Deep Learning Lab #3

Emily Bates, xzz320

Task 1: Add code below to preprocess the following cyberbullying text, and include the generated tokens in your report.
"Harlem shake is just an excuse to go full retard for 30 seconds".

```
# TODO: Enter your code here
sentences = "Harlem shake is just an excuse to go full retard for 30
seconds"

tokens = tokenizer.tokenize(sentences)
for token in tokens:
    print(token)
```

Output:

```
harlem
shake
is
just
an
excuse
to
go
full
re
##tar
##d
for
30
Seconds
```

Task 2: After training, what is the training accuracy that your model achieves?
The training accuracy achieved is 98.78%

```
#TODO: add your code below to print the final training accuracy out
print(f'| Epoch: {epoch+1:02} | Train Loss: {train_loss:.3f} | Train Acc:
{train_acc*100:.2f}% | Val. Loss: {valid_loss:.3f} | Val. Acc:
{valid_acc*100:.2f}% |')
```

```
| Epoch: 05 | Train Loss: 0.043 | Train Acc: 98.78%
| Val. Loss: 0.598 | Val. Acc: 85.94% |
```

Task 3: **Let's review the previous code then finish the next code cell**

```
#TODO: complete the code below
test_loss, test_acc = evaluate(model, test_data_loader, criterion)
print(f'| Test Loss: {test_loss:.3f} | Test Acc: {test_acc*100:.2f}% |')
```

```
| Test Loss: 0.685 | Test Acc: 82.92% |
```

Task 4: **Use the samples in [this file] and your model to detect the cyberbullying samples**

```
#TODO: complete the code below
text1 = "you guys are a bunch of losers, fuck you"
ret1 = predict_cb(text1)
print("Sample prediction: ", ret1[2], f'Confidence: {ret1[0].item() *
100:.2f}%')

print("=========================================")

text2 = "I'm never going to see your little pathetic self again"
ret2 = predict_cb(text2)
print("Sample prediction: ", ret2[2], f'Confidence: {ret2[0].item() *
100:.2f}%')

print("=========================================")

text3 = "She looks really nice today!"
ret3 = predict_cb(text3)
print("Sample prediction: ", ret3[2], f'Confidence: {ret3[0].item() *
100:.2f}%')
```

```
Sample prediction:  Cyberbullying detected.
Confidence: 99.85%
=========================================
Sample prediction:  Cyberbullying detected.
Confidence: 99.82%
=========================================
Sample prediction:  Cyberbullying not detected.
Confidence: 99.82%
```

```
#@title Input your own sentence to check the prediction.
```

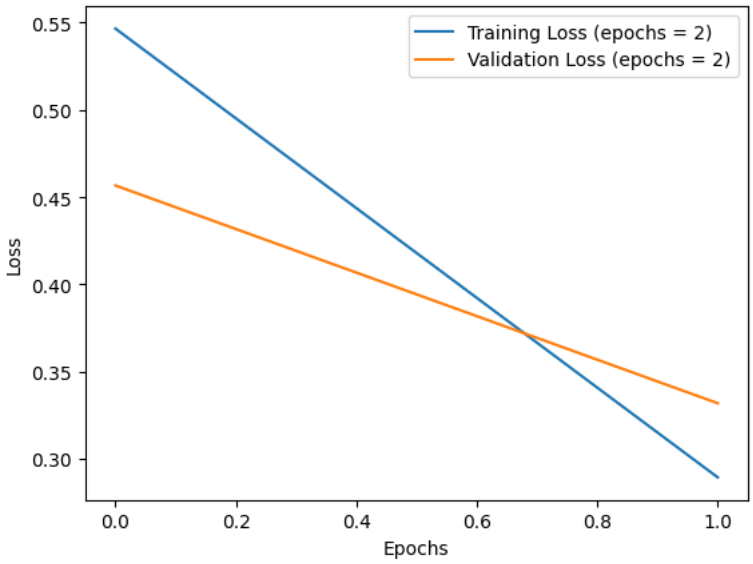
```
My_Sentence = 'Data Analytics is fun' #@param {type:"string"}

ret = predict_cb(My_Sentence)
print("===========The Model Prediciton is===========")
print("The input sentence is: ", ret[2], f'Confidence: {ret[0].item() *
100:.2f}%')
```

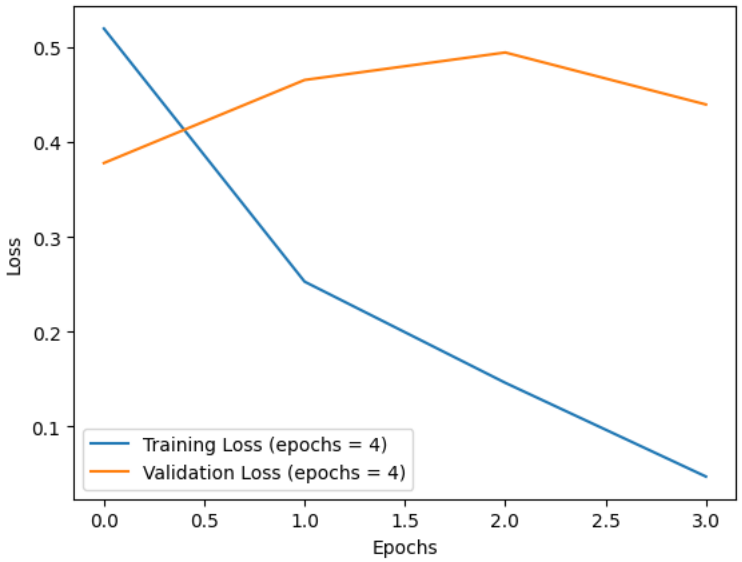
```
===========The Model Prediciton is===========
The input sentence is:  Cyberbullying not detected.
Confidence: 99.85%
```

Task 5: Compare different training epochs with 2, 10. You can try more different settings and find a suitable epoch number.

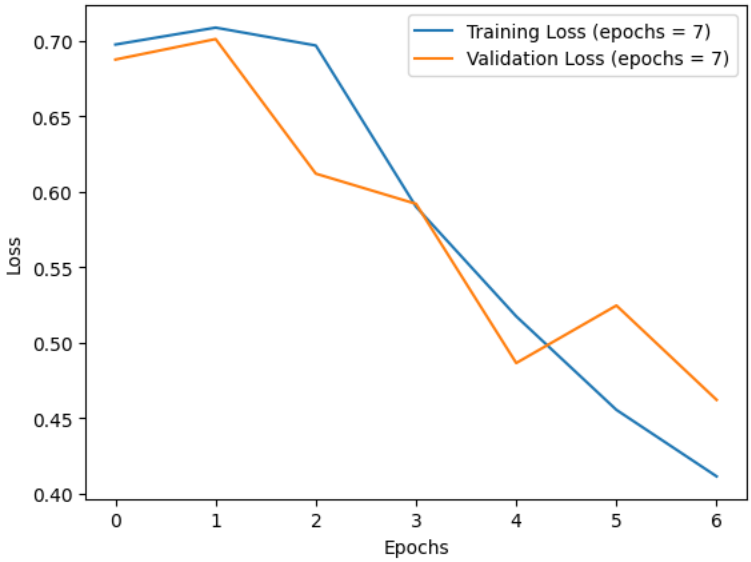
```
Training for 2 epochs: Test Loss: 0.396 | Test Acc:
85.10% |
```



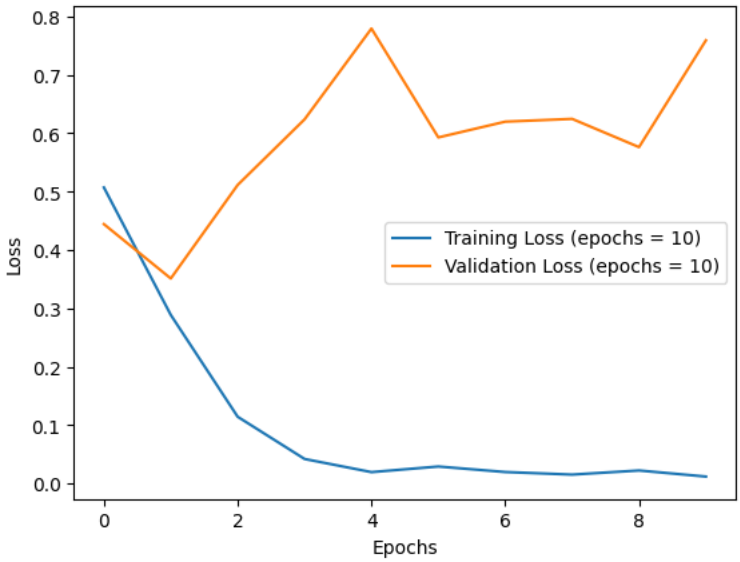

```
Training for 4 epochs: Test Loss: 0.587 | Test Acc:
85.10% |
```



Training for 7 epochs: Test Loss: 0.473 | Test Acc: 80.52% |



Training for 10 epochs: Test Loss: 0.714 | Test Acc: 83.54% |



In the 2 epochs graph, you can see that the training loss and validation loss are both decreasing gradually, which may indicate underfitting. In the 10 epoch graph, we see the training loss declining while the validation loss increases, suggesting overfitting. 4 or 7 epochs might be a suitable balance between underfitting and overfitting.

Task 6: Compare different learning rates with 0.1, 1e-3 and 1e-5. You can try with your own settings and find the best learning rate.

```
Learning rate: 0.01 Test Loss: 1.589 | Test Acc:
50.00% |
Learning rate: 0.001 Test Loss: 0.788 | Test Acc:
50.00% |
Learning rate: 1e-05 Test Loss: 0.499 | Test Acc:
83.23% |
Learning rate: 1e-07 Test Loss: 0.653 | Test Acc:
65.73% |
```

The best learning rate appears to be 1e-05 because it has the smallest test loss and the highest test accuracy.

Task 7: Discussion

We experimented with different hyperparameters in this lab, what can you conclude about training AI models? Specifically, what are your observations about the model before Vs. after hyperparameter tuning?

It's important to understand the model's performance metrics - training loss, validation loss, and accuracy during training to make decisions about hyperparameter adjustments.

Increasing the number of epochs generally allows the model to see the training data more times, potentially improving its performance up to a certain point. However, beyond a certain threshold, the model starts overfitting. We saw that a happy middle between 4 and 7 would be best.

The learning rate controls the size of the steps taken during optimization. A too high learning rate might cause the model to overshoot the optimal solution, whereas a too low learning rate might cause slow convergence or getting stuck in local minima. The results indicate that a learning rate of 1e-05 performed the best among the tested learning rates, as it achieved the lowest test loss and the highest test accuracy.

By finding the optimal combination of hyperparameters, the model is better able to generalize to unseen data and achieve higher accuracy. After hyperparameter tuning, AI models should perform better!