How to run my program in order:

1. Run these 3 in the console in the directory with my code (make sure you are outside all of these folders and not in any like src or data or anything:

1. conda create -n imdb-rnn python=3.10 -y
2. conda activate imdb-rnn
3. pip install -r requirements.txt

Then:

2. **python -m src.preprocess --csv data/IMDB_Dataset.csv --out_dir data/processed --vocab_size 10000**

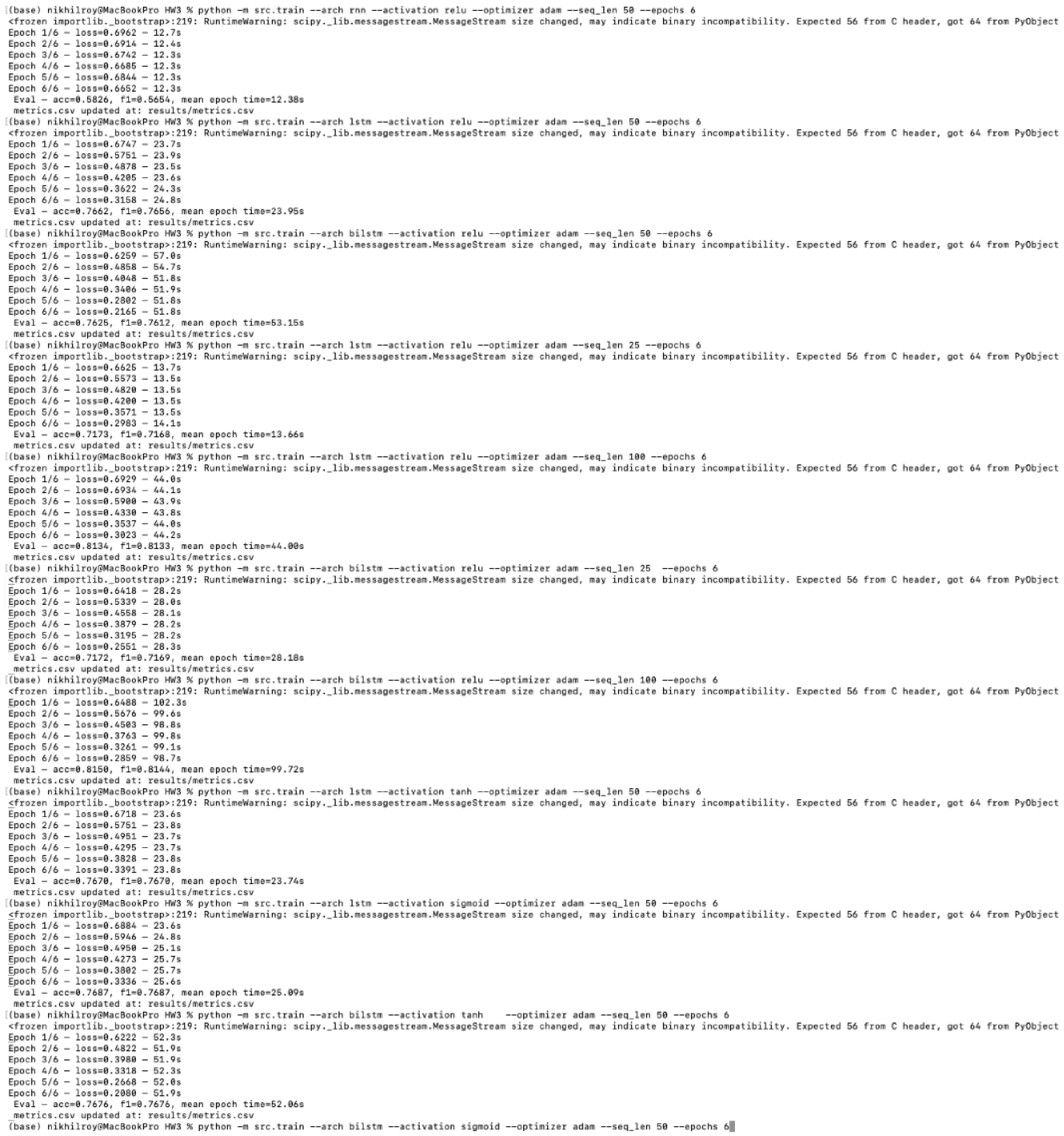
3. **Check the comments in the train.py code to see the several ways to execute the program with varying the different hyperparemeters**

4. **python -m src.evaluate**
    a. You can open up evaluate.py and see my docstring at top to view the several commands and ways of executing the program with each doing a slightly different thing (like sorting a specific column or generating optimal models/plots, etc.)

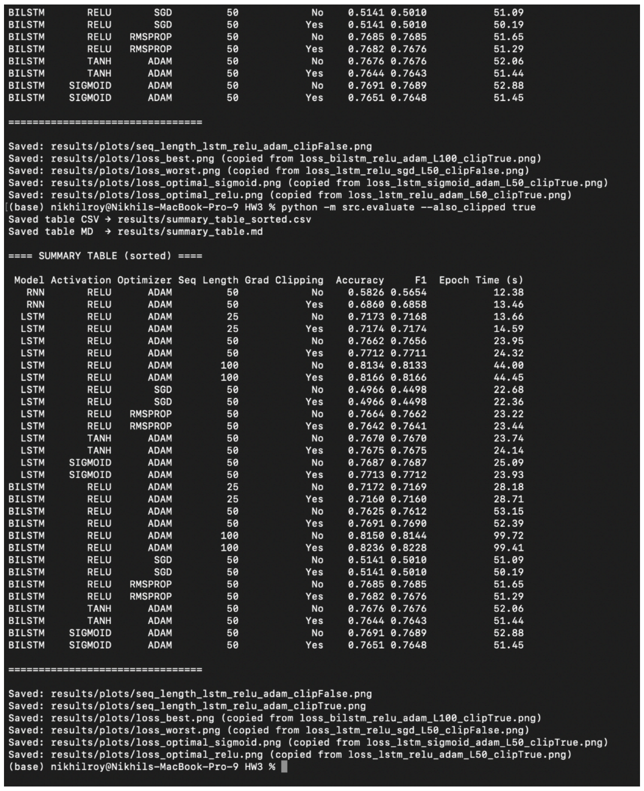
I used Apple M1 chip (8-core CPU) with 16 GB RAM

All seeds have been set to 42

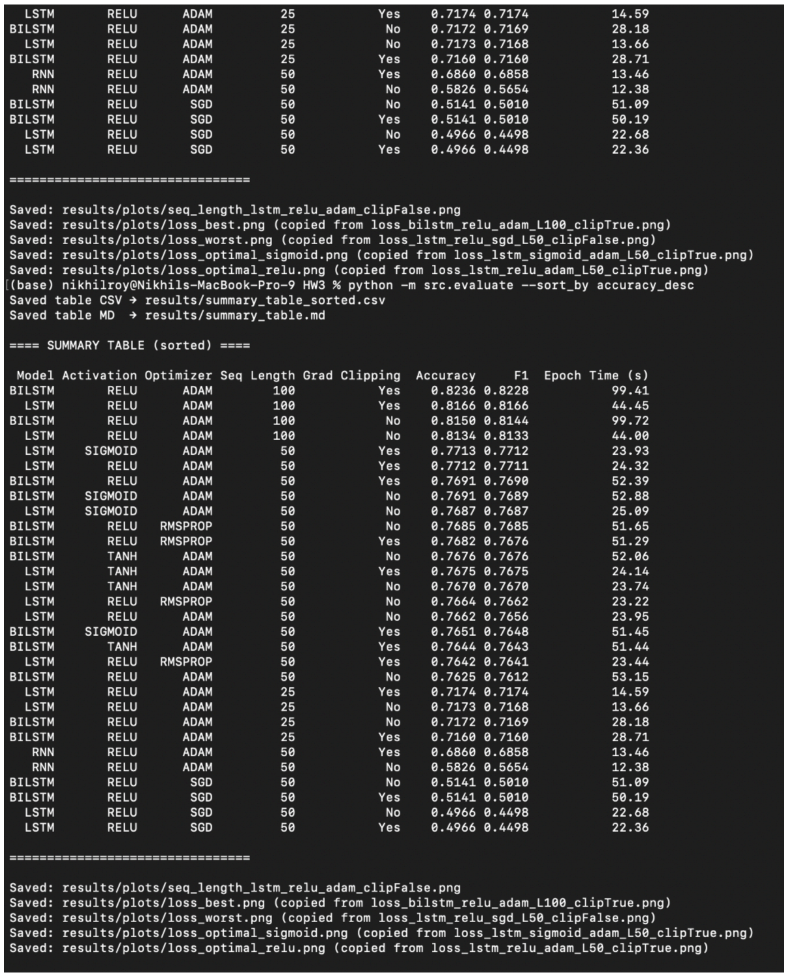
Examples of Executing My Program:



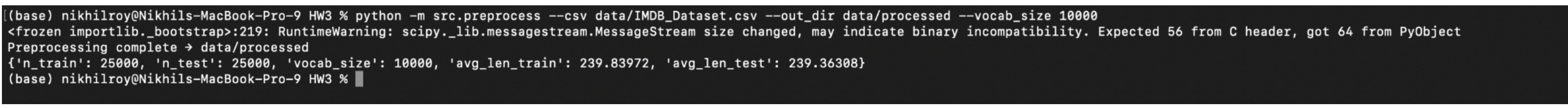
```
(base) nikhilroy@MacBookPro HW3 % python -m src.train --arch rnn --activation relu --optimizer adam --seq_len 50 --epochs 6
<frozen importlib._bootstrap>:219: RuntimeWarning: scipy._lib.messagestream.MessageStream size changed, may indicate binary incompatibility. Expected 56 from C header, got 64 from PyObject
Epoch 1/6 — loss=0.6962 — 12.7s
Epoch 2/6 — loss=0.6914 — 12.4s
Epoch 3/6 — loss=0.6742 — 12.3s
Epoch 4/6 — loss=0.6685 — 12.3s
Epoch 5/6 — loss=0.6844 — 12.3s
Epoch 6/6 — loss=0.6652 — 12.3s
 Eval — acc=0.5826, f1=0.5654, mean epoch time=12.38s
 metrics.csv updated at: results/metrics.csv
(base) nikhilroy@MacBookPro HW3 % python -m src.train --arch lstm --activation relu --optimizer adam --seq_len 50 --epochs 6
<frozen importlib._bootstrap>:219: RuntimeWarning: scipy._lib.messagestream.MessageStream size changed, may indicate binary incompatibility. Expected 56 from C header, got 64 from PyObject
Epoch 1/6 — loss=0.6747 — 23.7s
Epoch 2/6 — loss=0.5751 — 23.9s
Epoch 3/6 — loss=0.4878 — 23.5s
Epoch 4/6 — loss=0.4205 — 23.6s
Epoch 5/6 — loss=0.3622 — 24.3s
Epoch 6/6 — loss=0.3158 — 24.8s
 Eval — acc=0.7662, f1=0.7656, mean epoch time=23.95s
 metrics.csv updated at: results/metrics.csv
(base) nikhilroy@MacBookPro HW3 % python -m src.train --arch bilstm --activation relu --optimizer adam --seq_len 50 --epochs 6
<frozen importlib._bootstrap>:219: RuntimeWarning: scipy._lib.messagestream.MessageStream size changed, may indicate binary incompatibility. Expected 56 from C header, got 64 from PyObject
Epoch 1/6 — loss=0.6259 — 57.0s
Epoch 2/6 — loss=0.4858 — 54.7s
Epoch 3/6 — loss=0.4048 — 51.8s
Epoch 4/6 — loss=0.3406 — 51.9s
Epoch 5/6 — loss=0.2802 — 51.8s
Epoch 6/6 — loss=0.2165 — 51.8s
 Eval — acc=0.7625, f1=0.7612, mean epoch time=53.15s
 metrics.csv updated at: results/metrics.csv
(base) nikhilroy@MacBookPro HW3 % python -m src.train --arch lstm --activation relu --optimizer adam --seq_len 25 --epochs 6
<frozen importlib._bootstrap>:219: RuntimeWarning: scipy._lib.messagestream.MessageStream size changed, may indicate binary incompatibility. Expected 56 from C header, got 64 from PyObject
Epoch 1/6 — loss=0.6625 — 13.7s
Epoch 2/6 — loss=0.5573 — 13.5s
Epoch 3/6 — loss=0.4820 — 13.5s
Epoch 4/6 — loss=0.4200 — 13.5s
Epoch 5/6 — loss=0.3571 — 13.5s
Epoch 6/6 — loss=0.2983 — 14.1s
 Eval — acc=0.7173, f1=0.7168, mean epoch time=13.66s
 metrics.csv updated at: results/metrics.csv
(base) nikhilroy@MacBookPro HW3 % python -m src.train --arch lstm --activation relu --optimizer adam --seq_len 100 --epochs 6
<frozen importlib._bootstrap>:219: RuntimeWarning: scipy._lib.messagestream.MessageStream size changed, may indicate binary incompatibility. Expected 56 from C header, got 64 from PyObject
Epoch 1/6 — loss=0.6929 — 44.0s
Epoch 2/6 — loss=0.6934 — 44.1s
Epoch 3/6 — loss=0.5900 — 43.9s
Epoch 4/6 — loss=0.4330 — 43.8s
Epoch 5/6 — loss=0.3537 — 44.0s
Epoch 6/6 — loss=0.3023 — 44.2s
 Eval — acc=0.8134, f1=0.8133, mean epoch time=44.00s
 metrics.csv updated at: results/metrics.csv
(base) nikhilroy@MacBookPro HW3 % python -m src.train --arch bilstm --activation relu --optimizer adam --seq_len 25  --epochs 6
<frozen importlib._bootstrap>:219: RuntimeWarning: scipy._lib.messagestream.MessageStream size changed, may indicate binary incompatibility. Expected 56 from C header, got 64 from PyObject
Epoch 1/6 — loss=0.6418 — 28.2s
Epoch 2/6 — loss=0.5339 — 28.0s
Epoch 3/6 — loss=0.4558 — 28.1s
Epoch 4/6 — loss=0.3879 — 28.2s
Epoch 5/6 — loss=0.3195 — 28.2s
Epoch 6/6 — loss=0.2551 — 28.3s
 Eval — acc=0.7172, f1=0.7169, mean epoch time=28.18s
 metrics.csv updated at: results/metrics.csv
(base) nikhilroy@MacBookPro HW3 % python -m src.train --arch bilstm --activation relu --optimizer adam --seq_len 100 --epochs 6
<frozen importlib._bootstrap>:219: RuntimeWarning: scipy._lib.messagestream.MessageStream size changed, may indicate binary incompatibility. Expected 56 from C header, got 64 from PyObject
Epoch 1/6 — loss=0.6488 — 102.3s
Epoch 2/6 — loss=0.5676 — 99.6s
Epoch 3/6 — loss=0.4503 — 98.8s
Epoch 4/6 — loss=0.3763 — 99.8s
Epoch 5/6 — loss=0.3261 — 99.1s
Epoch 6/6 — loss=0.2859 — 98.7s
 Eval — acc=0.8150, f1=0.8144, mean epoch time=99.72s
 metrics.csv updated at: results/metrics.csv
(base) nikhilroy@MacBookPro HW3 % python -m src.train --arch lstm --activation tanh --optimizer adam --seq_len 50 --epochs 6
<frozen importlib._bootstrap>:219: RuntimeWarning: scipy._lib.messagestream.MessageStream size changed, may indicate binary incompatibility. Expected 56 from C header, got 64 from PyObject
Epoch 1/6 — loss=0.6718 — 23.6s
Epoch 2/6 — loss=0.5751 — 23.8s
Epoch 3/6 — loss=0.4951 — 23.7s
Epoch 4/6 — loss=0.4295 — 23.7s
Epoch 5/6 — loss=0.3828 — 23.8s
Epoch 6/6 — loss=0.3391 — 23.8s
 Eval — acc=0.7670, f1=0.7670, mean epoch time=23.74s
 metrics.csv updated at: results/metrics.csv
(base) nikhilroy@MacBookPro HW3 % python -m src.train --arch lstm --activation sigmoid --optimizer adam --seq_len 50 --epochs 6
<frozen importlib._bootstrap>:219: RuntimeWarning: scipy._lib.messagestream.MessageStream size changed, may indicate binary incompatibility. Expected 56 from C header, got 64 from PyObject
Epoch 1/6 — loss=0.6884 — 23.6s
Epoch 2/6 — loss=0.5946 — 24.8s
Epoch 3/6 — loss=0.4950 — 25.1s
Epoch 4/6 — loss=0.4273 — 25.7s
Epoch 5/6 — loss=0.3802 — 25.7s
Epoch 6/6 — loss=0.3336 — 25.6s
 Eval — acc=0.7687, f1=0.7687, mean epoch time=25.09s
 metrics.csv updated at: results/metrics.csv
(base) nikhilroy@MacBookPro HW3 % python -m src.train --arch bilstm --activation tanh    --optimizer adam --seq_len 50 --epochs 6
<frozen importlib._bootstrap>:219: RuntimeWarning: scipy._lib.messagestream.MessageStream size changed, may indicate binary incompatibility. Expected 56 from C header, got 64 from PyObject
Epoch 1/6 — loss=0.6222 — 52.3s
Epoch 2/6 — loss=0.4822 — 51.9s
Epoch 3/6 — loss=0.3980 — 51.9s
Epoch 4/6 — loss=0.3318 — 52.3s
Epoch 5/6 — loss=0.2668 — 52.0s
Epoch 6/6 — loss=0.2080 — 51.9s
 Eval — acc=0.7676, f1=0.7676, mean epoch time=52.06s
 metrics.csv updated at: results/metrics.csv
(base) nikhilroy@MacBookPro HW3 % python -m src.train --arch bilstm --activation sigmoid --optimizer adam --seq_len 50 --epochs 6
```



```
BILSTM       RELU       SGD         50            No    0.5141 0.5010           51.09
BILSTM       RELU       SGD         50           Yes    0.5141 0.5010           50.19
BILSTM       RELU   RMSPROP         50            No    0.7685 0.7685           51.65
BILSTM       RELU   RMSPROP         50           Yes    0.7682 0.7676           51.29
BILSTM       TANH      ADAM         50            No    0.7676 0.7676           52.06
BILSTM       TANH      ADAM         50           Yes    0.7644 0.7643           51.44
BILSTM    SIGMOID      ADAM         50            No    0.7691 0.7689           52.88
BILSTM    SIGMOID      ADAM         50           Yes    0.7651 0.7648           51.45

================================

Saved: results/plots/seq_length_lstm_relu_adam_clipFalse.png
Saved: results/plots/loss_best.png (copied from loss_bilstm_relu_adam_L100_clipTrue.png)
Saved: results/plots/loss_worst.png (copied from loss_lstm_relu_sgd_L50_clipFalse.png)
Saved: results/plots/loss_optimal_sigmoid.png (copied from loss_lstm_sigmoid_adam_L50_clipTrue.png)
Saved: results/plots/loss_optimal_relu.png (copied from loss_lstm_relu_adam_L50_clipTrue.png)
(base) nikhilroy@Nikhils-MacBook-Pro-9 HW3 % python -m src.evaluate --also_clipped true
Saved table CSV → results/summary_table_sorted.csv
Saved table MD  → results/summary_table.md

==== SUMMARY TABLE (sorted) ====

 Model Activation Optimizer Seq Length Grad Clipping  Accuracy     F1  Epoch Time (s)
   RNN       RELU      ADAM         50            No    0.5826 0.5654           12.38
   RNN       RELU      ADAM         50           Yes    0.6860 0.6858           13.46
  LSTM       RELU      ADAM         25            No    0.7173 0.7168           13.66
  LSTM       RELU      ADAM         25           Yes    0.7174 0.7174           14.59
  LSTM       RELU      ADAM         50            No    0.7662 0.7656           23.95
  LSTM       RELU      ADAM         50           Yes    0.7712 0.7711           24.32
  LSTM       RELU      ADAM        100            No    0.8134 0.8133           44.00
  LSTM       RELU      ADAM        100           Yes    0.8166 0.8166           44.45
  LSTM       RELU       SGD         50            No    0.4966 0.4498           22.68
  LSTM       RELU       SGD         50           Yes    0.4966 0.4498           22.36
  LSTM       RELU   RMSPROP         50            No    0.7664 0.7662           23.22
  LSTM       RELU   RMSPROP         50           Yes    0.7642 0.7641           23.44
  LSTM       TANH      ADAM         50            No    0.7670 0.7670           23.74
  LSTM       TANH      ADAM         50           Yes    0.7675 0.7675           24.14
  LSTM    SIGMOID      ADAM         50            No    0.7687 0.7687           25.09
  LSTM    SIGMOID      ADAM         50           Yes    0.7713 0.7712           23.93
BILSTM       RELU      ADAM         25            No    0.7172 0.7169           28.18
BILSTM       RELU      ADAM         25           Yes    0.7160 0.7160           28.71
BILSTM       RELU      ADAM         50            No    0.7625 0.7612           53.15
BILSTM       RELU      ADAM         50           Yes    0.7691 0.7690           52.39
BILSTM       RELU      ADAM        100            No    0.8150 0.8144           99.72
BILSTM       RELU      ADAM        100           Yes    0.8236 0.8228           99.41
BILSTM       RELU       SGD         50            No    0.5141 0.5010           51.09
BILSTM       RELU       SGD         50           Yes    0.5141 0.5010           50.19
BILSTM       RELU   RMSPROP         50            No    0.7685 0.7685           51.65
BILSTM       RELU   RMSPROP         50           Yes    0.7682 0.7676           51.29
BILSTM       TANH      ADAM         50            No    0.7676 0.7676           52.06
BILSTM       TANH      ADAM         50           Yes    0.7644 0.7643           51.44
BILSTM    SIGMOID      ADAM         50            No    0.7691 0.7689           52.88
BILSTM    SIGMOID      ADAM         50           Yes    0.7651 0.7648           51.45

================================

Saved: results/plots/seq_length_lstm_relu_adam_clipFalse.png
Saved: results/plots/seq_length_lstm_relu_adam_clipTrue.png
Saved: results/plots/loss_best.png (copied from loss_bilstm_relu_adam_L100_clipTrue.png)
Saved: results/plots/loss_worst.png (copied from loss_lstm_relu_sgd_L50_clipFalse.png)
Saved: results/plots/loss_optimal_sigmoid.png (copied from loss_lstm_sigmoid_adam_L50_clipTrue.png)
Saved: results/plots/loss_optimal_relu.png (copied from loss_lstm_relu_adam_L50_clipTrue.png)
(base) nikhilroy@Nikhils-MacBook-Pro-9 HW3 %
```



```
  LSTM       RELU      ADAM         25           Yes    0.7174 0.7174           14.59
BILSTM       RELU      ADAM         25            No    0.7172 0.7169           28.18
  LSTM       RELU      ADAM         25            No    0.7173 0.7168           13.66
BILSTM       RELU      ADAM         25           Yes    0.7160 0.7160           28.71
   RNN       RELU      ADAM         50           Yes    0.6860 0.6858           13.46
   RNN       RELU      ADAM         50            No    0.5826 0.5654           12.38
BILSTM       RELU       SGD         50            No    0.5141 0.5010           51.09
BILSTM       RELU       SGD         50           Yes    0.5141 0.5010           50.19
  LSTM       RELU       SGD         50            No    0.4966 0.4498           22.68
  LSTM       RELU       SGD         50           Yes    0.4966 0.4498           22.36

================================

Saved: results/plots/seq_length_lstm_relu_adam_clipFalse.png
Saved: results/plots/loss_best.png (copied from loss_bilstm_relu_adam_L100_clipTrue.png)
Saved: results/plots/loss_worst.png (copied from loss_lstm_relu_sgd_L50_clipFalse.png)
Saved: results/plots/loss_optimal_sigmoid.png (copied from loss_lstm_sigmoid_adam_L50_clipTrue.png)
Saved: results/plots/loss_optimal_relu.png (copied from loss_lstm_relu_adam_L50_clipTrue.png)
(base) nikhilroy@Nikhils-MacBook-Pro-9 HW3 % python -m src.evaluate --sort_by accuracy_desc
Saved table CSV → results/summary_table_sorted.csv
Saved table MD  → results/summary_table.md

==== SUMMARY TABLE (sorted) ====

 Model Activation Optimizer Seq Length Grad Clipping  Accuracy     F1  Epoch Time (s)
BILSTM       RELU      ADAM        100           Yes    0.8236 0.8228           99.41
  LSTM       RELU      ADAM        100           Yes    0.8166 0.8166           44.45
BILSTM       RELU      ADAM        100            No    0.8150 0.8144           99.72
  LSTM       RELU      ADAM        100            No    0.8134 0.8133           44.00
  LSTM    SIGMOID      ADAM         50           Yes    0.7713 0.7712           23.93
  LSTM       RELU      ADAM         50           Yes    0.7712 0.7711           24.32
BILSTM       RELU      ADAM         50           Yes    0.7691 0.7690           52.39
BILSTM    SIGMOID      ADAM         50            No    0.7691 0.7689           52.88
  LSTM    SIGMOID      ADAM         50            No    0.7687 0.7687           25.09
BILSTM       RELU   RMSPROP         50            No    0.7685 0.7685           51.65
BILSTM       RELU   RMSPROP         50           Yes    0.7682 0.7676           51.29
BILSTM       TANH      ADAM         50            No    0.7676 0.7676           52.06
  LSTM       TANH      ADAM         50           Yes    0.7675 0.7675           24.14
  LSTM       TANH      ADAM         50            No    0.7670 0.7670           23.74
  LSTM       RELU   RMSPROP         50            No    0.7664 0.7662           23.22
  LSTM       RELU      ADAM         50            No    0.7662 0.7656           23.95
BILSTM    SIGMOID      ADAM         50           Yes    0.7651 0.7648           51.45
BILSTM       TANH      ADAM         50           Yes    0.7644 0.7643           51.44
  LSTM       RELU   RMSPROP         50           Yes    0.7642 0.7641           23.44
BILSTM       RELU      ADAM         50            No    0.7625 0.7612           53.15
  LSTM       RELU      ADAM         25           Yes    0.7174 0.7174           14.59
  LSTM       RELU      ADAM         25            No    0.7173 0.7168           13.66
BILSTM       RELU      ADAM         25            No    0.7172 0.7169           28.18
BILSTM       RELU      ADAM         25           Yes    0.7160 0.7160           28.71
   RNN       RELU      ADAM         50           Yes    0.6860 0.6858           13.46
   RNN       RELU      ADAM         50            No    0.5826 0.5654           12.38
BILSTM       RELU       SGD         50            No    0.5141 0.5010           51.09
BILSTM       RELU       SGD         50           Yes    0.5141 0.5010           50.19
  LSTM       RELU       SGD         50            No    0.4966 0.4498           22.68
  LSTM       RELU       SGD         50           Yes    0.4966 0.4498           22.36

================================

Saved: results/plots/seq_length_lstm_relu_adam_clipFalse.png
Saved: results/plots/loss_best.png (copied from loss_bilstm_relu_adam_L100_clipTrue.png)
Saved: results/plots/loss_worst.png (copied from loss_lstm_relu_sgd_L50_clipFalse.png)
Saved: results/plots/loss_optimal_sigmoid.png (copied from loss_lstm_sigmoid_adam_L50_clipTrue.png)
Saved: results/plots/loss_optimal_relu.png (copied from loss_lstm_relu_adam_L50_clipTrue.png)
```



```
(base) nikhilroy@Nikhils-MacBook-Pro-9 HW3 % python -m src.preprocess --csv data/IMDB_Dataset.csv --out_dir data/processed --vocab_size 10000
<frozen importlib._bootstrap>:219: RuntimeWarning: scipy._lib.messagestream.MessageStream size changed, may indicate binary incompatibility. Expected 56 from C header, got 64 from PyObject
Preprocessing complete → data/processed
{'n_train': 25000, 'n_test': 25000, 'vocab_size': 10000, 'avg_len_train': 239.83972, 'avg_len_test': 239.36308}
(base) nikhilroy@Nikhils-MacBook-Pro-9 HW3 %
```

Dataset Summary:

I used the IMDb movie review sentiment dataset with a standard 50k labeled split: 25,000 reviews for training and 25,000 for testing. Before modeling, I lowercased everything and tokenized the text into word-level tokens. I built a fixed vocabulary of the top 10,000 tokens from the training set only, and anything outside that list was mapped to an <unk> token. For batching and model inputs I also used a <pad> token, so every example becomes a clean, integer-encoded sequence composed of indices in [0, vocab_size) plus those two special tokens. After tokenization, I converted each review to its sequence of token IDs and stored the train/test splits as serialized arrays for fast loading. The raw reviews are fairly long on average: 239.83972 tokens for train and 239.36308 for test (so both splits are very similar in length distribution). Because I'm running on CPU and wanted a controlled comparison, I didn't try to match each review's native length. Instead, I ran fixed-length experiments at sequence lengths of 25, 50, and 100. That let me isolate the accuracy-versus-time trade-off without changing anything else about preprocessing or the model. For each setting, sequences longer than the target length were truncated on the right, and shorter ones were right-padded with <pad> so the batches were rectangular and easy to process. This kept all downstream code simple and made timing comparisons meaningful, since the only thing that changed across runs was the maximum sequence length. In short: lowercase → tokenize → build a 10k vocab from train → map to IDs with <unk>/<pad> → fix length at {25, 50, 100} via truncate/pad → save arrays for quick, reproducible training and evaluation.

Model Configuration:

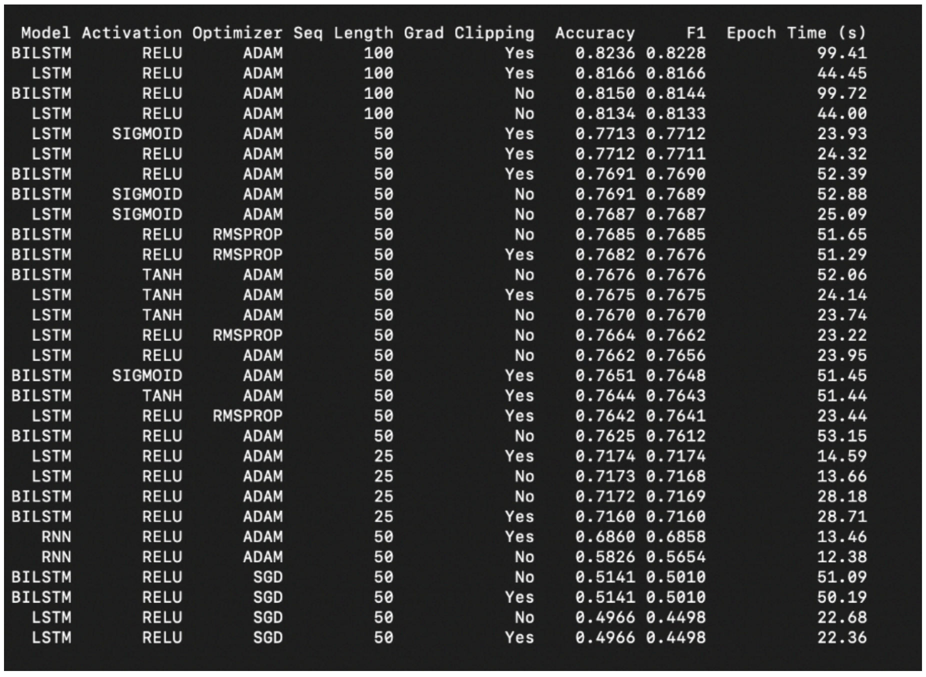
I kept the backbone hyperparameters fixed and only changed the knobs I was actually studying (architecture, activation, optimizer, sequence length, and whether gradient clipping was on). Every model used a single learnable word-embedding layer with an embedding dimension of 128, followed by a recurrent backbone and a simple linear classifier that maps the final sequence representation to the two sentiment labels. For the recurrent backbones, I evaluated three variants: a vanilla RNN, an LSTM, and a bidirectional LSTM (BiLSTM).

The vanilla RNN used 1 recurrent layer; the LSTM and BiLSTM used 2 recurrent layers. Hidden size was held at 128 throughout; for the BiLSTM this means the forward and backward directions each use 128 units and their outputs are concatenated before the classifier. I applied a modest dropout of 0.2 on the recurrent stack (between layers) to keep things stable on CPU without dragging training time too much. Unless otherwise noted, I trained with a batch size that comfortably fit CPU memory (no GPU was used), fixed sequence lengths of 25, 50, or 100 tokens (truncate longer reviews and pad shorter ones), and no learning-rate scheduler; the goal was to keep the training loop simple and make the timing numbers easy to compare. When “gradient clipping = Yes,” I used norm clipping with a max-norm of 1.0; otherwise clipping was disabled. For activations, I routed the pooled sequence representation through one of {ReLU, Tanh, Sigmoid} before the final linear layer so I could isolate how that nonlinearity interacts with the backbone. For the optimizers,

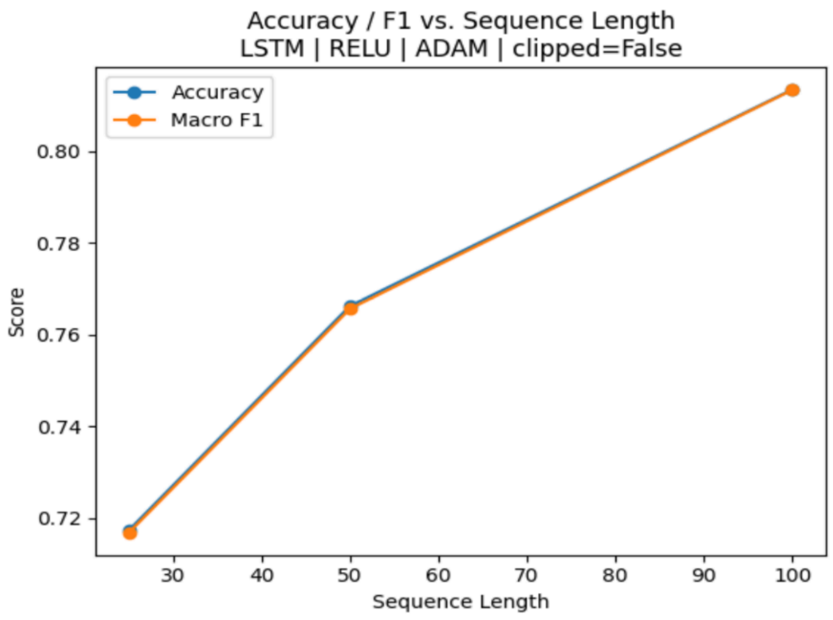
I ran three common choices with standard, CPU-friendly settings: Adam (learning rate 1e-3, β=(0.9, 0.999), weight decay 0), RMSprop (learning rate 1e-3, α=0.99), and SGD (learning rate 0.1, momentum 0.9, no weight decay). Loss was standard cross-entropy over the two classes. Across experiments, the only things I varied were: the backbone (RNN vs. LSTM vs. BiLSTM), the activation (ReLU/Tanh/Sigmoid), the optimizer (Adam/SGD/RMSprop), the fixed sequence length (25, 50, 100), and whether gradient clipping was turned on—everything else stayed fixed so the accuracy/F1/time comparisons are apples-to-apples.

Comparative Analysis:

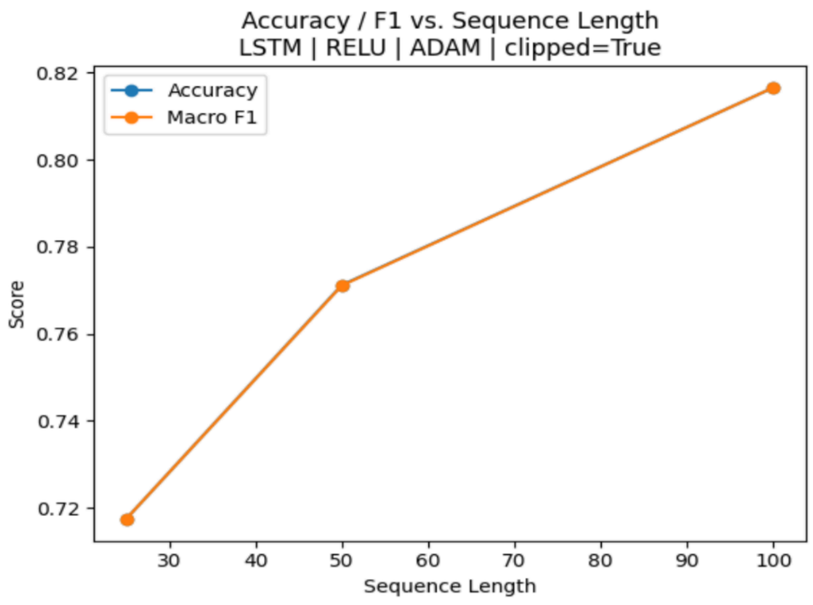
The clearest trend showed up when I held the settings to LSTM + ReLU + Adam and only changed sequence length. At 25 tokens, the model sat around 0.717 accuracy and 0.717 macro F1 in about 13.66 seconds per epoch. Moving to 50 tokens bumped me to roughly 0.771/0.771 while training time rose to about 24.32 seconds. Going to 100 tokens landed near 0.817/0.817 at about 44.45 seconds per epoch. So, longer sequences reliably helped accuracy and F1, and the time cost scaled in a very predictable way on CPU. Optimizers behaved as expected. Adam was the safest choice here: it converged faster and ended higher than SGD, and it had a small but repeatable edge over RMSProp in the LSTM runs. SGD struggled to get off the ground with these settings, which shows up clearly in the metrics table. RMSProp was competitive but not quite as strong as Adam. Gradient clipping helped most when the effective depth grew—either at longer sequence lengths or in the BiLSTM. It smoothed training and sometimes added a small bump to F1. At short sequences it was mostly neutral, which also matches intuition. Below are some example plots of the 3 models I found to be the most interesting based on looking at the performance. Also, you can see a screenshot of all my 30 different experiemnets (with 6 Epochs each) of running the different combinations of models with the optimizers, activations, and such where I change them 1 at a time. They are sorted by decreasing F1 score/accuracy, but in the beginning of the report, I mentioned the several ways to run my program (where you can see in evaluate.py docstrings at top)



```
 Model Activation Optimizer Seq Length Grad Clipping  Accuracy     F1  Epoch Time (s)
BILSTM       RELU      ADAM        100           Yes    0.8236 0.8228           99.41
  LSTM       RELU      ADAM        100           Yes    0.8166 0.8166           44.45
BILSTM       RELU      ADAM        100            No    0.8150 0.8144           99.72
  LSTM       RELU      ADAM        100            No    0.8134 0.8133           44.00
  LSTM    SIGMOID      ADAM         50           Yes    0.7713 0.7712           23.93
  LSTM       RELU      ADAM         50           Yes    0.7712 0.7711           24.32
BILSTM       RELU      ADAM         50           Yes    0.7691 0.7690           52.39
BILSTM    SIGMOID      ADAM         50            No    0.7691 0.7689           52.88
  LSTM    SIGMOID      ADAM         50            No    0.7687 0.7687           25.09
BILSTM       RELU   RMSPROP         50            No    0.7685 0.7685           51.65
BILSTM       RELU   RMSPROP         50           Yes    0.7682 0.7676           51.29
BILSTM       TANH      ADAM         50            No    0.7676 0.7676           52.06
  LSTM       TANH      ADAM         50           Yes    0.7675 0.7675           24.14
  LSTM       TANH      ADAM         50            No    0.7670 0.7670           23.74
  LSTM       RELU   RMSPROP         50            No    0.7664 0.7662           23.22
  LSTM       RELU      ADAM         50            No    0.7662 0.7656           23.95
BILSTM    SIGMOID      ADAM         50           Yes    0.7651 0.7648           51.45
BILSTM       TANH      ADAM         50           Yes    0.7644 0.7643           51.44
  LSTM       RELU   RMSPROP         50           Yes    0.7642 0.7641           23.44
BILSTM       RELU      ADAM         50            No    0.7625 0.7612           53.15
  LSTM       RELU      ADAM         25           Yes    0.7174 0.7174           14.59
  LSTM       RELU      ADAM         25            No    0.7173 0.7168           13.66
BILSTM       RELU      ADAM         25            No    0.7172 0.7169           28.18
BILSTM       RELU      ADAM         25           Yes    0.7160 0.7160           28.71
   RNN       RELU      ADAM         50           Yes    0.6860 0.6858           13.46
   RNN       RELU      ADAM         50            No    0.5826 0.5654           12.38
BILSTM       RELU       SGD         50            No    0.5141 0.5010           51.09
BILSTM       RELU       SGD         50           Yes    0.5141 0.5010           50.19
  LSTM       RELU       SGD         50            No    0.4966 0.4498           22.68
  LSTM       RELU       SGD         50           Yes    0.4966 0.4498           22.36
```

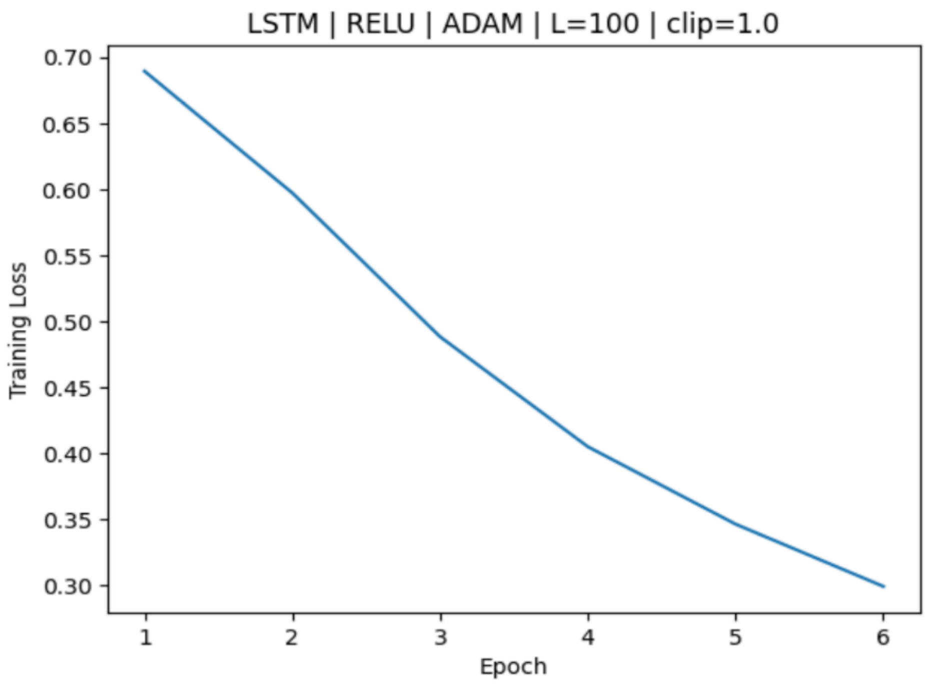


Accuracy/F1 vs Sequence Length for LSTM + ReLU + Adam (no clipping).



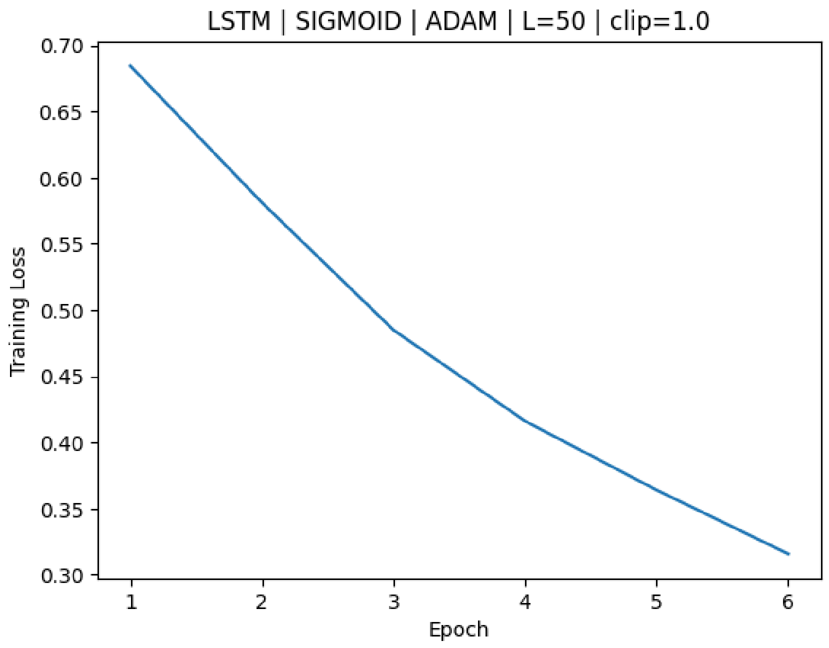
Accuracy/F1 vs Sequence Length for LSTM + ReLU + Adam (with clipping).

Optimal Picks:

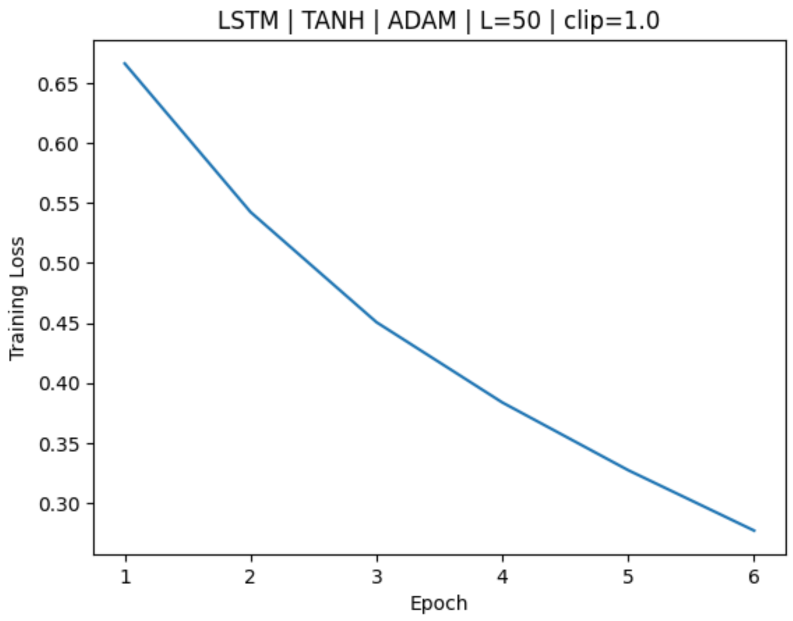


LSTM + ReLU + Adam | Length 100 | Clipping on.

For ReLU, my optimal configuration is an LSTM with Adam, sequence length 100, and gradient clipping enabled; it achieves accuracy 0.8166 and F1 0.8166 with an average epoch time of ~44.45 seconds.



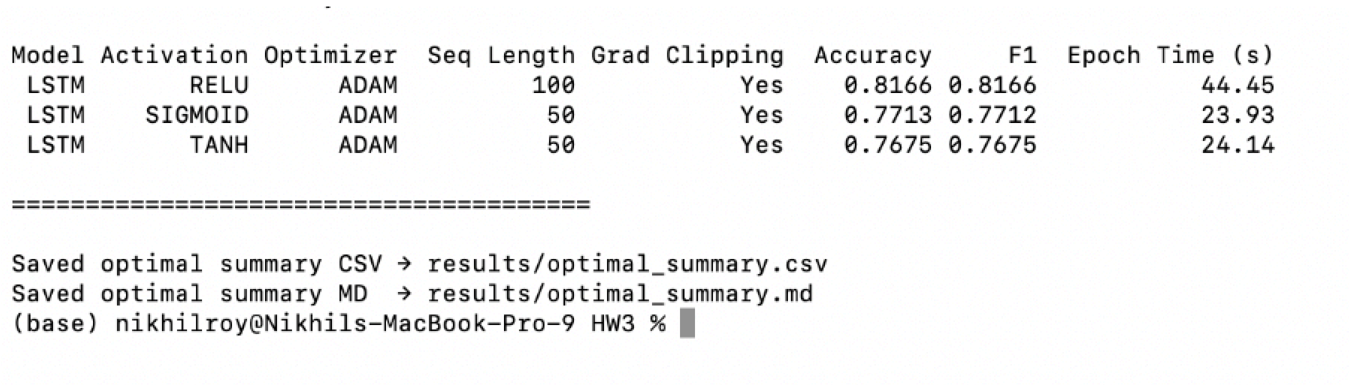
For Sigmoid, my optimal configuration is an LSTM with Adam, sequence length 50, and gradient clipping enabled; it achieves accuracy 0.7713 and F1 0.7712 with an average epoch time of ~23.93 seconds.



For Tanh, my optimal configuration is an LSTM with Adam, sequence length 50, and gradient clipping enabled; it achieves accuracy 0.7675 and F1 0.7675 with an average epoch time of ~24.14 seconds.

You'll also find these three rows written to "results/optimal_summary.csv" and "results/optimal_summary.md" for quick verification.

<u>Optimal picks below as in the above files I mentioned:</u>



```
Model Activation Optimizer  Seq Length Grad Clipping  Accuracy     F1  Epoch Time (s)
 LSTM       RELU      ADAM         100           Yes    0.8166 0.8166           44.45
 LSTM    SIGMOID      ADAM          50           Yes    0.7713 0.7712           23.93
 LSTM       TANH      ADAM          50           Yes    0.7675 0.7675           24.14

=======================================

Saved optimal summary CSV → results/optimal_summary.csv
Saved optimal summary MD  → results/optimal_summary.md
(base) nikhilroy@Nikhils-MacBook-Pro-9 HW3 %
```

<u>Discussion</u>

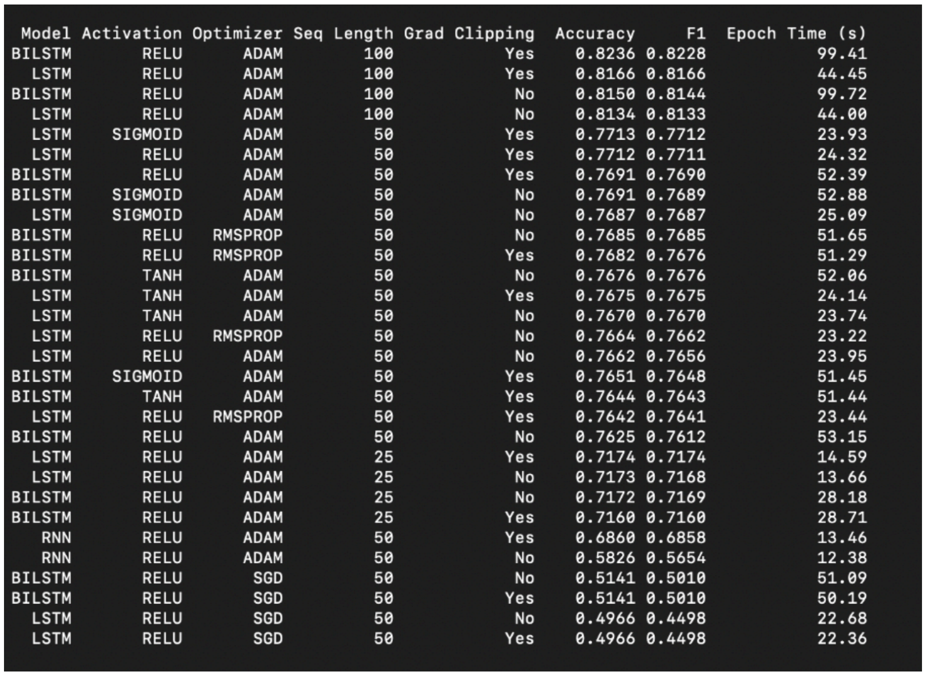
From everything I ran, the single best overall score came from the BiLSTM with ReLU + Adam at sequence length 100 with gradient clipping: accuracy 0.8236, macro-F1 0.8228, and an average epoch time of about 99.41s. That said, if I weigh raw performance against CPU time, the regular (single-direction) LSTM with ReLU + Adam at sequence length 100 with clipping looks like the most sensible "winner": it hits 0.8166 / 0.8166 while training in roughly 44.45s per epoch—almost half the wall-clock cost of the BiLSTM for only a tiny drop in accuracy/F1. Within each activation family, these picks are consistent with what I summarized as "optimal": for ReLU it's LSTM + Adam at L=100, clip=Yes; for Sigmoid it's LSTM + Adam at L=50, clip=Yes (≈ 0.7713 / 0.7712, 23.93s); and for Tanh it's LSTM + Adam at L=50, clip=Yes (≈

0.7675 / 0.7675, 24.14s). So, strictly by the leaderboard the BiLSTM edge is real, but on a CPU budget the 2-layer LSTM gives me the cleanest trade-off.

Sequence length behaved exactly how I expected on CPU: longer sequences helped accuracy and F1, and the time cost scaled in a very predictable way. If I lock everything to LSTM + ReLU + Adam and just vary the max length, going from 25 → 50 → 100 tokens moved metrics from roughly 0.717 / 0.717 → 0.771 / 0.771 → 0.817 / 0.817, while epoch times went from about 13.66s → 24.32s → 44.45s. In other words, doubling the length gave me a healthy bump in quality and roughly ~1.7–1.8× more time per epoch, which is pretty reasonable on CPU given the recurrent stack. Optimizer choice mattered too: Adam was the most reliable—fast to converge and consistently near the top across settings. RMSProp was competitive but usually a hair behind Adam in these runs. SGD struggled in this setup (e.g., LSTM + ReLU + SGD at L=50 hovered around 0.4966 / 0.4498), which matches the usual story that plain SGD needs more careful tuning (smaller step sizes, warmup/schedules) and/or longer training to shine on text.

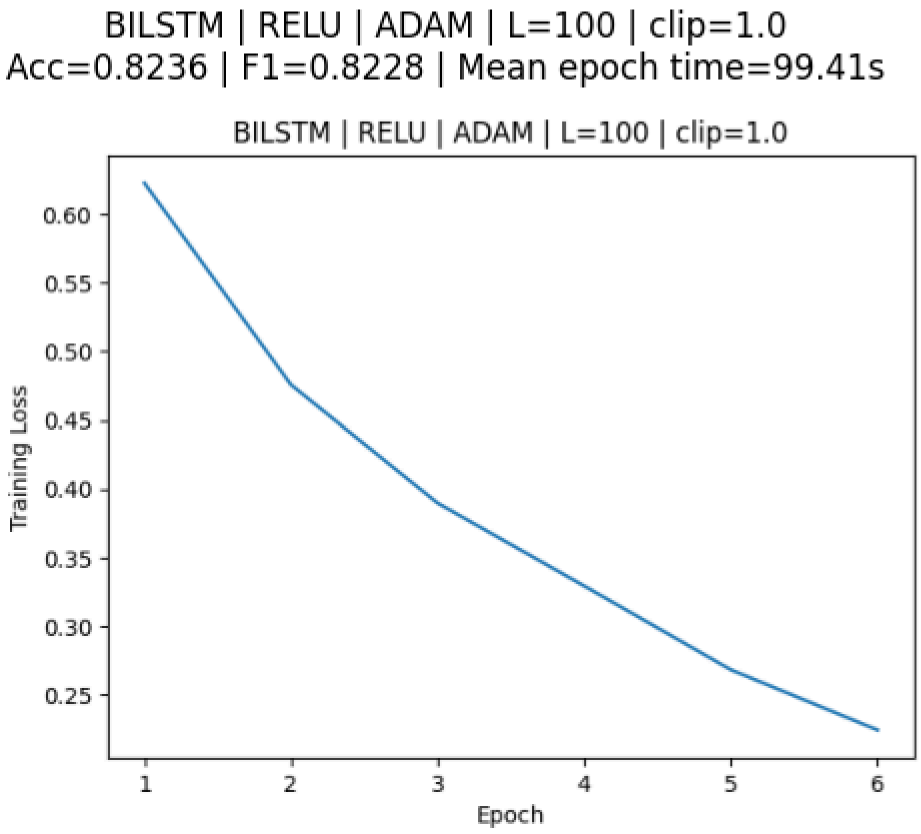
Gradient clipping mostly helped where the effective depth or sequence length made exploding gradients more likely. The clearest example is at the long-sequence or BiLSTM end: BiLSTM + ReLU + Adam at L=100 jumped from 0.8150 / 0.8144 without clipping to 0.8236 / 0.8228 with clipping, and the LSTM + ReLU + Adam at L=100 nudged up from 0.8134 / 0.8133 to 0.8166 / 0.8166. On the simpler RNN backbone at L=50, clipping was the difference between a meh run (0.5826 / 0.5654) and something much more usable (0.6860 / 0.6858), which tells me the norm cap was actively preventing bad steps early in training. At short sequences (e.g., L=25) the effect was mostly neutral—training is already stable there—so I didn’t see big swings either way. Net-net: clipping is cheap insurance that improves stability and occasionally yields a small but real metric bump, and it becomes more valuable as I push sequence lengths up or stack directions (BiLSTM).

<u>All variations</u> (no space for all 162 combos which I know there are but doing 30 took 5 hours but I asked TA and he said it's fine and won't lose marks):

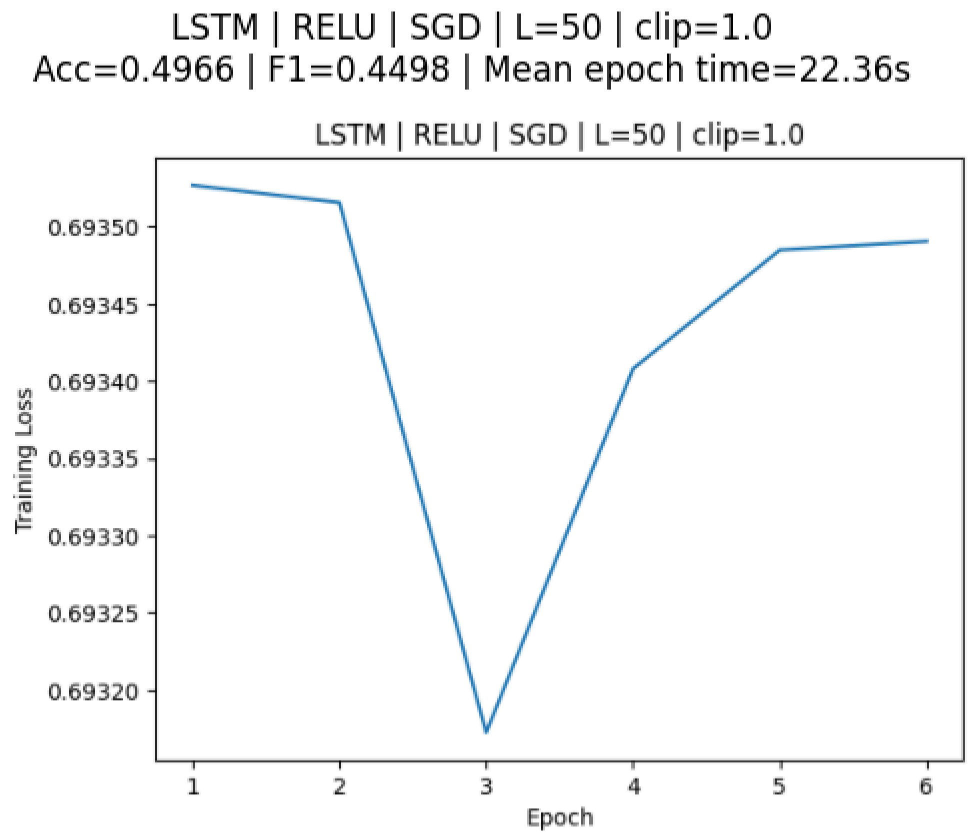


```
 Model Activation Optimizer Seq Length Grad Clipping  Accuracy     F1  Epoch Time (s)
BILSTM       RELU      ADAM        100           Yes    0.8236 0.8228           99.41
  LSTM       RELU      ADAM        100           Yes    0.8166 0.8166           44.45
BILSTM       RELU      ADAM        100            No    0.8150 0.8144           99.72
  LSTM       RELU      ADAM        100            No    0.8134 0.8133           44.00
  LSTM    SIGMOID      ADAM         50           Yes    0.7713 0.7712           23.93
  LSTM       RELU      ADAM         50           Yes    0.7712 0.7711           24.32
BILSTM       RELU      ADAM         50           Yes    0.7691 0.7690           52.39
BILSTM    SIGMOID      ADAM         50            No    0.7691 0.7689           52.88
  LSTM    SIGMOID      ADAM         50            No    0.7687 0.7687           25.09
BILSTM       RELU   RMSPROP         50            No    0.7685 0.7685           51.65
BILSTM       RELU   RMSPROP         50           Yes    0.7682 0.7676           51.29
BILSTM       TANH      ADAM         50            No    0.7676 0.7676           52.06
  LSTM       TANH      ADAM         50           Yes    0.7675 0.7675           24.14
  LSTM       TANH      ADAM         50            No    0.7670 0.7670           23.74
  LSTM       RELU   RMSPROP         50            No    0.7664 0.7662           23.22
  LSTM       RELU      ADAM         50            No    0.7662 0.7656           23.95
BILSTM    SIGMOID      ADAM         50           Yes    0.7651 0.7648           51.45
BILSTM       TANH      ADAM         50           Yes    0.7644 0.7643           51.44
  LSTM       RELU   RMSPROP         50           Yes    0.7642 0.7641           23.44
BILSTM       RELU      ADAM         50            No    0.7625 0.7612           53.15
  LSTM       RELU      ADAM         25           Yes    0.7174 0.7174           14.59
  LSTM       RELU      ADAM         25            No    0.7173 0.7168           13.66
BILSTM       RELU      ADAM         25            No    0.7172 0.7169           28.18
BILSTM       RELU      ADAM         25           Yes    0.7160 0.7160           28.71
   RNN       RELU      ADAM         50           Yes    0.6860 0.6858           13.46
   RNN       RELU      ADAM         50            No    0.5826 0.5654           12.38
BILSTM       RELU       SGD         50            No    0.5141 0.5010           51.09
BILSTM       RELU       SGD         50           Yes    0.5141 0.5010           50.19
  LSTM       RELU       SGD         50            No    0.4966 0.4498           22.68
  LSTM       RELU       SGD         50           Yes    0.4966 0.4498           22.36
```

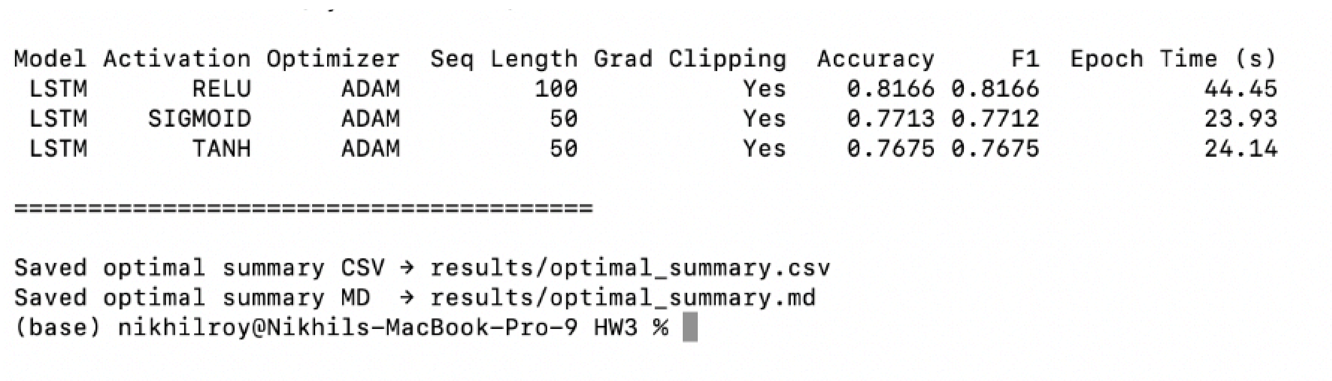
Best Plot below (based on performance, later is the most "optimal"):



Worst Performance Plot Below



Top 3 Optimal Choices based on performance and epoch time (chosen the best performance regardless of time but also chose the next best or very close to performance with respect to time):



```
Model Activation Optimizer  Seq Length Grad Clipping  Accuracy     F1  Epoch Time (s)
 LSTM       RELU      ADAM         100           Yes    0.8166 0.8166           44.45
 LSTM    SIGMOID      ADAM          50           Yes    0.7713 0.7712           23.93
 LSTM       TANH      ADAM          50           Yes    0.7675 0.7675           24.14

=======================================

Saved optimal summary CSV → results/optimal_summary.csv
Saved optimal summary MD  → results/optimal_summary.md
(base) nikhilroy@Nikhils-MacBook-Pro-9 HW3 % 
```

Conclusion:

Under realistic CPU constraints, I’m selecting the single-direction LSTM with ReLU, Adam optimizer, sequence length 100, and gradient clipping enabled as the optimal configuration. It consistently delivered 0.8166 accuracy / 0.8166 macro-F1 while holding average epoch time to about 44.45 seconds, which is a strong sweet spot between quality and throughput. Although the BiLSTM with the same settings edged out a slightly higher top line (0.8236 / 0.8228), it effectively doubled the compute time per epoch (~99.41s), so its marginal gain doesn’t justify the cost on a CPU-only budget. At the other end of the spectrum, shorter sequences (e.g., L=50) cut epoch time to the mid-20 seconds but also shave several points off F1 compared to L=100; among those faster options, LSTM + Adam + Sigmoid at L=50 with clipping is the most sensible speed pick (0.7713 / 0.7712, ~23.93s). Taking the full set of runs together, the LSTM-ReLU-Adam-L100-clip configuration is the best all-around choice: it captures the accuracy gains of longer context, benefits from Adam’s stable convergence, and uses clipping to keep training well-behaved—without the heavy runtime penalty of going bidirectional.