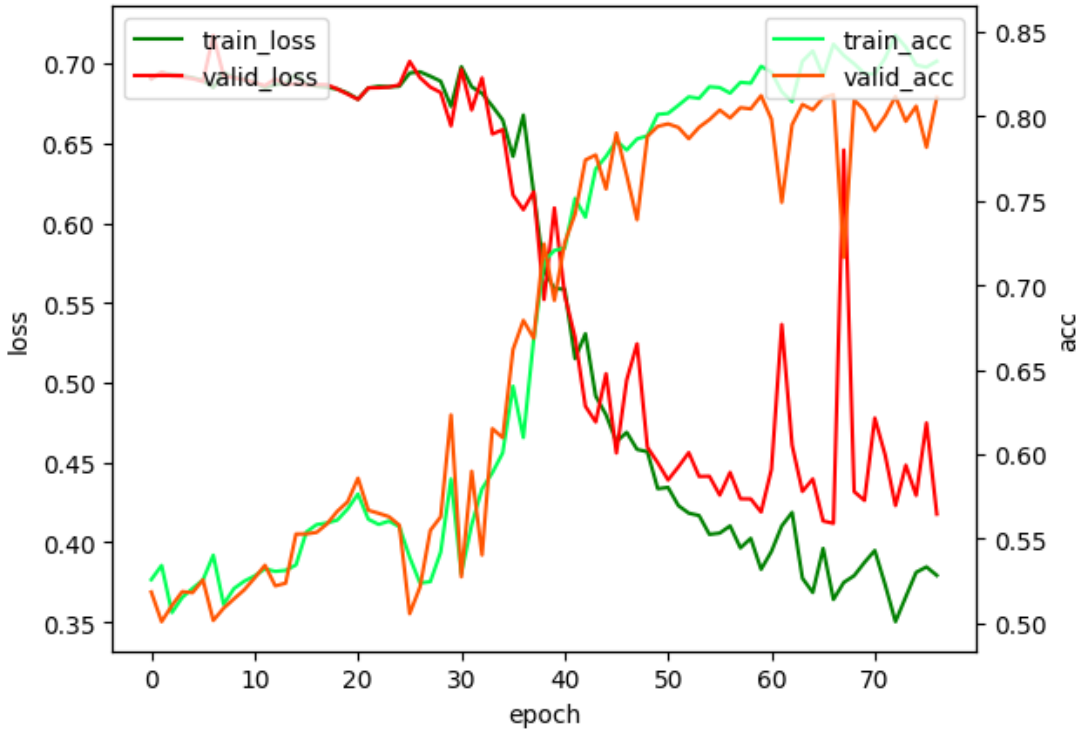
This information will help us to set the hyperparameter of sequence length. If we want to train a model, the input shape must be the same, so we need to pad or truncate the sequence length. It is very important because if the sequence length you set is too small it will cut a lot of words and too big will padding too many unnecessary tokens.

We use Glove's word2vec to help us transform the words to vectors, which embedding dimensions are 100.

- **Experiment**
  - **RNN architecture**

  LOSS: CrossEntropyLoss, Batch size=512

	RNN								
	M1	M2	M3	M4	M5	M6	M7	M8	M9
opt	Adam			Adam		Adam		SGD	RMSp
seq_len	126			150	180	126		126	
num_layers	1	2	3	2		2		2	
hidden size	64			64		128	196	128	
params	10k	19k	27k	19k	19k	62k	130k	62k	62k
Train_loss	0.627	0.658	0.677	0.665	0.647	0.364	0.693	0.639	0.668
Train_acc	67.59	61.01	57.9	59.41	63.64	84.28	52.81	50.48	58.21
Valid_loss	0.689	0.623	0.678	0.672	0.64	0.411	0.689	0.694	0.661
Valid_acc	67.37	67.55	58.4	60.28	63.66	81.29	57.25	50.24	61.37

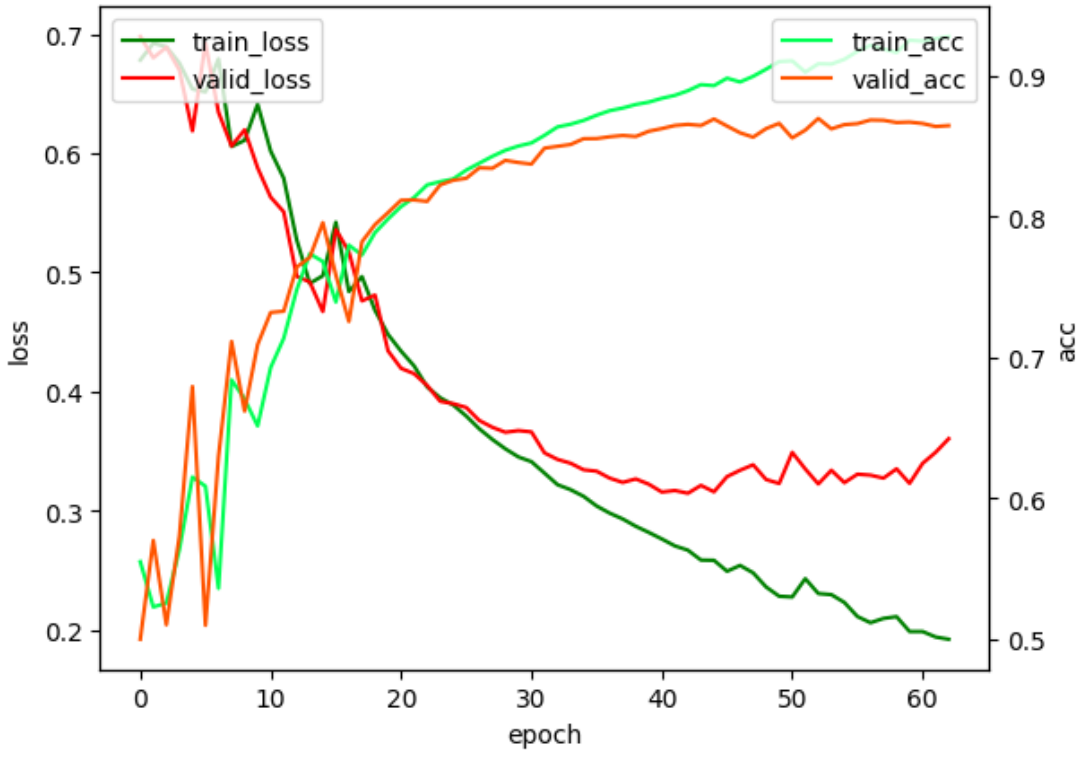


RNN-M6

➢ **LSTM architecture**

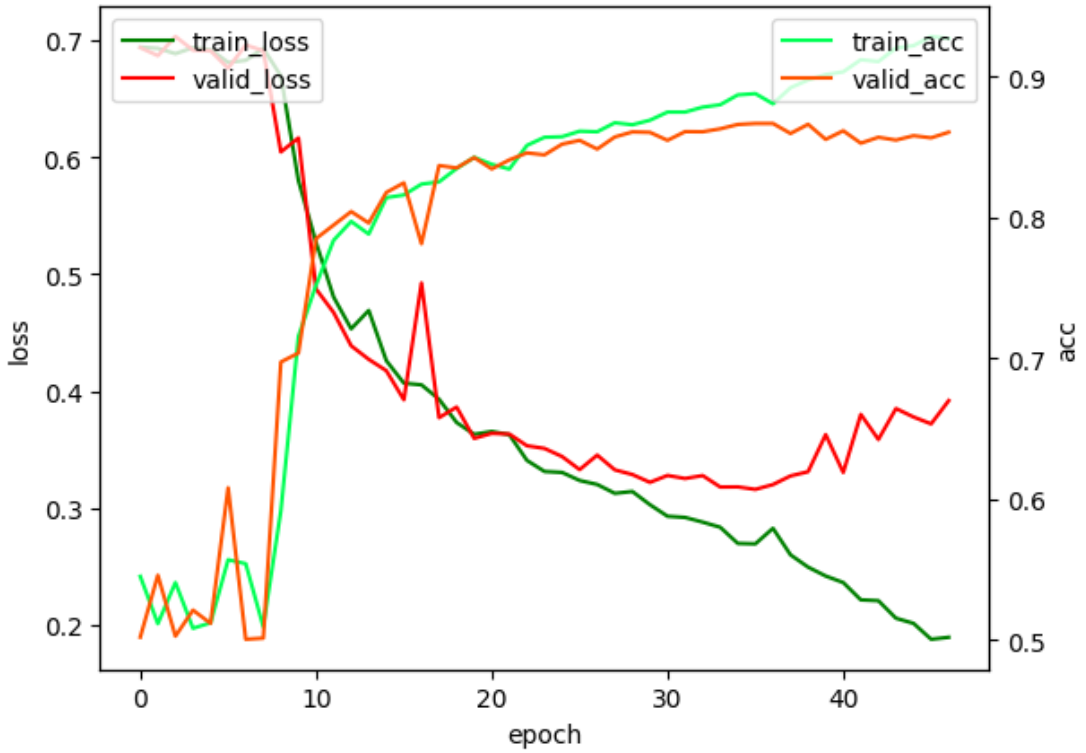
LOSS: CrossEntropyLoss, Batch size=512

	LSTM								
	M1	M2	M3	M4	M5	M6	M7	M8	M9
opt	Adam				Adam		Adam		
seq_len	126				150	180	126		
num_layers	1	2	3	4	3		3		
hidden size	64				64		128	196	256
params	42k	75k	109k	140k	109k	109k	382k	851k	1419k
Train_loss	0658	0.674	0.2087	0.647	0.678	0.657	0.27	0.661	0.677
Train_acc	58.21	55.68	92.03	61.68	56.52	62.57	88.88	59.17	56.01
Valid_loss	0.647	0.622	0.369	0.617	0.602	0.657	0.319	0.667	0.668
Valid_acc	60.49	66.7	86.55	69.67	70.86	62.23	86.32	54.86	60.85



LSTM-M3

	LSTM								
	M10	M11	M12	M13	M14	M15	M16	M17	M18
opt	Adam				Adam			SGD	RMSp
seq_len	126				110	120	130	126	
num_layers	3				3			3	
hidden size	64				64			64	
drop	0.2	0.3	0.4	0.5	0.3			0.3	
params	382k	382k	382k	382k	109k	109k	109k	109k	109k
Train_loss	0.667	0.214	0.259	0.289	0.264	0.213	0.249	0.693	0.307
Train_acc	55.36	91.64	89.43	88.04	88.99	91.55	89.66	49.89	87.11
Valid_loss	0.68	0.328	0.325	0.369	0.38	0.336	0.318	0.693	0.355
Valid_acc	57.73	87.25	87.12	85.36	85.37	86.85	86.62	50.06	85.74

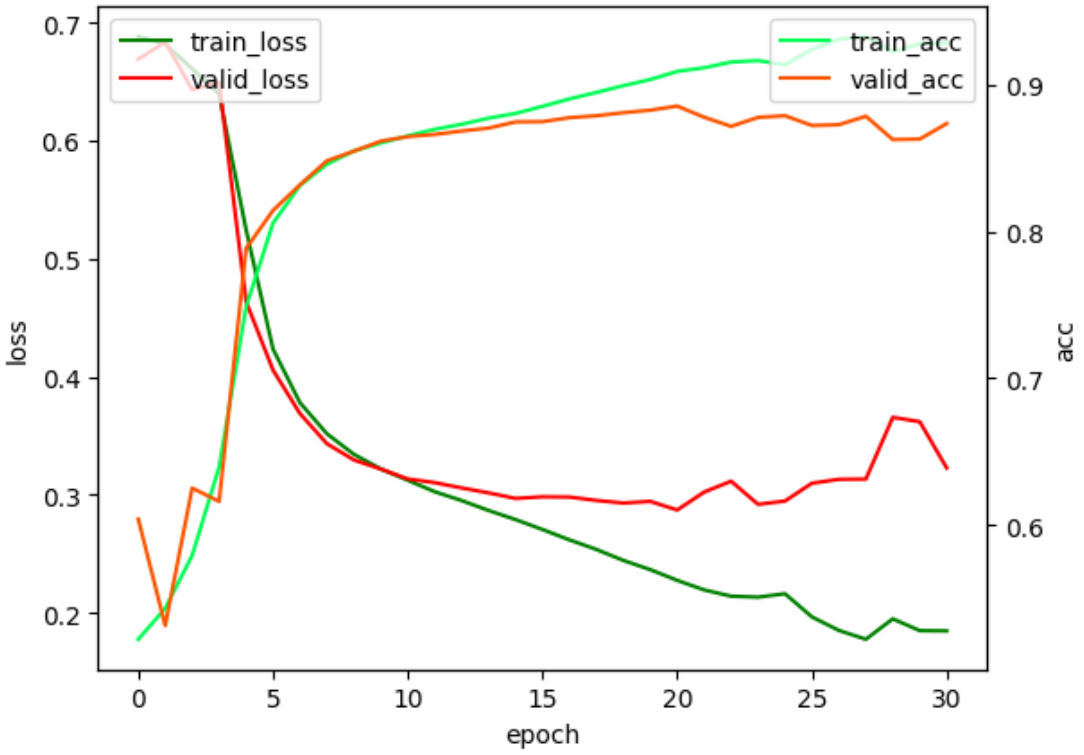


LSTM-M11

➢ **GRU architecture**

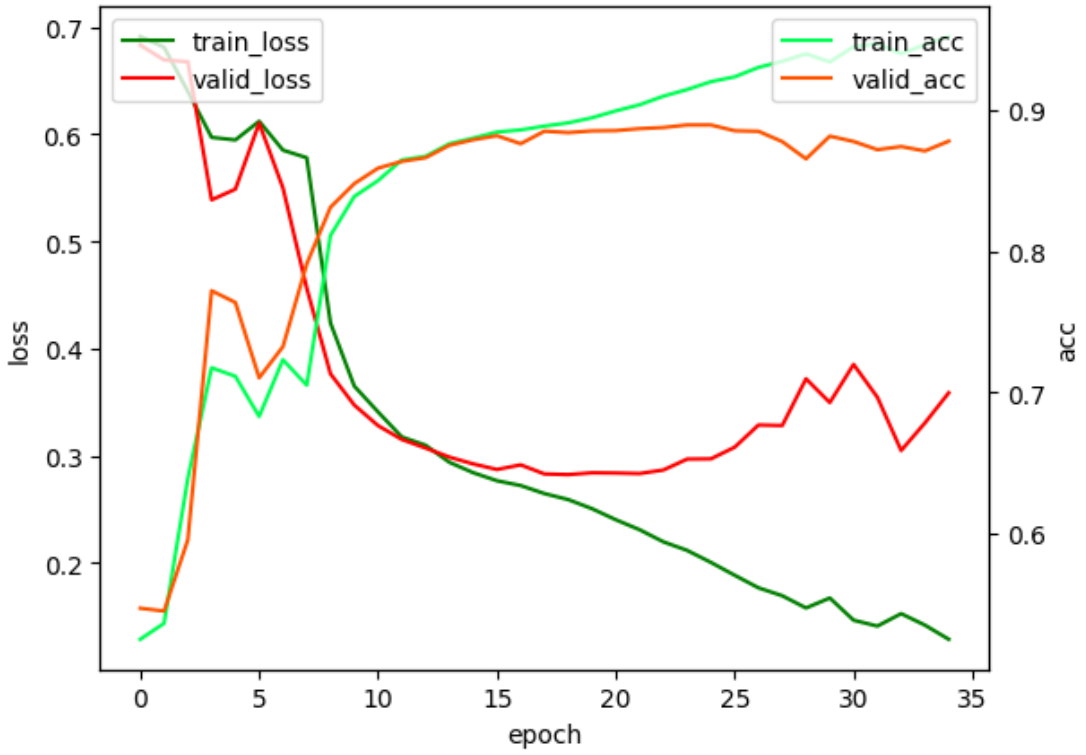
LOSS: CrossEntropyLoss Batch size=512

	GRU								
	M1	M2	M3	M4	M5	M6	M7	M8	M9
opt	Adam			Adam			Adam		
seq_len	126			150	180	200	180		
num_layers	1	2	3	2			2		
hidden size	64			64			128	196	256
params	32k	56k	81k	56k	56k	56k	187k	407k	670k
Train_loss	0.275	0.206	0.222	0.251	0.252	0.261	0.230	0.225	0.202
Train_acc	88.35	91.86	90.79	89.6	89.65	89.19	90.75	90.73	92.14
Valid_loss	0.322	0.337	0.359	0.33	0.282	0.299	0.293	0.3	0.335
Valid_acc	86.41	86.7	86.19	86.72	88.03	87.46	88.24	87.82	87.82



GRU-M7

	GRU								
	M10	M11	M12	M13	M14	M15	M16	M17	M18
opt	Adam			Adam			Adam		
seq_len	180			200	250	300	180		
num_layers	1	2	3	2			2		
hidden size	128			128			160	196	256
dropout	0.2	0.3	0.4	0.3			0.3		
params	187k	187k	187k	187k	187k	187k	280k	407k	670k
Train_loss	0.227	0.2	0.208	0.192	0.204	0.188	0.215	0.206	0.192
Train_acc	90.89	92.03	91.78	92.24	92.01	92.58	91.13	91.75	92.29
Valid_loss	0.287	0.297	0.301	0.3	0.29	0.32	0.296	0.306	0.329
Valid_acc	88.53	88.96	88.03	88.01	88.55	88.42	88.38	88.21	87.72



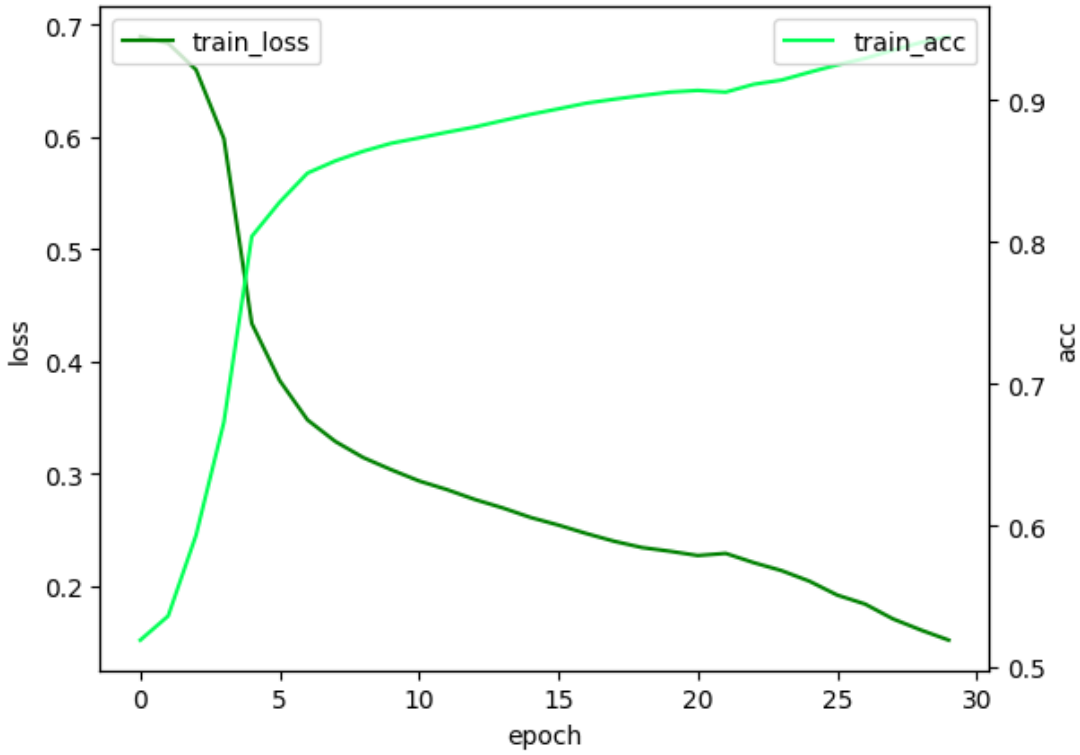
GRU-M11

According to our experiment result, we finally choose the GRU-M11 model as our best model.

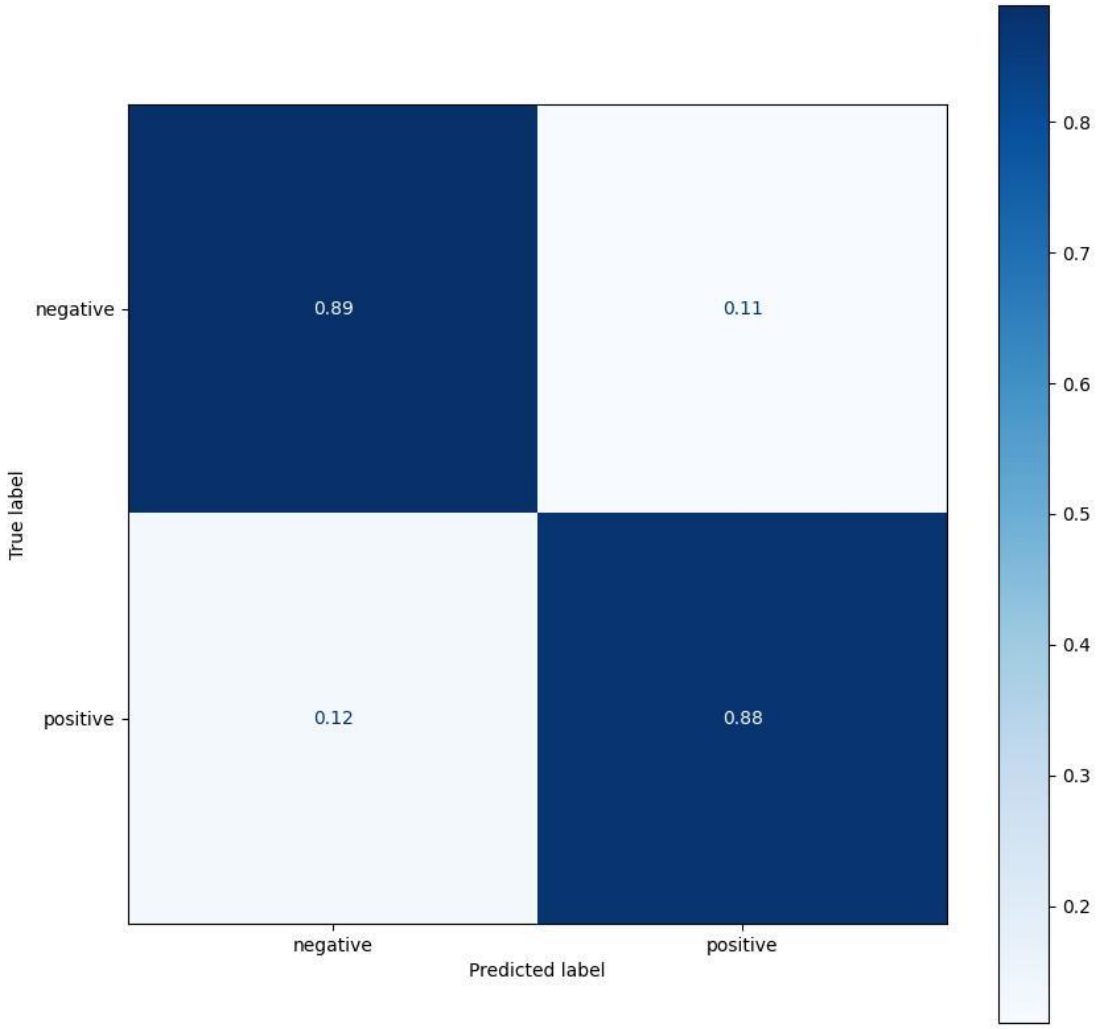
So, we retrain the whole train data and use GRU-M11 hyperparameters to get the final model. Then, let's see what the final performance will be.

Because have no validation, so we choose the loss which is the same as the GRU-M11 model to avoid overfitting.

So we choose the weight at epoch 14.



Submit test score:0.8825



On test set confusion matrix

The confusion matrix is very balance!! So, the preprocess is meaning!!

- **Conclusion**

We process data from the origin csv, rather than call IMDB api data. Because we want to know more information about it. Although preprocessing by self is very tiring, it let us more impressed with the NLP processing technique. Thus, when we do the experiments the preprocess information make us some good ideas to try.

In the above experiment, we tried tuning hyperparameters, including the sequence length, number of units, hidden size, and optimizer in RNN.

The sequence length is 126 because the train data average words are 126, so we test this parameter first. Then choose more sequence lengths to try to improve model performance at the validation set. But after trying, the sequence length 126 appears as the better number.

Then, try the hidden size layer from 64 to 256. The more hidden size of the model will have more parameters needed to train. Finally, the hidden size of 128 is better than 64 and 256 a lot. Thus we still test the optimizer, but Adam is always great than others.

In RNN, our best performance in validation is 81.29% and costs 62k parameters.

Next, we experiment with the stronger model: LSTM, and try the dropout about to next time steps.

Same as the RNN test strategy, try the number of layers and the result is 3 better. Then, turning the sequence lengths 150 and 180, but not make it better too. The hidden size rather than RNN's situation may be the parameter 128 in LSTM at this preprocessing method is inappropriate.

Next, we adjust the dropout ratio about the time step input. The best performance is at 0.3. Because we use dropout, so we return to adjust the sequence length hope may get a different result. But as the table show, 126 still get better performance although their gap was very little. Experiments showed that the optimizer Adam is still better than SGD and RMSprop.

In LSTM, our best performance in validation is 87.25% and costs 382k parameters.

The final model is GRU, which is newer architecture than LSTM.

As in the test before, we do the same initial turning method and get a good performance than RNN and LSTM. Worth mentioning is the GRU feed 180 sequence lengths are better which was different with LSTM and RNN.

The better hidden size the same as LSTM is 128. But when adding the dropout, the performance gets better. So, after adding dropout, we retry the sequence lengths and hidden size. Maybe the performance for our model cannot be through 90%, the performance does not improve, still around 88%.

In GRU, our best performance in validation is 88.96% and costs 187k parameters.

Finally, we choose the GRU model as our best model and the test score is 88.25% which is close to the validation. And we plot the confusion matrix to know if our model predicts the situation.

We can see the confusion matrix is perfect because the positive and negative mistake ratios are very close. That means our model is reliable! It can be used in the real world!

Experiments equipment:

GPU: RTX3060 12G VRAM