NLP202 assignment 1

Cheng-Tse Liu

February 2, 2024

1 Introduction

The objective of the assigned homework is to gain a comprehensive understanding of minibatching implementation in PyTorch. The task involves the development of both a logistic regression model and a Long Short-Term Memory (LSTM) model within the PyTorch framework, specifically tailored for the sentiment analysis task using minibatching. Sentiment analysis, in this context, aims to ascertain whether a given text, such as a sentence, conveys a positive or negative sentiment. The IMDB reviews dataset, a commonly used benchmark in sentiment analysis, will serve as the foundational dataset for this exercise. It consists of movie reviews and is well-regarded in the field.

1.1 Data Pre-processing

We tokenized each line of data and set words that appear less than 3 times to be unknown tokens and add padding token. We set unknown token to be default index and transform the original labels to be binary labels. We used text transform function to change the text in original data to be the index of the vocabulary dictionary. We shuffle the data and split the data to have 7500 lines of data as dev dataset.

2 Logistic Regression

2.1 Question 1-1

Q: Are your minibatching output scores exactly the same as the single instanced model? Why or why not?
A: No, it's different. However, the scores are close to each other. It might because the batch size is different, so when the model is doing back propagation, it will have different gradient and different optimize result.

2.2 Question 1-2

Q: Make a plot to compare training time of different batch sizes.
A: As shown below in figure 1.

2.3 Question 1-3

Q: Make a plot to compare model accuracy on dev set of different batch sizes.
A: As shown below in figure 2.

2.4 Question 1-4

Q: Tune the learning rate for a specific batch size and make a plot to compare model accuracy on dev set of different learning rates.
A: As shown below in figure 3.

2.5 Question 1-5

Q: Report the accuracy of your best model on the dev and test sets after training.
A: As shown below in figure 4 and 5.

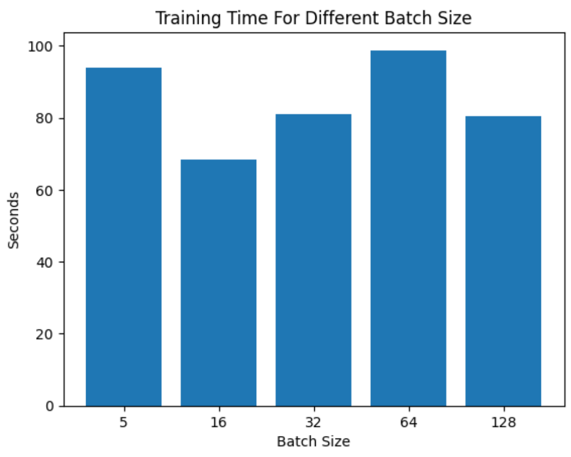


Figure 1: Training Time For Different Batch Size

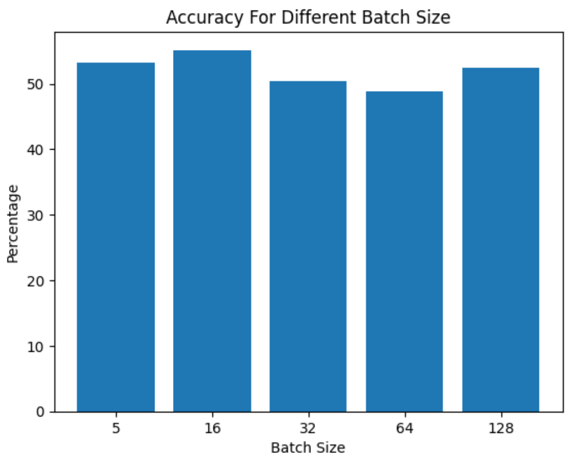


Figure 2: Accuracy For Different Batch Size

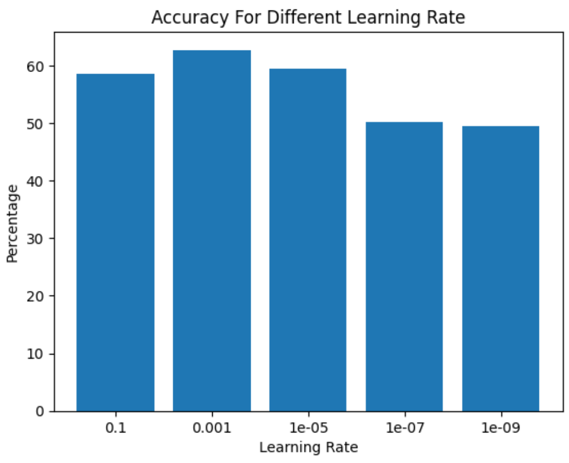


Figure 3: Accuracy For Different Learning Rate

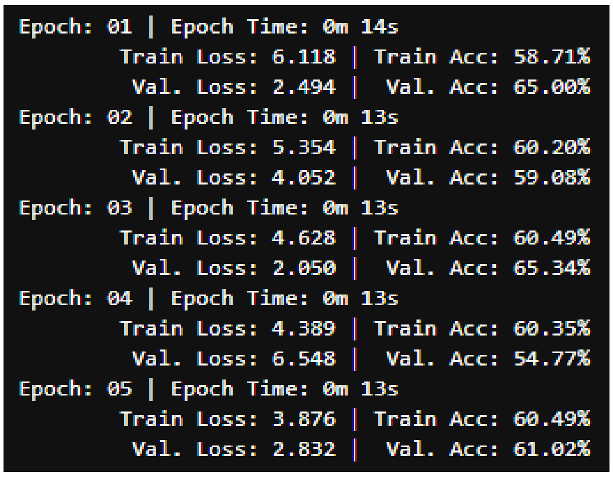
2.6 Question 1-7

Q: Look at the output on the dev set and compare to the gold labels. What are your observations about the errors your model makes?
A: It's hard to tell what might cause the model to make such models. Because when I look at the data that my model has made wrong prediction. The comment data is full of nonsense or not logical statements. For instance, some of the comment is using really roundabout way to describe the content of that movie. It's already hard for human to tell the sentiment of that kind of data needless to say for model to recognize those. Other situation of model making wrong predictions will be like the comment data contains really positive and negative content at the same time. Those kind of noise in data effect the model to recognize the sentiment label.

3 Long Short-term Memory

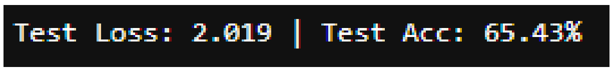
3.1 Question 2-1

Q: Are your minibatching output scores exactly the same as the single instanced model? Why or why not?
A: No, it's different. However, the scores are close to each other. It might because the batch size is different, so when the model is doing back propagation, it will have different gradient and different optimize result.



```
Epoch: 01 | Epoch Time: 0m 14s
        Train Loss: 6.118 | Train Acc: 58.71%
         Val. Loss: 2.494 |  Val. Acc: 65.00%
Epoch: 02 | Epoch Time: 0m 13s
        Train Loss: 5.354 | Train Acc: 60.20%
         Val. Loss: 4.052 |  Val. Acc: 59.08%
Epoch: 03 | Epoch Time: 0m 13s
        Train Loss: 4.628 | Train Acc: 60.49%
         Val. Loss: 2.050 |  Val. Acc: 65.34%
Epoch: 04 | Epoch Time: 0m 13s
        Train Loss: 4.389 | Train Acc: 60.35%
         Val. Loss: 6.548 |  Val. Acc: 54.77%
Epoch: 05 | Epoch Time: 0m 13s
        Train Loss: 3.876 | Train Acc: 60.49%
         Val. Loss: 2.832 |  Val. Acc: 61.02%
```

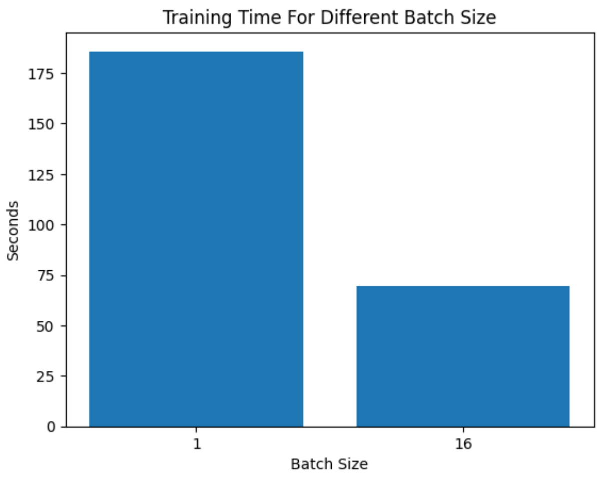
(a) Accuracy For my best model on dev set



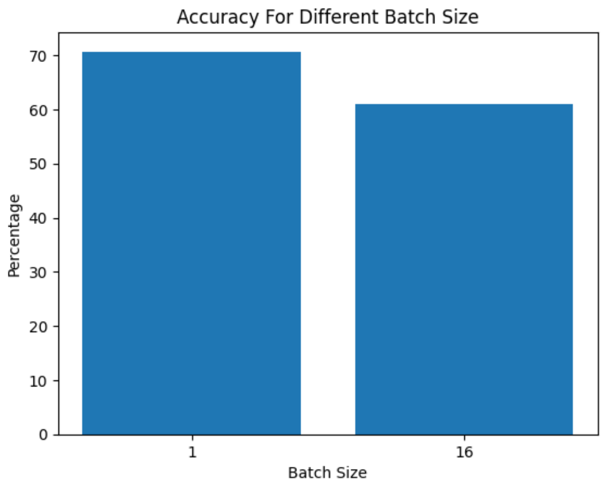
```
Test Loss: 2.019 | Test Acc: 65.43%
```

(b) Image 2

Figure 4: Accuracy For my best LR model on dev and test set



(a) Training Time Comparison between batch size 1 and 16



(b) Image 2

Figure 5: Accuracy Comparison between batch size 1 and 16

3.2 Question 2-2

Q: Make a plot to compare training time of different batch sizes.
A: As shown below in figure 6. We can tell that lower the batch size is longer the training time will be. It's because the smaller the batch size is the more batches that will be passed in to the model to train. Therefore, it takes more time to finish training.

3.3 Question 2-3

Q: Make a plot to compare model accuracy on dev set of different batch sizes.
A: As shown below in figure 7. We can tell that for different batch size the accuracy are all around 86 percent. which is a little bit weird to us, because we assumed that there should be some difference. It might because of the model is to robust or the data is a lot so either way the model is well trained.

3.4 Question 2-4

Q: Tune the learning rate for a specific batch size and make a plot to compare model accuracy on dev set of different learning rates.

A: As shown below in figure 8.

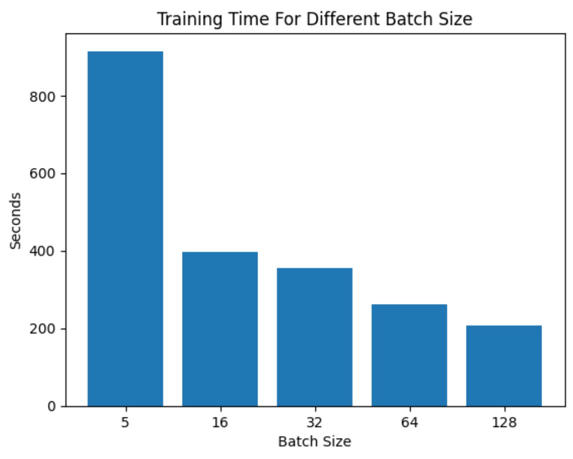


Figure 6: Training Time For Different Batch Size

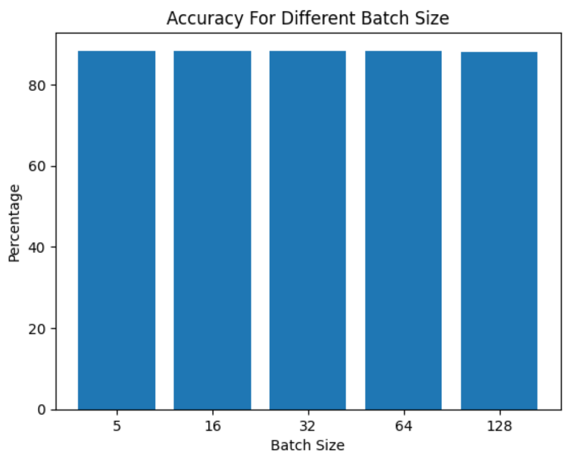


Figure 7: Accuracy For Different Batch Size

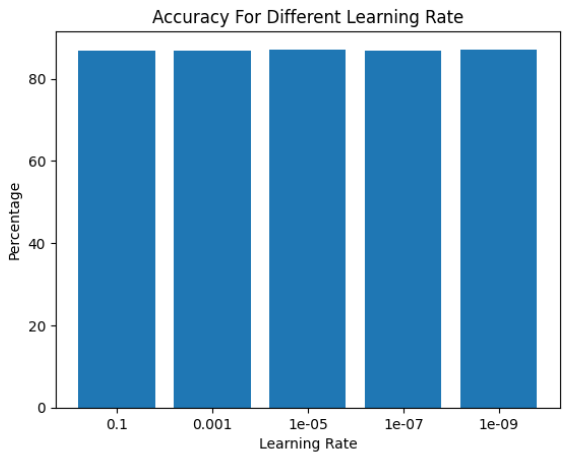


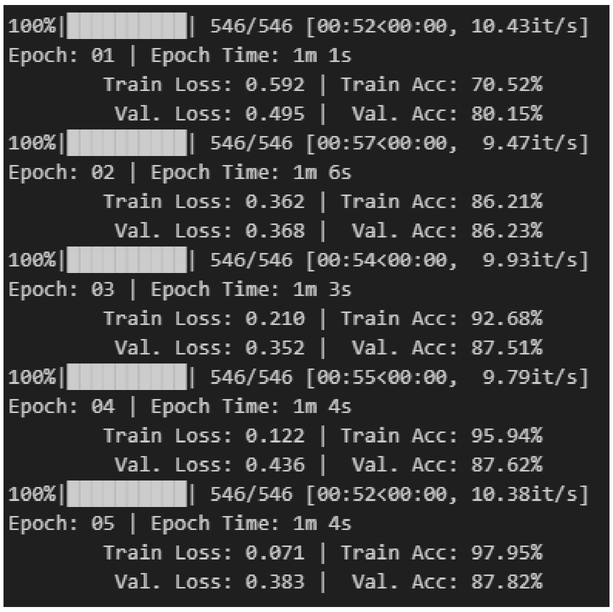
Figure 8: Accuracy For Different Learning Rate

3.5 Question 2-5

Q: Report the accuracy of your best model on the dev and test sets after training.
A: As shown below in figure 9 and 10.

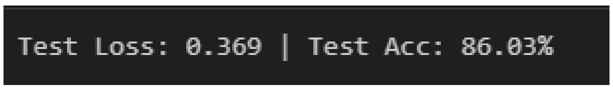
3.6 Question 2-7

Q: Look at the output on the dev set and compare to the gold labels. What are your observations about the errors your model makes?
A: It's hard to tell what might cause the model to make such models. Because when I look at the data that my model has made wrong prediction. The comment data is full of nonsense or not logical statements. For instance, some of the comment is using really roundabout way to describe the content of that movie. It's already hard for human to tell the sentiment of that kind of data needless to say for model to recognize those. Other situation of model making wrong predictions will be like the comment data contains really positive and negative content at the same time. Those kind of noise in data effect the model to recognize the sentiment label.



```
100%|██████████| 546/546 [00:52<00:00, 10.43it/s]
Epoch: 01 | Epoch Time: 1m 1s
        Train Loss: 0.592 | Train Acc: 70.52%
         Val. Loss: 0.495 |  Val. Acc: 80.15%
100%|██████████| 546/546 [00:57<00:00,  9.47it/s]
Epoch: 02 | Epoch Time: 1m 6s
        Train Loss: 0.362 | Train Acc: 86.21%
         Val. Loss: 0.368 |  Val. Acc: 86.23%
100%|██████████| 546/546 [00:54<00:00,  9.93it/s]
Epoch: 03 | Epoch Time: 1m 3s
        Train Loss: 0.210 | Train Acc: 92.68%
         Val. Loss: 0.352 |  Val. Acc: 87.51%
100%|██████████| 546/546 [00:55<00:00,  9.79it/s]
Epoch: 04 | Epoch Time: 1m 4s
        Train Loss: 0.122 | Train Acc: 95.94%
         Val. Loss: 0.436 |  Val. Acc: 87.62%
100%|██████████| 546/546 [00:52<00:00, 10.38it/s]
Epoch: 05 | Epoch Time: 1m 4s
        Train Loss: 0.071 | Train Acc: 97.95%
         Val. Loss: 0.383 |  Val. Acc: 87.82%
```

(a) Accuracy For my best model on dev set



```
Test Loss: 0.369 | Test Acc: 86.03%
```

(b) Image 2

Figure 9: Accuracy For my best LSTM model on dev and test set

4 Discussion

Theoretically, the training time should be shorter when the batch size is bigger since it's parsing more data to the model per batch. However, the plots above are not in this pattern. One of the possible reasons is that the number of epochs is not big enough so the model hasn't converge to a certain point, causing the pattern of the figures are not accurate. Therefore, we mad two plots to display the pattern. For instance, we have two plots that displays the difference between batch size equals to 1 and batch size equals to 16. As it displays on figure 6, we can tell that the training time for batch size equals to 1 is really long because batch size equals to 1 means it's parsing one line of data per batch which is more inefficient. Let's look at figure 7, the accuracy didn't drop a lot. Consider the time we saved by setting batch size to 16. The drop of accuracy is acceptable.