Recurrent Neural Networks
Homework #2

https://github.com/kiwiiiiiiiiO/RNN-Transformer/tree/main/HW2

1. Introduction

This assignment aims to build binary classifiers using LSTM and GRU architectures to detect whether a tweet is about a real disaster (target: 1) or not (target: 0). In this homework, I accomplished the following tasks:

- Built two models: one using **LSTM** (with starter code), the other using a **self-implemented GRU**
- Preprocessed tweets with customized text normalization, including abbreviation expansion and emoji replacement
- Trained and evaluated both models using PyTorch and visualized the results
- Compared model performance in terms of **accuracy, precision, recall**, training dynamics, and leaderboard scores
- Applied both models to an external test.csv file and generated submission files (lstm_submission.csv, gru_submission.csv) in the required format

2. Models and Implementation

2-0. Data Preprocessing

To improve the robustness of model training on noisy tweet data, I applied a preprocessing pipeline following the sample code:

1. Remove punctuation and convert all characters to lowercase
2. Replace entities with tokens:
   - URLs -> URL, HTML tags -> ' ',
   - Mentions -> USER, numbers -> NUMBER,
   - Emojis -> EMOJI, emoticons -> SMILE or SADFACE, <3 -> HEART
3. Expand abbreviations using a custom dictionary (e.g., "lol" "laugh out loud")
4. Remove non-ASCII characters
5. Remove stopwords using NLTK's English stopword list

After cleaning, I used Keras's Tokenizer (limited to the top 3,000 most frequent words) to tokenize the cleaned text. The sequences were then padded or truncated to a fixed length of 20 tokens.

2-1. LSTM Model

- Architecture:
  - nn.Embedding -> nn.LSTM -> mean pooling -> Linear output
- Embedding Size: 32
- Hidden Size: 32
- Dropout: 0.2 on the embedding layer, 0.4 in LSTM
- Activation: Sigmoid (via BCEWithLogitsLoss)

2-2. GRU Model

- Architecture: Similar to LSTM model, but using nn.GRU
- Layers: 2-layer GRU
- Embedding Size: 32
- Hidden Size: 32
- Dropout: 0.2 on the embedding layer, 0.4 in GRU
- Output via mean pooling followed by a linear layer

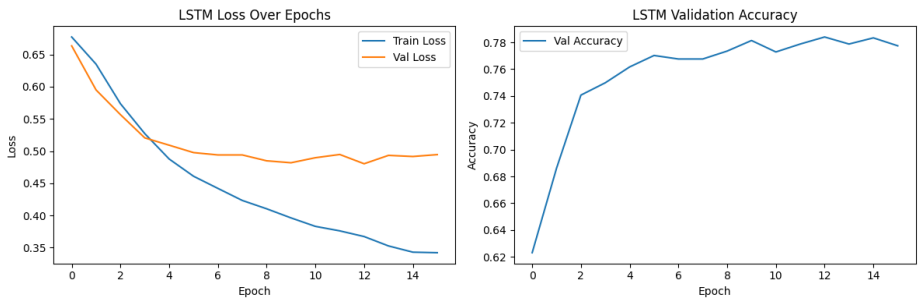
Both models used BCEWithLogitsLoss and were optimized with Adam.

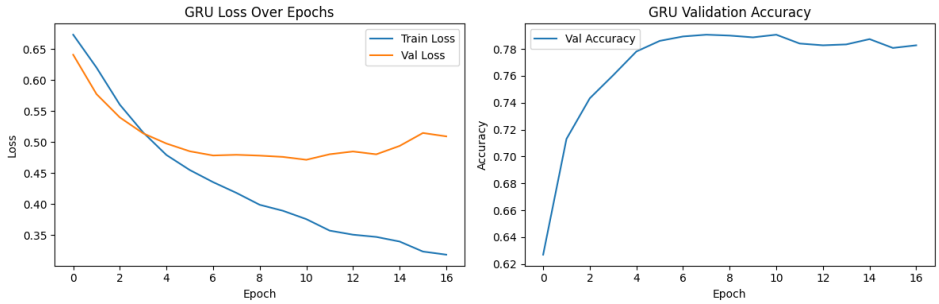
3. Training and Evaluation Strategy

Component	Setting
Batch Size	32
Optimizer	Adam
Loss Function	Binary Cross Entropyð
Epochs	20 with early stopping
Validation	20% of training set
Evaluation	Accuracy, Recall, Precision
Device	GPU (auto-detect)

I tracked training and validation loss, and plotted learning curves for each model.

4. Model Performance Comparison





The plots show both models exhibit steady convergence during training:

- LSTM:
  - Training loss decreases consistently, while validation loss flattens after epoch 8.
  - Validation accuracy peaks at 78.40%, showing a stable performance after early epochs.
- GRU:
  - Faster drop in loss and earlier plateau in accuracy.
  - Achieved a higher peak validation accuracy of 79.05%, with a smooth convergence around epoch 10.

This suggests that the GRU model converges slightly faster and more effectively captures the signal in the data with fewer fluctuations.

Model	Validation Accuracy	Validation Recall	Validation Precision
LSTM	0.7774	0.6857	0.7672
GRU	**0.7827**	0.6533	0.8

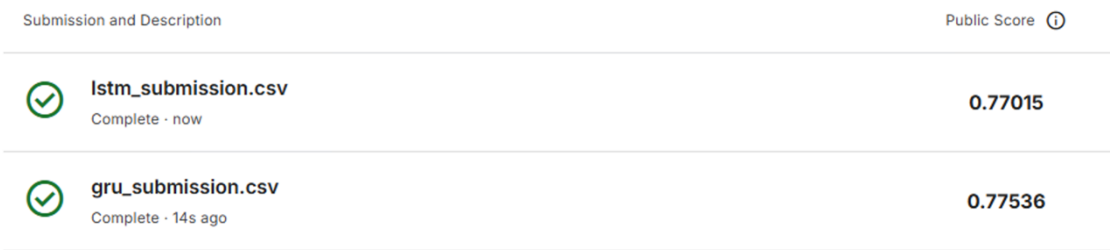
- GRU achieved higher overall accuracy and precision, which indicates it makes more confident positive predictions.
- LSTM had better recall, meaning it captured slightly more true disaster cases, albeit with a small drop in precision.
- Overall, both models performed comparably, but GRU exhibited slightly better generalization based on these metrics.

Evaluation on E3 Test Set (All target = 0)

```
LSTM Accuracy: 0.6350
GRU Accuracy: 0.6693
```

- These results demonstrate each model's ability to minimize false positives when applied to real-world, unseen data with a strongly imbalanced class distribution.

Upload to Kaggle



- These public scores on Kaggle indicate that the GRU model consistently generalized better to the test data.

5. Conclusion

In this assignment, I successfully implemented and compared two RNN-based models for disaster tweet classification: LSTM and a GRU model.

- Both models achieved strong validation accuracy (LSTM: 78.40%, GRU: 79.05%).
- GRU consistently outperformed LSTM in external testing scenarios, especially on the Kaggle leaderboard and the E3 test set with strong class imbalance.
- The results demonstrate that GRU not only converges faster but also generalizes better in practice.

Through this process, I gained hands-on experience with PyTorch, data preprocessing for NLP, model tuning, and evaluation using real-world data. Overall, this assignment enhanced my ability to design and evaluate deep learning models for NLP applications, especially in noisy and imbalanced real-world scenarios.