Programming Assignment-2 Report

Team Members:

1. Sandesh Kumar Srivastava
2. Charan Reddy Bodennagari
3. Venkata Narayana Rohit Kintali

Team Number: 35

Course Number: CSE 574

Report 1:

Each word that is present in the review will have three features, positive count (number of times the word appeared in positive reviews) Negative_counter (Number of times the word appeared in negative reviews) and total count (no if times the word appeared in both positive and negative reviews).

Our decision is based on the pos_negative_ratio () of the words. If the positive word count is more than the negative count the review is considered positive if it is less then it is considered negative.

As we are using the log of counter ratio to assign a sentiment to each word, we are taking the positive and negative counter ratios, and we choose to ignore words appearing approximately equally in both positive and negative reviews.

In order to classify words as neutral, we use a range and calculate corresponding test accuracy.

Neutral words ratio range	Test Accuracy
(0,0)	76.3
(-0.1,0.1)	77.1%
(-0.5,0.5)	80.9%
(-0.6,0.6)	81.7%
(-0.7,0.7)	80.6%
(-1.0,1.0)	74.9%

We observe that test accuracy increases and gets maximum at around (-0.6,0.6) range.

Report 2:

For last few iterations, we observe following accuracies:

Train acc: 0.949583, Test_acc: 0.852500

Train acc: 0.948792, Test_acc: 0.847500

Train acc: 0.946667, Test_acc: 0.853750

Train acc: 0.947750, Test_acc: 0.856250

Train acc: 0.946667, Test_acc: 0.853750

Time elapsed - 34.05550026893616 seconds.

The train accuracy of the vanilla neural network is approximately 94.7 while the test accuracy is approximately 85.3 which is significantly higher that the non-ML model.

The time took for the model to run is 34 seconds.

The vanilla neural network is better than the rule-based algorithm used because the rule-based algorithm doesn't do any learning as the model stays constant always where as the neural network is better as it gets better with time and data.

Report 3:

Effect of different parameters on performance of the model:

In order to study the effect of individual parameters on the model, we have kept the default values of parameters as what was provided in the notebook initially.

```
# parameters of the network
num_epochs = 50
n_hidden_1 = 10  # 1st layer number of neurons
and so on...
```

Task 1: Effect of hidden layer width:

We study the effect of hidden layer width by changing n_hidden_1 value and present it below:

Number of neurons	Training accuracy	Test accuracy	Time taken(seconds)
1	0.926	0.841	23.766
10	0.948	0.860	31.502
100	0.903	0.853	37.984
250	0.873	0.818	61.086
500	0.500	0.500	94.874

As we increase the number of neurons, both training and test accuracy increase initially. But if we increase the number of neurons to a large value, accuracy begins to fall and for very large value of 500, it becomes roughly 50% depicting a random guess.

Time taken to run the model increases with increase in number of neurons.

Task 2: Effect of number of hidden layers:

We study the effect of number of hidden layers by adding layers and present it below:

Number of hidden layers	Training accuracy	Test accuracy	Time taken(seconds)
1	0.948	0.860	31.502
2	0.902	0.846	32.578
3	0.852	0.816	33.504
5	0.500	0.501	35.768

As we increase the number of hidden layers, both training and test accuracy decrease with each layer addition. Once we reach 5 layers, the accuracy gets substantially low.

Time taken to run the model increases with increase in number of hidden layers.

Task 3: Effect of number of epochs

We study the effect of number of hidden layers by modifying "num_epochs" and present it below:

Number of epochs	Training accuracy	Test accuracy	Time taken(seconds)
5	0.828	0.782	3.286
25	0.925	0.857	15.833
50	0.943	0.845	32.045
100	0.954	0.857	66.500

As we increase the number of epochs, there is a slight increase in both training and test accuracy initially but as number of epochs increase.

Time taken to run the model increases with increase in number of epochs.

Task 4: Effect of ignoring set of words:

We study the effect of ignoring set of words by modifying "find_ignore_words()" and present it below:

Range of pos_neg_ratios	Vocab size after ignoring words	Training accuracy	Test accuracy	Time taken(seconds)
Null (Not ignoring any words)	4116	0.943	0.856	32.149
(0,0)	4084	0.948	0.860	31.502
(-0.3,0.3)	2249	0.941	0.841	19.984
(-0.5,0.5)	1379	0.914	0.843	14.028
(-1.0,1.0)	402	0.857	0.816	8.387

We mention the vocab size obtained by removing the ignored words in second column above.

As we increase the range of pos_neg_ratios to consider for ignoring words, we observe that the vocab size is reduced as more words are ignored. Also, training and testing accuracies are obtained maximum when only words with pos_neg_ratio = 0 are ignored.

Time taken to run the model decreased with increase in range of pos_neg_ratio as number of words are reduced.

Report 4:

Task 1: Evaluation of 1 hidden layer neural network

In this scenario we are running the neural network for 1 hidden layer with 'RELU' activation function.

The model was fit with 500 epochs and then evaluated. The accuracies and running time for few epochs are reported as follows:

```
Epoch 498/500
100000/100000 [==============================] - 6s 64us/step - loss: 0.7371 - accuracy: 0.8024
Epoch 499/500
100000/100000 [==============================] - 6s 61us/step - loss: 0.7325 - accuracy: 0.8018
Epoch 500/500
100000/100000 [==============================] - 6s 63us/step - loss: 0.7286 - accuracy: 0.8024
```

Training accuracy: 80.24
Test accuracy : 78.2
Time elapsed - 4230.87 seconds.

After running the model twice with same number of epochs we have observed that there is not much change in the accuracies.

Task 2: Evaluation of 3 and 5 hidden layer neural network

For comparing the performance of the model, we are increasing the number of hidden layers to three and five. These both combinations experimented will result in different values of training accuracies and test accuracies. The instances used for obtaining the results are as follows:

THREE HIDDEN LAYERS:

Code:

```
model = Sequential()
model.add(Dense(256, activation='relu',input_dim=data_train.shape[1]))
model.add(Dense(256, activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

```
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
model.fit(data_train, label_train, epochs=500, batch_size=32)
```

Results:

```
Epoch 495/500
100000/100000 [==============================] - 9s 93us/step - loss: 2.5525 - accuracy: 0.2045
Epoch 496/500
100000/100000 [==============================] - 9s 94us/step - loss: 2.1681 - accuracy: 0.2067
Epoch 497/500
100000/100000 [==============================] - 9s 94us/step - loss: 2.1250 - accuracy: 0.2022
Epoch 498/500
100000/100000 [==============================] - 9s 93us/step - loss: 2.1156 - accuracy: 0.2156
Epoch 499/500
100000/100000 [==============================] - 9s 94us/step - loss: 2.1779 - accuracy: 0.2089
Epoch 500/500
100000/100000 [==============================] - 9s 93us/step - loss: 2.1883 - accuracy: 0.2083
```

Training accuracy: 21
Test accuracy: 20.52
Time elapsed - 5296.77 seconds.

FIVE HIDDEN LAYERS:

Code:

```
model = Sequential()
model.add(Dense(256, activation='relu',input_dim=data_train.shape[1]))
# you can add more Dense layers here
model.add(Dense(256, activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
model.fit(data_train, label_train, epochs=500, batch_size=32)
```

Results:

```
Epoch 496/500
100000/100000 [==============================] - 13s 126us/step - loss: 2.3027 - accuracy:
0.0993
Epoch 497/500
100000/100000 [==============================] - 13s 130us/step - loss: 2.3028 - accuracy:
0.0980
Epoch 498/500
100000/100000 [==============================] - 13s 127us/step - loss: 2.3028 - accuracy:
0.0971
Epoch 499/500
100000/100000 [==============================] - 15s 154us/step - loss: 2.3027 - accuracy:
0.1007
Epoch 500/500
100000/100000 [==============================] - 14s 142us/step - loss: 2.3027 - accuracy:
0.1004
```

Training accuracy: 10.04
Test accuracy : 10
Time elapsed – 6125.74 seconds.

We can observe from the above results that as the number of hidden layers increases, the model accuracy decreases due to overfitting and we can see that from hidden layer 1 to hidden layer 3 the training as well as testing accuracies come down drastically. Also, as observed for hidden layer 3 to 5 there is a further decrease in the respective accuracies from around 21 to 10 percent.

Task 3: Evaluation of 1 hidden layer neural network with lower resolution images.

We will compare the results from task1,with original resolution (28 x 28) with results obtained using lower resolution images.

Resolution (20 x 20)

By using the resize_images() function, we decrease the resolution of the image to (20 x 20).We will fit the model using the combination from the task1.

```
data_train_20=resize_images(data_train,[20,20])
model = Sequential()
model.add(Dense(256, activation='relu', input_dim=data_train_20.shape[1]))
# you can add more Dense layers here
model.add(Dense(10, activation='softmax'))
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
model.fit(data_train_20, label_train, epochs=500, batch_size=32)
```

The above-mentioned model is fit with 500 epochs and then evaluated. The accuracies and running time are reported as follows:

```
Epoch 495/500
100000/100000 [==============================] - 8s 78us/step - loss: 1.0259 - accuracy: 0.7373
Epoch 496/500
100000/100000 [==============================] - 8s 79us/step - loss: 1.0201 - accuracy: 0.7367
Epoch 497/500
100000/100000 [==============================] - 8s 82us/step - loss: 1.0358 - accuracy: 0.7363
Epoch 498/500
100000/100000 [==============================] - 8s 83us/step - loss: 1.0151 - accuracy: 0.7364
Epoch 499/500
100000/100000 [==============================] - 8s 81us/step - loss: 1.0122 - accuracy: 0.7371
Epoch 500/500
100000/100000 [==============================] - 9s 89us/step - loss: 1.0164 - accuracy: 0.7376
```

Training accuracy: 73.76
Test accuracy: 72.57
Time elapsed - 2932.86 seconds.

The accuracies obtained have decreased as compared to task1 and the running time reduced considerably. Hence, lowering the resolution to (20x20) hinders the accuracy slightly and drastically the running time.

Resolution (15 x 15)

Reducing the resolution to (15x15) and evaluating the model using 500 epochs would yield the following results.

Code is similar to the 20x20 code as shown above.

Results:

```
Epoch 495/500
100000/100000 [==============================] - 4s 42us/step - loss: 0.7313 - accuracy: 0.8155
Epoch 496/500
100000/100000 [==============================] - 4s 43us/step - loss: 0.7390 - accuracy: 0.8158
Epoch 497/500
100000/100000 [==============================] - 4s 42us/step - loss: 0.7316 - accuracy: 0.8153
Epoch 498/500
100000/100000 [==============================] - 4s 43us/step - loss: 0.7219 - accuracy: 0.8156
Epoch 499/500
100000/100000 [==============================] - 4s 42us/step - loss: 0.7285 - accuracy: 0.8158
Epoch 500/500
100000/100000 [==============================] - 4s 42us/step - loss: 0.7252 - accuracy: 0.8163
```

Training accuracy: 81.63
Test accuracy : 69.58
Time elapsed - 2244.63 seconds.

The accuracy has slightly increased as compared to the base case of 28x28(task1). On the other side, the running time decreases by 50% of the time taken to complete task1.Hence, lowering the resolution by (15x15) doesn't affect the accuracy much but definitely reduces the running time by a large number.

Resolution (10 x 10)

Reducing the resolution to (10x10) and evaluating the model using 500 epochs would yield the following results.

Code is similar to the 20x20 code as shown above.

Results:

```
Epoch 496/500
100000/100000 [==============================] - 3s 33us/step - loss: 1.2034 - accuracy: 0.7115
Epoch 497/500
100000/100000 [==============================] - 3s 32us/step - loss: 1.1998 - accuracy: 0.7108
Epoch 498/500
100000/100000 [==============================] - 3s 32us/step - loss: 1.2027 - accuracy: 0.7099
Epoch 499/500
100000/100000 [==============================] - 3s 32us/step - loss: 1.1973 - accuracy: 0.7092
Epoch 500/500
100000/100000 [==============================] - 3s 33us/step - loss: 1.2055 - accuracy: 0.7097
```

Training accuracy: 70.97
Test accuracy: 68.06
Time elapsed - 1645.25 seconds.

As observed the accuracy reduces by around 10 percent compared to task1 and the running time decreases by 60 percent of the task1 finishing time. Therefore, lowering the resolution to (10x10) hinders the accuracy slightly but lowers the running time considerably.

Resolution (5 x 5)

Reducing the resolution to (5x5) and evaluating the model using 500 epochs would yield the following results:

Code is similar to the 20x20 code as shown above.

Results:

```
Epoch 496/500
100000/100000 [==============================] - 3s 26us/step - loss: 1.7366 - accuracy: 0.4673
Epoch 497/500
100000/100000 [==============================] - 3s 26us/step - loss: 1.7422 - accuracy: 0.4678
Epoch 498/500
100000/100000 [==============================] - 3s 26us/step - loss: 1.7269 - accuracy: 0.4680
Epoch 499/500
100000/100000 [==============================] - 3s 28us/step - loss: 1.7346 - accuracy: 0.4658
Epoch 500/500
100000/100000 [==============================] - 3s 26us/step - loss: 1.7405 - accuracy: 0.4687
```

Training accuracy: 46.87
Test accuracy: 46.42
Time elapsed - 1314.94 seconds.

As expected the accuracies and the running time decreases compared to task1.The accuracy reduces by 30 percent of the task1 accuracy and time taken to finish is around 33 percent of the time taken for task1.Hence, lowering the resolution to (5x5) both the accuracy and running time decreases phenomenally.

Conclusion:

1. Accuracy decreases as we increase the number of hidden layers in the model. It will cause the network to overfit to the training set, that is, it will learn the data, but it won't be able to generalize to new unseen data.
2. We can definitely lower the resolution of the image in order to reduce the running time in a good amount without affecting the accuracies significantly except for the (5x5) resolution.

Code for all relevant changes is added in commented state in notebook files.