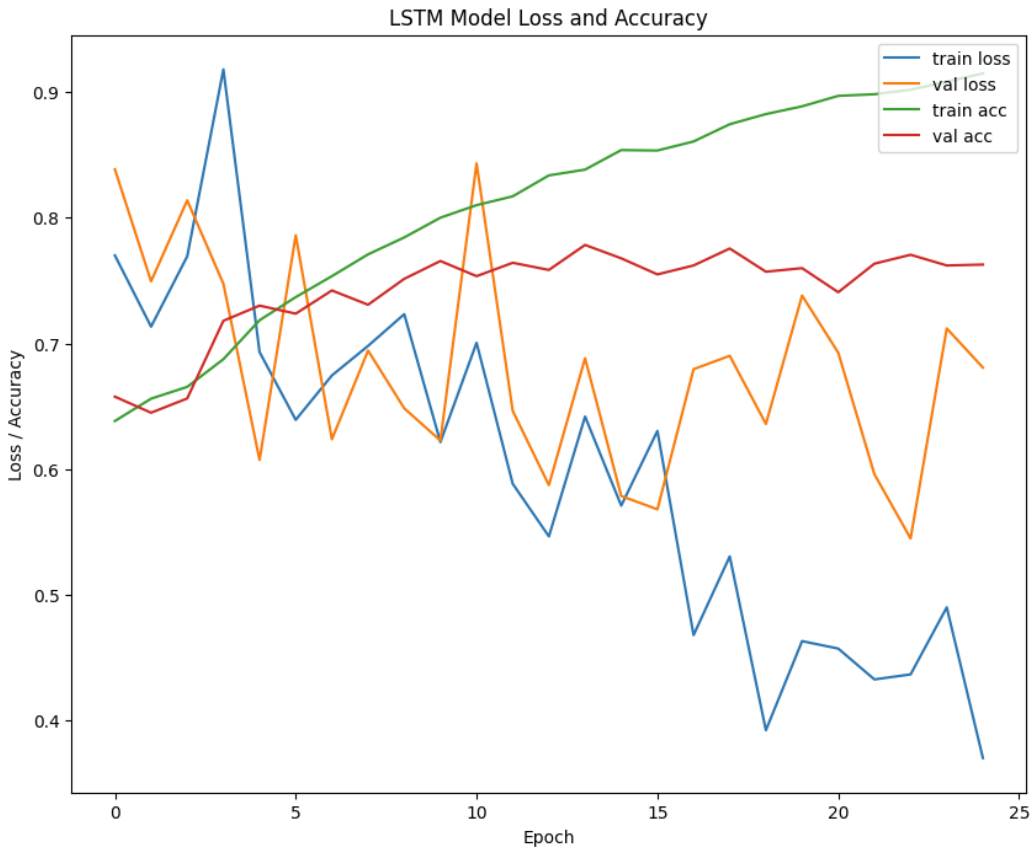
Shea Durgin & Loic Sine

https://github.com/sheaDurgin/NLP23/tree/main/Lab7

My LSTM model has an average 75.3% accuracy on the test set. I took an average since every time you run the test code it can give slightly different results. I determined an average was a fair answer to report.

My parameters are as such. Sequence Length stayed at 16, this is because I checked the average length for each tweet in all the sets, and they all came to 16.xxx lengths. Thus I determined that 16 is a value that just makes sense considering the data (also consistently gave higher scores in my testing). The batch size is something I tested with a lot. I ultimately decided on 128 as going to 64 would decrease my accuracy by a few % and the graphs looked worse when putting it up to 256, however the score difference was not as drastic as 64. I kept the layers of the lstm to 1, as I was not getting better results when bumping the numbers up (3 layers was better for me when I was still using an RNN though). My hidden_dim mainly hovered around either 64 and 128 as everytime I made it more or smaller than those two values, my score would seemingly tank. I ended up on 128 as it just seemed more consistent in giving me good scores. My dropout is 0.4, which is in the normal range of 0.2-0.5, it was the lowest I could go without having my scores go down. I set the model to use bidirectionality, it always seemed to just bump my numbers up so I didn't change it much after noticing. I stuck with CrossEntropyLoss as my research suggested it was the ideal for this type of classification and I changed it once and got not so great results.My optimizer is Adam with a learning rate of 0.004 and a weight decay of 0.001. I did a lot of testing with this and this is what really shot my score up past the 72% barrier I was stuck at for a while. Weight decay helped a bit with overfitting (although it is still doing that if you look at the graph) and learning rate is ideal for the amount of epochs I am running. My best epoch is usually in the 10 - 20 range, I am saving models based on best validation accuracy.




```
[ ]  batch_acc = []
     acc_total = 0
     for _ in range(100):
       for batch_idx, batch in enumerate(test_dl):
           input = batch[0].to(device)
           target = batch[1].to(device)
           optimizer.zero_grad()
           with torch.set_grad_enabled(False):
               out, hidden = best_model(input, h0)
               target = target.argmax(dim=1)
               preds = out.argmax(dim=1)
               batch_acc.append(accuracy_score(preds.cpu(), target.cpu()))
       acc = sum(batch_acc) / len(batch_acc)
       acc_total += acc
     avg = acc_total / 100
     print(f"Average prediction accuracy: {avg}")

     Average prediction accuracy: 0.7531444452733471
```