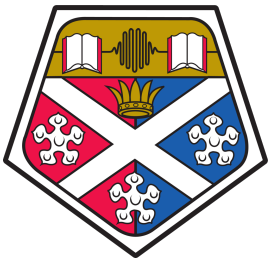

CS985/CS987 Advanced Machine Learning for Data Analytics Assignment

Team: Kappa

Name	ID	E-mail
Ruixian Zhao	201867867	ruixian.zhao.2018@uni.strath.ac.uk
Jie Cao	201861356	jie.cao.2018@uni.strath.ac.uk
Szilvia Kondorai	201884363	szilvia.kondorai.2018@uni.strath.ac.uk

Dataset: IMDB Reviews, Fashion MNIST

Date: 24th March 2019

Lecturer: Dr. Leif Azzopardi

Pages Count: 4 Pages

(1 page of cover and 2 pages of appendix)

Dataset One: IMDB Reviews

Best Architecture

The architecture of the best model for classifying the IMDB reviews shows in Figure 1.

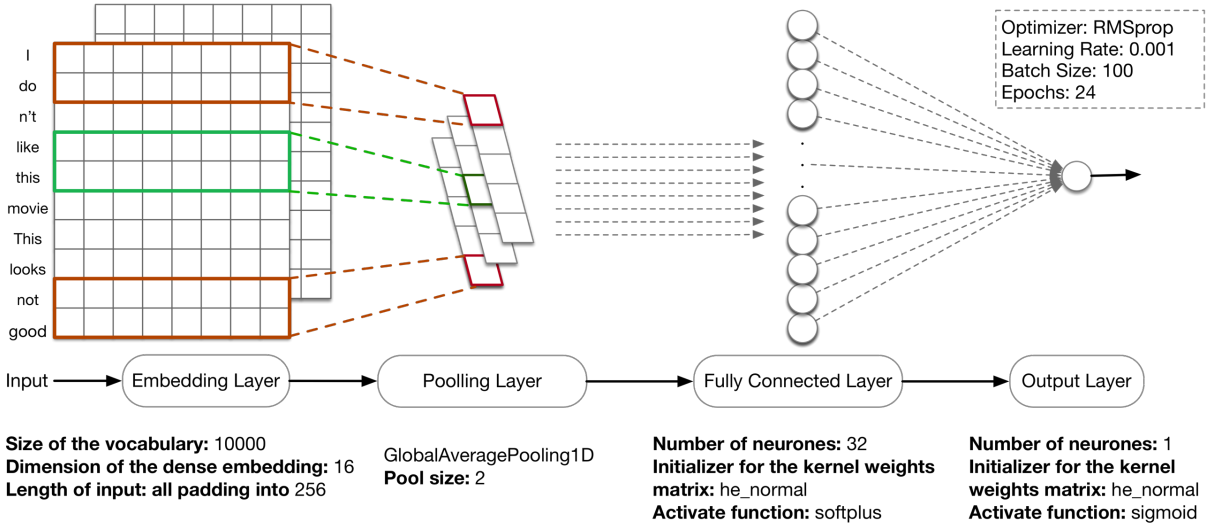

Figure 1 Architecture of Best Classify Model for Dataset IMDB

The reviews in the dataset is array of integers, they have different length. We pad them to get an integer tensor of shape `max_length * num_reviews`, and here the `max_length` is 256. The first `Embedding` layer get the embedding vector for each word-index, the vectors add a dimension to the output array. Since in the downloading part only the top 10000 most frequently occurring words have been downloaded, the size of the vocabulary is 10000. The second layer is a `Global average pooling layer and` connected with a fully connected layer with 32 neurons, with `he_normal` as the weights initializer and `softplus` as the active function. Finally, the last layer is the `output layer` with a single node, the activation function is `sigmoid`, it will output a floating value between 0 and 1 to represent the probability of the results. To train this model, we choose optimizer `RMSprop` with learning rate `0.001` and `batch size` of 100, takes `24 epochs`. All the hyper-parameters in this model is decided by grid search, all the attempts have been recorded in the appendix Table 3 and raw code named `gridsearch1.py` is provided. It only takes **19 seconds**[1] to train this model and achieved **96%** in training accuracy and **88%** in testing accuracy.

Second Good Architecture

Different from the network above, combination 2 has a Convolutional layer after the embedding layer, and used max pooling in the pooling layer. The Convolutinal layer can produce a feature map based on the previous window of words, `relu` activate function used here. Drop out method has been applied to the fully connected layer. Also there have many turning among the hyper parameters, the details and architecture are shown in Figure 2. Even this model seems to be more complicated, it only needs 5 epochs to achieve a good accuracy. To train this model, it needs **25 seconds** and can achieved **97%** in

[1] My computer is MacBook Pro (Retina, 13-inch, Early 2015), macOS Majove 10.14.4, 2.7 GHz Intel Core i5, 8 GB 1867 MHz DDR3, and Intel Iris Graphics 6100 1536 MB (which is not used). Different computers needs different time.

training accuracy and **87%** in testing accuracy. Again, the different settings for the hyper parameters of Combination 1 and 2 are in Appendix Table 3 and 4.

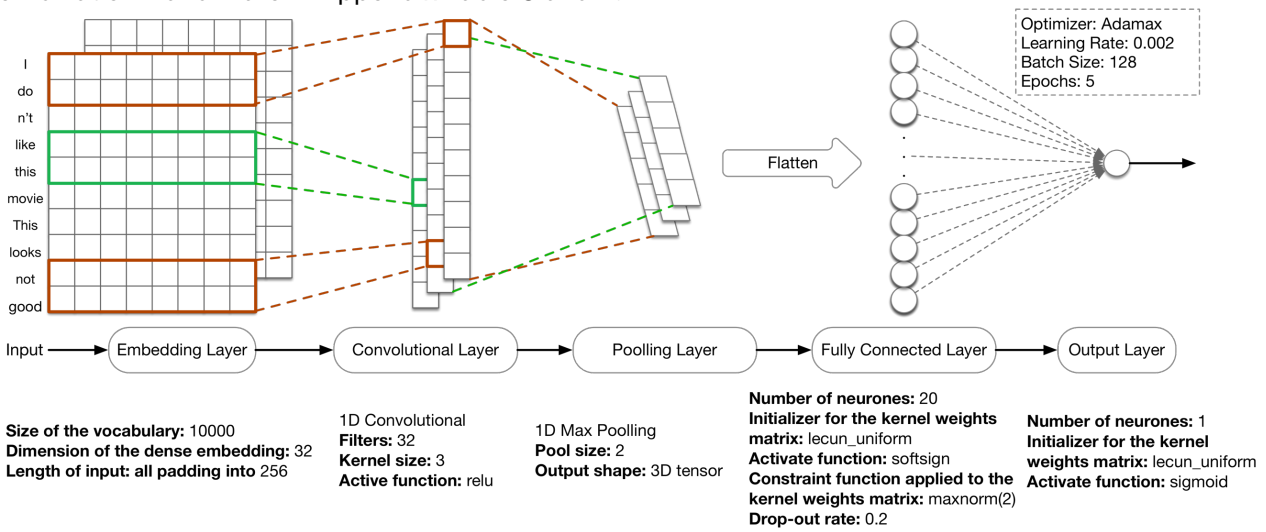

Figure 2 Architecture of the Second Best Classify Model

Other Architectures

We also tried some other architectures. The Combination 3 used recurrent layers instead of convolutional and pooling layers. We tried different one, LSTM with 100 output space and GRU(Gated Recurrent Unit) with 32 output space. Simple fully connected network (combination 4) and combine the convolutional layer with recurrent layer (combination 5) also has been tested. The detailed of the test is in Table 1.

Table 1 Different Architectures and Different Hyper-Parameters for IMDB Reviews Model

ID	Comb-ination	Configuration	Parameters	Training Time (seconds)	Training Accuracy (%)	Testing Accuracy (%)
1	**1**	**GlobalAveragePooling**	**n=32, e=24, bs=100, lr=0.001, details in Figure 1**	**19**	**95.8**	**87.8**
2	2	Conv + MaxPooling	n=20, e=40, bs=128, lr=0.002, details in Figure 2	24	96.7	87.0
3	3	LSTM	e=20, bs=64, lr=0.002, dr=0.2	360	83.4	78.6
4	3	GRU	e=19, bs=500, lr=0.002,	358	84.5	75.6
5	3	FCN	n=250, e=14, bs=512, lr=0.002 af=relu, op=Adam	66	100	85.4
6	5	Conv+Pooling+LSTM	e=10, bs=500, lr=0.002, af=relu,op=Adam	339	98.2	86.0
7	4	FCN + drop rate	model 5, e=16, dr=0.5	81	100	85.6
8	5	CNN + LSTM + FCN	n=16, e=10, bs=500,lr=0.005, af=relur, op=Adam	325	99	85.0

Seed=7; n = number of neurons in Dense layer; e = number of epochs; bs = the batch size; lr = learning rate, af = active function; op = optimizer; dr = drop-out rate.

By compare with those different models, and after grid search to find the best combination of the hyper-parameters, **model 1** is the best one: the simple fully connected layers reached limited accuracy and easy to overfit. The recurrent layers are time consuming in training, and the combination of CNN with RNN is also not good. As for model 1, its architecture is simple and easy to understand, it has less parameters and fast only training, also it achieves greatest testing accuracy among all the models.

Dataset 2: Fashion MNIST

Best Architecture

The architecture of the best model to identify the clothes dataset shows in Figure 3.

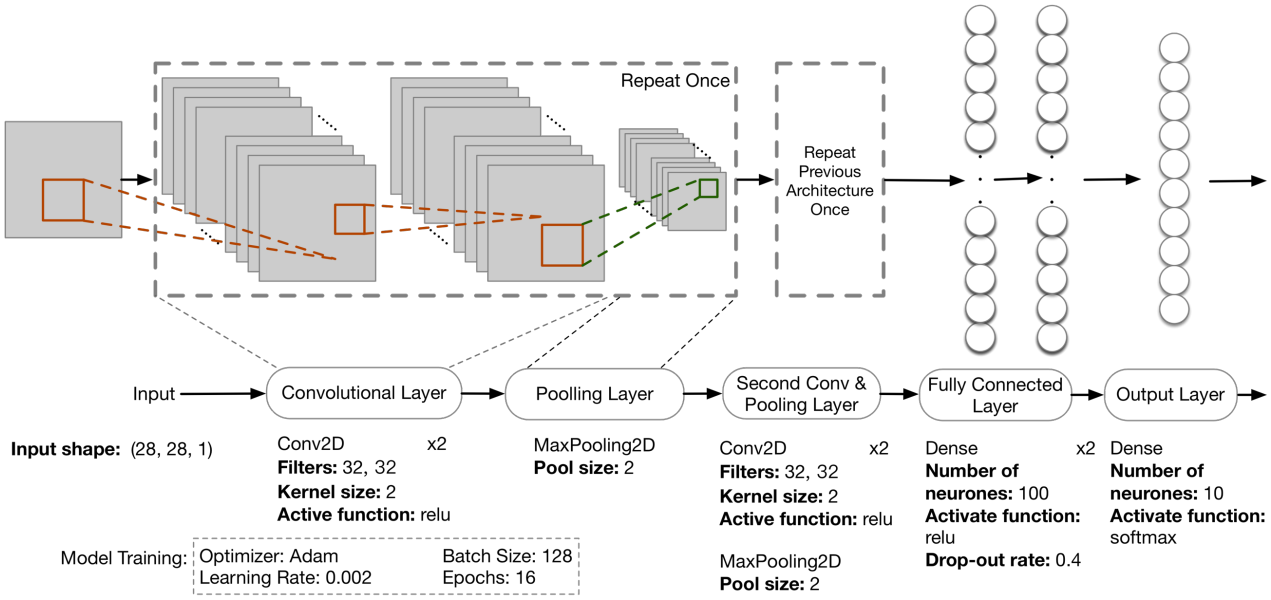

Figure 3 Architecture of Best Classify Model for Dataset Fashion MNIST

First the input layer received the 28×28 image (the input has been reshaped to `(28, 28, 1)`) and passes it to the `Convolutional layer`, which is the second layer of the architecture. It can filter the image to find some patterns. There has a double `Conv layer` where each of them have 32 filters, with both kernel size is 2×2. After the Double convolutional layer, it is the `Max Pooling layer`. This can filter out the patterns in the image and reduce the parameters in the network by a 2×2 size of pooling. To have a better performance we repeat this CNN once: both 32 filters. Then connected them with a double hidden layer of `FCN`. There have 100 neurons in each layer and with `drop-out rate of 0.4`. This can avoid overfit in the training test but can increase the training time. Finally, the `output layer` is a combination of 10 neurons with `softmax` active function, to represent the 10 classes in the dataset need to be classify. The more details of the architecture are in the Figure 3. To train this model, we choose `batch size` of `128` and `16 epochs` with `Adam` optimizer. This model can achieve **93%** accuracy on testing dataset and **94%** on training dataset.

Other Architectures

The architecture mentioned above with double CNN is combination 1, single CNN is combination 2, simple whole FCN is combination 3. We have tried different combinations of them and adjust the different hyper-parameters in it, some of the model can achieve good results but too time consuming. The details of it are shown in Table 2. The learning rate has been proofed that the 0.002 is best for Adam optimizer, others will not change the results so much.

Implement double CNN can let the network more stable, but also increase the training time. With different setting parameters and slightly changes on the architecture, can influence the results much. For example the model 4 and 3 achieved much similar results on the testing accuracy but model 4 is much more faster than 3. With single layer of CNN the network is easier to get overfit while they mostly

achieved good among training dataset but not good on testing accuracy. The advantage of that is most of them is much faster than combination 1. For the simple FCN, it is easy to train but they will reach a ceiling of accuracy and the performance are not good enough. There is still much space to turning the parameters in the network.

Table 2 Different Architectures and Different Hyper-Parameters for Fashion MNIST Model

ID	Comb-ination	CNNs Configuration		Parameters	Training Time (min)	Training Accuracy (%)	Testing Accuracy (%)
1	1	32, 64	16, 32	n^2=64, dr=0.5, e=18, bs=256	21	88.9	91.5
2	1	64	64, 64	n^2=256, dr=0.4, e=24, bs=128	32	97.7	92.3
3	1	64, 64	32, 32	n^2=128, dr=0.4, e=24, bs=128	55	95.8	92.6
4	**1**	**32, 32**	**32, 32**	**n^2=100, dr=0.4, e=16, bs=128**	**17**	**93.8**	**92.6**
5	1	32, 64	32	n^2=100, dr=0.4, e=15, bs=256	21	93.1	92.1
6	1	32	32	n^2=100, e=18, bs=256	7.9	96.3	91.7
7	2	32, 32, bias constant		n^2=128, dr=0.4, e=15, bs=64	14	98.8	91.0
8	2	64 + AveragePooling		n^2=16, dr=0.4, e=10, bs=256	5.6	92.2	90.0
9	3	double hidden layer		n^2=256, dr=0.5, e=16, bs=64	1.9	87.4	87.5
10	3	double hidden layer		n^2=256, dr=0.5, e=30, bs=32	4.9	87.7	88.3

Learning Rate: 0.002; Seed = 7;

The configuration shows the details of number of filters in convolutional layers, two column refers to the double CNN combination, where the first column represents the one or two convolutional layers in part 1 CNN. One parameter means there only have single convolutional layer. Other changes will be described in column "Parameters", if not they are the same with Model 1.

n = number of neurons in Dense layer; n^2 means double hidden layer in FCN; e = number of epochs; bs = the batch size; dr = drop-out rate.

Conclusion

By comparing all the models, we choose the **model 4** as the best model. As mentioned before the combination 1 is better than combination 2, and among all the models in combination 1, model 4 achieved the best testing accuracy while cost the least of time. Moreover, its architecture is not that complicated only with a repeated CNN. The combination 3 with the FCN performance not that good so it will not be into the consideration.

Appendix

Files Introduction

The environment used in this project is built on Python 3.7 and required packages is in "`requirements.txt`". Here are the list of files in the folder and briefly instructions about them.

File name	Introductions	Notes
README.MD	Introduction and Instructions	Recommended:
requirements.txt	Required packages	Details are
fashion.py	Run varies of models for fashion MNIST dataset	provided in
fashion-final.ckpt; fashion_launcher.py	Launch the final best model of fashion MNIST dataset	README.MD
imdb.py	Run varies of models for IMDB dataset	
imdb-final.ckpt; imdb_launcher.py	Launch the final best model of IMDB dataset	
logs_fashion_1; logs_imdb_1 (folder)	Logs to tensorboard for ther two best models	
logs (folder)	Logs to tensorboard for all other models in this report	
helper1, 2, 3, 4, 5.py	Helper script to build model and analyses	
results_fashion.txt; results_imdb.txt	The running log and model summary for all models	
gridsearch1, 2.py	The raw code of gridsearch for IMDB mdels	

Grid Search for Combination 1 & 2 of IMDB Reviews Model

Table 3 Grid Search for Combination 1 of IMDB Reviews Model

No.	Parameters	Range of grid parameters	Best combinations	Testing Acc
1	batch size, epochs	batch_size = [128, 256, 512] epochs = [5, 10, 15, 20]	batch size = 128 epochs = 20	87.48
2	batch size, epochs	batch_size = [32, 64, 100, 128] epochs = [18, 22, 24]	**batch size = 100** **epochs = 24**	87.78
3	batch size, epochs	batch_size = [90, 100, 110] epochs = [24, 25, 26, 27]	batch size = 110 epochs = 24	87.75
4	optimizer	optimizer = ['SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Adamax', 'Nadam']	**optimizer = 'RMSprop'**	87.80
5	learning rate	learning_rate = [0.001, 0.002, 0.005, 0.01, 0.02]	**learning_rate = 0.001**	87.81
6	learning rate	learning_rate = [0.0015, 0.001, 0.0005]	learning_rate = 0.0005	87.67
7	weight initial model	init_mode = ['uniform', 'lecun_uniform', 'normal', 'zero', 'glorot_normal', 'glorot_uniform', 'he_normal', 'he_uniform'	**init_mode = 'he_normal'**	87.88
8	active function	activation = ['softmax', 'softplus', 'softsign', 'relu', 'tanh', 'sigmoid', 'hard_sigmoid', 'linear']	**activation = softplus**	88.05
9	number of neurons	neurons = [8, 10, 16, 32, 64, 128, 256, 512]	**neurons = 32**	**88.17**
10	number of neurons	neurons = [28, 30, 32, 34, 36, 38]	neurons = 32	88.16
11	drop-out rate, weight constraint	dropout_rate = [0.0, 0.2, 0.4, 0.6, 0.8] weight_constraint = maxnorm [1, 2, 3, 4, 5]	**dropout_rate = 0.0** weight_constraint = 3	73.17
12	weight constraint	weight_constraint = maxnorm [1, 2, 3, 4, 5]	weight_constraint = 4	78.17

The parameters been bolder is the combinations has been chosen:
Batch size: 100, epoch: 24, optimizer: RMSprop, learning rate: 0.001, init_mode: he_nromal, activation: softplus, number of neurons: 32, drop-out rate: 0, no weight constraint function.

Table 4 Grid Search for Combination 2 of IMDB Reviews Model

No.	Parameters	Range of grid parameters	Best combinations	Testing Acc
1	batch size, epochs	batch_size = [256, 512] epochs = [5, 10, 15]	batch size = 256 epochs = 5	86.56
2	batch size, epochs	batch_size = [64, 128, 256, 512] epochs = [4, 5, 8]	**batch size = 128** **epochs = 5**	86.60
3	optimizer	optimizer = ['SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Adamax', 'Nadam']	**optimizer = 'Adamax**	86.62
4	learning rate	learning_rate = [0.001, 0.002, 0.005, 0.01, 0.02]	learning_rate = 0.01	86.37
5	learning rate	learning_rate = [0.001, 0.002, 0.0025, 0.003]	**learning_rate = 0.002**	87.18
6	learning rate	learning_rate = [0.001, 0.002, 0.0025, 0.003]	learning_rate = 0.0015	87.08
7	weight initial model	init_mode = ['uniform', 'lecun_uniform', 'normal', 'zero', 'glorot_normal', 'glorot_uniform', 'he_normal', 'he_uniform'	init_mode = 'glorot_normal'	87.03
8	weight initial model	init_mode = ['uniform', 'lecun_uniform', 'normal', 'zero', 'glorot_normal', 'glorot_uniform', 'he_normal', 'he_uniform'	**init_mode = 'uniform'**	87.30
9	active function	activation = ['softmax', 'softplus', 'softsign', 'relu', 'tanh', 'sigmoid', 'hard_sigmoid', 'linear']	**activation = tanh**	87.18
10	active function	activation = ['softmax', 'softplus', 'softsign', 'relu', 'tanh', 'sigmoid', 'hard_sigmoid', 'linear']	activation = sigmoid	87.08
11	number of neurons	neurons = [1, 5, 10, 16, 32, 64, 128, 256, 512]	**neurons = 16**	87.18
12	number of neurons	double neurons = [1, 2, 3, 4, 5, 6, 7, 8, 16]	neurons = 4, 16	87.07
13	drop-out rate, weight constraint	dropout_rate = [0.0, 0.2, 0.4, 0.6, 0.8] weight_constraint = maxnorm [1, 2, 3, 4, 5]	dropout_rate = 0.8 weight_constraint = 1	87.17
14	drop-out rate, weight constraint	dropout_rate = [0.0, 0.2, 0.4, 0.6, 0.8] weight_constraint = maxnorm [1, 2, 3, 4, 5] dropout_rate = [0.0, 0.2, 0.4, 0.6, 0.8]	**dropout_rate = 0.2** **weight_constraint = 2**	**87.62**

The parameters been bolder is the combinations has been chosen.

Batch size: 128, epoch: 5, optimizer: Adamax, learning rate: 0.002, init_mode: uniform, activation: tanh, number of neurons: 16, drop-out rate: 0.2, weight constraint function: maxnorm(2).