Lab Assignment 1 Report

Zhou Gaoqiang – G2304064A

# Introduction

This report is divided into two sections. The first section examines the performance changes of the model by modifying vectorization, adding hidden layers, and adjusting the frequency cutoff value. The second section focuses on modifying the output nodes from 1 to 2, ensuing the program runs correctly and explaining the reasons for the modifications.

# Section 1

## Task 1-1 Changing One-Hot Encoding to Term-Frequency Vectorization **ONLY**

### Result Overview:

|  |  |  |  |
| --- | --- | --- | --- |
| Loss | Accuracy | Test Loss | 0.482 |
| Test Accuracy | 86.07% |

### Observation and analysis: Simply from the results, compared to the original model, the accuracy slightly increased from 85.03% to 86.07%. However, from the loss and accuracy graphs of both the original and current models, it can be observed that overfitting may have occurred after 20 epochs, which resulted in no significant improvement in validation and test accuracy.

## Task 1-2 Add Additional Hidden Layer **ONLY**

### Result Overview:

|  |  |  |  |
| --- | --- | --- | --- |
| Loss | Accuracy | Test Loss | 0.887 |
| Test Accuracy | 84.64% |

Dim=20

|  |  |  |  |
| --- | --- | --- | --- |
| A graph with blue dots  AI-generated content may be incorrect.Loss | A graph of a training and validation accuracy  AI-generated content may be incorrect.Accuracy | Test Loss | 0.727 |
| Test Accuracy | 86.07% |

Dim=10

|  |  |  |  |
| --- | --- | --- | --- |
| A graph with blue dots  AI-generated content may be incorrect.Loss | A graph of a training and validation accuracy  AI-generated content may be incorrect.Accuracy | Test Loss | 1.199 |
| Test Accuracy | 83.59% |

Dim=40

### Observation and analysis: Adding a new hidden layer requires specifying the number of dimensions (Dim). Since the optimal value was unclear, I experimented with three different settings: 20 (same as the previous layer), 10 (half of the previous layer), and 40 (double the previous layer). However, the results showed that except for Dim = 10, performance actually declined. This could be due to excessive layers and dimensions leading to overfitting or training instability. Even for Dim = 10, a similar issue to Task 1-1 was observed, where validation loss increased significantly, indicating potential generalization problems.

## Task 1-3 Changing the Frequency Cutoff to 0 **ONLY**

### Result Overview:

|  |  |  |  |
| --- | --- | --- | --- |
| Loss | Accuracy | Test Loss | 0.314 |
| Test Accuracy | 87.50% |

### Observation and analysis: Both the test loss and accuracy show significant improvements compared to the original model, with no signs of overfitting. This make it the best modification approach so far.

## Task 1-4 Find the Best Combination

To find the optimal approach, I tested all possible combinations. However, for adding a hidden layer, I only selected the Dim=10 option.

### Result Overview:

|  |  |  |  |
| --- | --- | --- | --- |
| A graph with blue dots  AI-generated content may be incorrect.Loss | A graph of a training and validation accuracy  AI-generated content may be incorrect.Accuracy | Test Loss | 0.754 |
| Test Accuracy | 84.24% |

Term-Frequency + Hidden Layer (Dim=10)

|  |  |  |  |
| --- | --- | --- | --- |
| A graph with blue dots  AI-generated content may be incorrect.Loss | A graph of a training and validation accuracy  AI-generated content may be incorrect.Accuracy | Test Loss | 0.338 |
| Test Accuracy | 87.89% |

Term-Frequency + Frequency Cutoff = 0

|  |  |  |  |
| --- | --- | --- | --- |
| A graph with blue dots  AI-generated content may be incorrect.Loss | A graph of a training and validation accuracy  AI-generated content may be incorrect.Accuracy | Test Loss | 0.417 |
| Test Accuracy | 88.54% |

Hidden Layer (Dim=10) + Frequency Cutoff = 0

|  |  |  |  |
| --- | --- | --- | --- |
| A graph with blue dots  AI-generated content may be incorrect.Loss | A graph of a training and validation accuracy  AI-generated content may be incorrect.Accuracy | Test Loss | 0.462 |
| Test Accuracy | 87.63% |

All Three Approaches Combined

### Observation and analysis: Except for the first combination, all other combined approaches outperformed the individual modifications. The best combination was adding a hidden layer (Dim = 10) + Frequency Cutoff = 0, as it did not significantly increase overfitting, maintained stable validation accuracy, and achieved the highest test accuracy of 88.54%. This experiment also highlights that more modifications do not necessarily lead to better performance—finding the most suitable combination is key.

# Section 2

## Task 2: Modifying the output layer to 2 Nodes

### To modify the output layer to have two output nodes, the first step is to update the code to change the output feature size to 2. Modify the **constructor** of ReviewClassifier as follows:

* Before:  
  self.fc2 = nn.Linear(in\_features=hidden\_dim, out\_features=1)
* After:

self.fc2 = nn.Linear(in\_features=hidden\_dim, out\_features=**2**)

### Since the sigmoid function can only return a value in the range of (0,1), it is not suitable for mulit-class classification tasks. Therefore, the **forward** function of ReviewClassifier has been modification to output the results directly, as shown below:

* Before:  
  return torch.sigmoid(y\_out).squeeze()
* After:

return y\_out

### In the compute\_accuracy function, since y\_pred now has two values, we need to use the max function to find the maximum value and obtain its index, then transfer it to CPU memory. The previously used .long() function is no longer needed because the index values are already integers, unlike (y\_pred > 0.5), which produces boolean values that require explicit conversion to integers. The specific modification is as follows:

* Before:  
  y\_pred\_indices = (y\_pred>0.5).cpu().long()
* After:

y\_pred\_indices = y\_pred.max(dim=1).indices.cpu()

### Similarly to (3), the code for running the test dataset also needs to be modified:

* Before:  
  y\_pred\_list.extend((y\_pred>0.5).cpu().long().numpy())
* After:

y\_pred\_list.extend(y\_pred.max(dim=1).indices.cpu().numpy())

### Binary Cross Entropy Loss (BCELoss) measures the loss between the target and the input probabilities, making it unsuitable for multi-class classification. Therefore, CrossEntropyLoss, which is specifically designed for multi-class classification, is introduced. The specific modification is as follows:

* Before:  
  loss\_func = nn.BCELoss()
* After:

loss\_func = nn.CrossEntropyLoss()

### CrossEntropyLoss requires class indices (integer values) as y\_target, so the original code that forcibly converted it to float for BCELoss has been removed. All relevant code modifications are as follows:

* Before:  
  loss = loss\_func(y\_pred, batch\_dict['y\_target'].float())
* After:

loss = loss\_func(y\_pred, batch\_dict['y\_target'])

### In the final predict\_rating function, for the same reasons as (3) and (4), the ReviewClassifier now directly outputs two values (i.e., positive and negative). Therefore, there is no need to use a threshold to determine whether the prediction is positive. The modification is as follows:

* Before:  
  probability\_value = result.item()

index=1

if probability\_value < decision\_threshold:

index = 0

* After:

index = result.max(dim=1).indices.item()