Neural Networks for Sentence Classification

Deepti Suresh, Aparna Ganesh, Ryan Maxey

About the Team

- Deepti
  - MEng Student
  - Concentration in Data Analytics and AI
  - Graduating in May
- Aparna
  - MEng Student
  - Concentration in Software Engineering
  - Graduating in May
- Ryan
  - MEng student
  - Concentration in Software Development & Applications
  - Graduating in May

Problem Description

- Text classification can be used for a variety of tasks such as sentiment analysis, topic detection, intent identification
- Real life applications include:
  - Detecting Hate Speech
  - AI-driven chatbots (eg. for assisting with mental health)
  - Writing Assistants (eg. Grammarly)
- Two popular models for text classification: RNN and CNN
  - How do they perform for the task of sentiment analysis?

Approaches

- Our goal is to compare a simple RNN, advanced RNN, and CNN model to investigate which neural network performs better through analysis of various performance metrics:
    - Accuracy
    - Precision
    - Recall
    - F1 score

- We will evaluate the models by performing various exploratory tasks.

Approaches (Simple RNN)

- For the recurrent neural network (RNN), a sequence of words will be used as input. The RNN will produce a hidden state for each word, processing them sequentially.
- To calculate the hidden state for a word, the RNN will take the current word, xt, the hidden state produced for the previous word, ht-1, and use the equation below:
    - ht=RNN(xt, ht-1)
- Once the final hidden state is produced, it is passed to a linear layer which gives the predicted sentiment of the input sentence.

Approaches (Advanced RNN)

For the advanced RNN model, we enhance the model to achieve better performance by using the following:

- packed padded sequences
- pre-trained word embeddings *("glove.6B.100d")*
- different RNN architecture *(Long Short-Term Memory (LSTM))*
- bidirectional RNN
- multi-layer RNN *(also called deep RNNs)*
- Regularization *(Dropout)*
- a different optimizer *(Adam)*

Approaches (CNN)

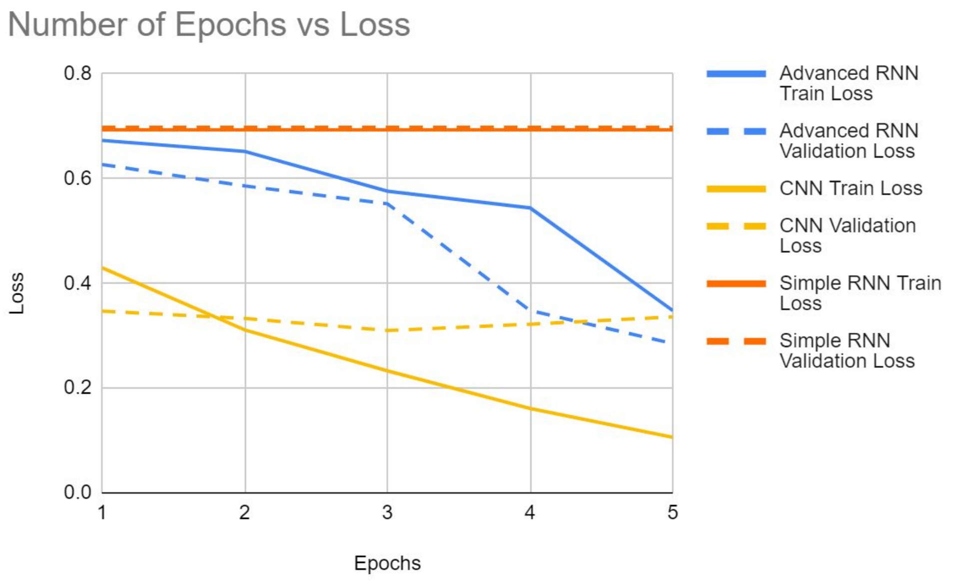
- With Convolutional Neural Networks (CNN), each sentence is used as a sequence of word embeddings as input to the model.
- The input layer takes each word in a sentence and converts it into a higher-dimensional vector.
- This is fed into the convolutional layers which will apply a set of kernels to create a feature map which will indicate local patterns and relationships between words.
- Lastly, the fully connected layer will flatten the produced feature maps into fully connected layers which utilize weights to predict the final output.
- **The output will be a probability distribution of the likelihood of a sentence belonging to the different sentiment classes (positive or negative).**

Results

Model	Accuracy	Precision	Recall	F1-Score
Simple RNN *(Base Model)*	44.50%	0.4762	0.6854	0.5223
Advanced RNN	84.39%	0.7537	0.9260	0.8310
CNN	85.52%	0.8021	0.8585	0.8211

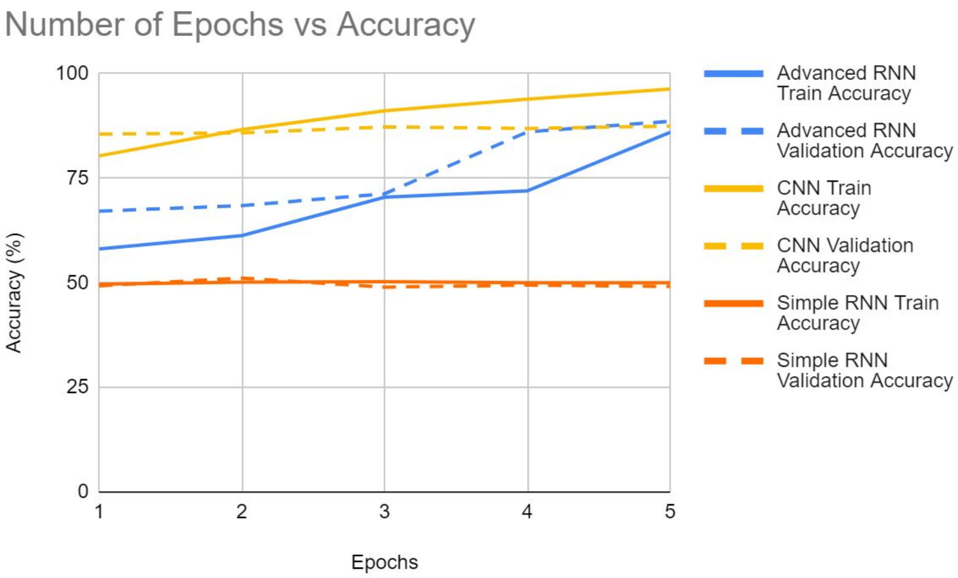
Training & Validation Loss

- Little change in training and validation loss for Simple RNN.
- Advanced RNN model training and validation loss decreased overtime indicating that the model is learning and generalizing well to the validation data.
- Significant decrease in train loss for CNN (not the case for validation loss)
    - This may be a case of overfitting.



Training & Validation Accuracy

- Training and validation accuracy increases for CNN and advanced RNN.
- CNN reaches 96.35% train accuracy by the 5th epoch.



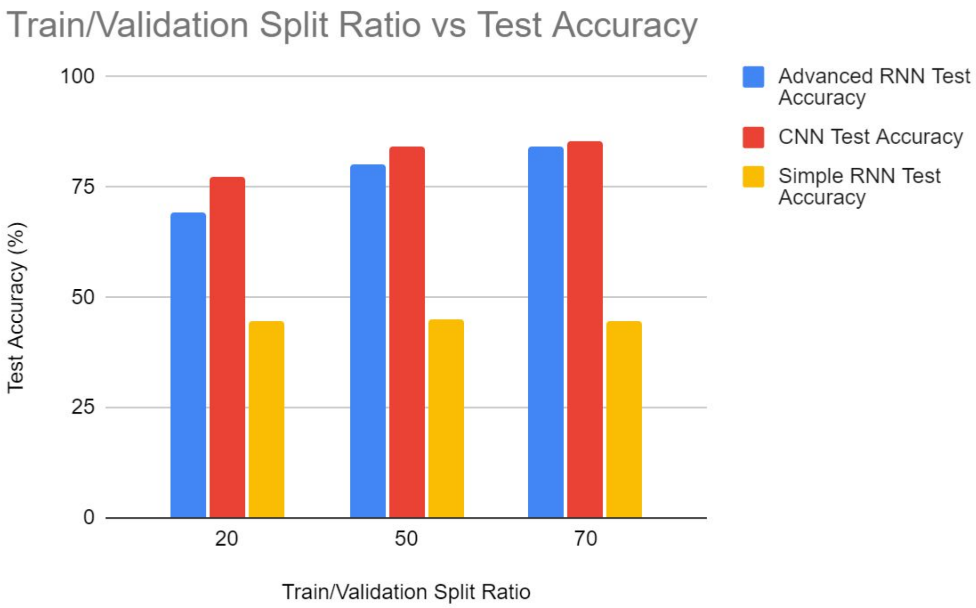
Exploratory Tasks

Exploratory tasks help us evaluate the performance of our models. By experimenting with different hyperparameters, we can identify optimal settings for our model:

1. Train/Val Split: Modify the data split (20, 50, 70) to determine the optimal ratio for training and validation sets.
2. Hyperparameter Tuning:
    a. Batch Sizes: Experiment with 32, 64, and 128 to find the optimal tradeoff between accuracy and training time.
    b. Optimizer: Test SGD and Adam to see which performs better.
    c. Dropout: Experiment with different rates (0.5, 0.6, 0.8) to evaluate the impact on accuracy and generalization.

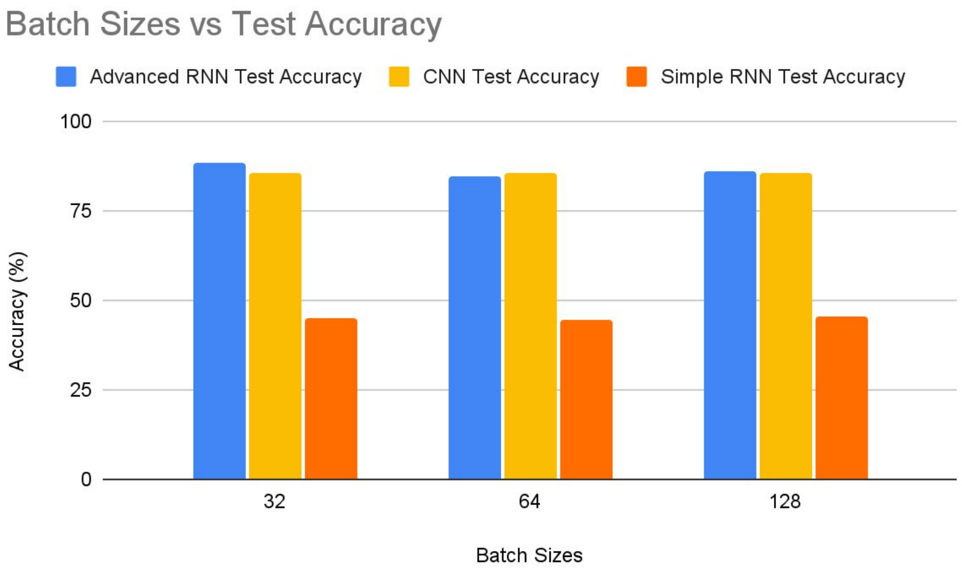
1. Train/Validation Dataset Split Ratio

- Split ratio had little effect on the simple RNN
- Considerable jump in accuracy from 20% to 50% split for both Advanced RNN and CNN



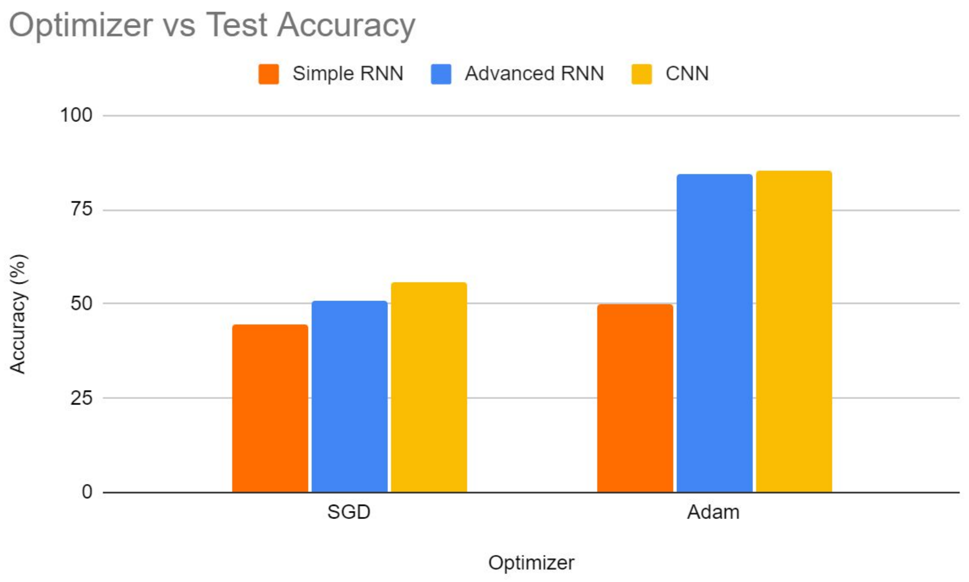
2a. Hyperparameter Tuning: Batch Sizes

- Our Advanced RNN and CNN model outperforms the Simple RNN model in all cases.
- The most optimal batch size was 32 for the Advanced RNN model, whereas the CNN model produced similar results across all three batch sizes.



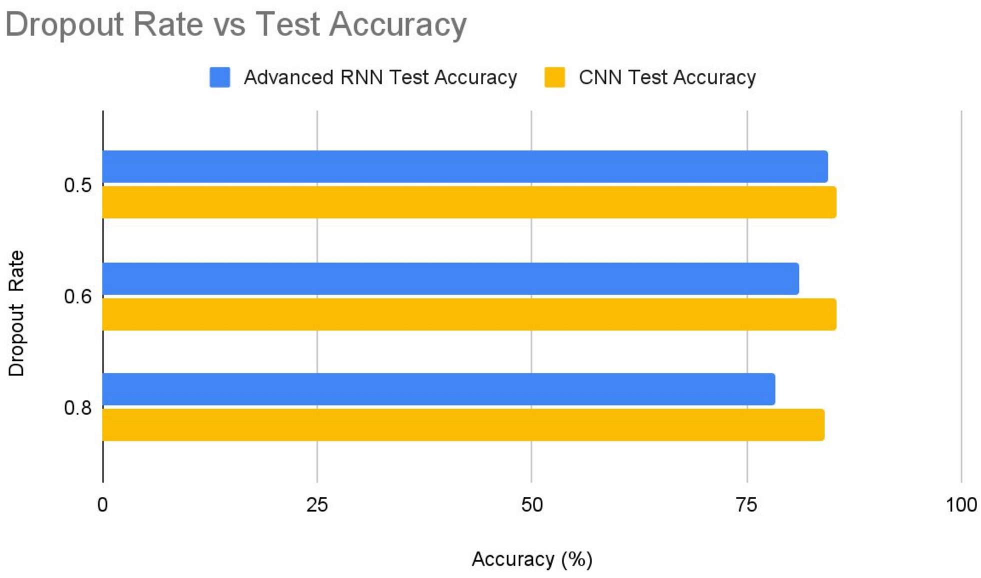
2b. Hyperparameter Tuning: Optimizer (SGD vs Adam)

- Both Advanced RNN and CNN experienced a significant increase in test accuracy with the Adam optimizer
- Advanced RNN increased from 50.95% to 84.39%
- CNN increased from 55.64% to 85.52%



2d. Hyperparameter Tuning: Dropout Rate

- Dropout was only applicable to our Advanced RNN and CNN model.
- Test Accuracy decreased with increase in dropout value indicating that our models were not able to fit properly.
- This behavior might be an indication of higher dropout rate resulting in a higher variance to some of the layers, which also degraded training.



Lessons Learned

- Ensure that we have sufficient computing resources for our models
- Make sure the runtime type for the Colab notebook is set to GPU
  - Some of our initial trainings were using the CPU which resulted in slow computation
- RNN with slight modifications can be an effective model for sentiment analysis
- CNNs are well suited for sentiment analysis

Future Work

- Support more complex classification of movie reviews
  - Rather than the binary classification of “positive” and “negative” there could be additional labels for reviews with sentiment in between
- Explore and tune other hyperparameters
  - Embedding dimension
  - Hidden dimension
- Test the models on different datasets
  - Amazon Product Data, Stanford Sentiment Treebank

References

- Trevett, B (2021) Simple Sentiment Analysis https://github.com/bentrevett/pytorch-sentiment-analysis/blob/master/1%20-%20Simple%20Sentiment%20Analysis.ipynb
- "Introduction to Convolutional Neural Networks CNNs," aigents.co. https://aigents.co/data-science-blog/publication/introduction-to-convolutional-neural-networks-cnns
- Shreya Ghelani, "Text Classification — RNN's or CNN's?," Medium, Jun. 02, 2019. https://towardsdatascience.com/text-classification-rnns-or-cnn-s-98c86a0dd361
- "What are recurrent neural networks?," IBM. [Online]. Available: https://www.ibm.com/topics/recurent-neural-networks. [Accessed: 26-Mar-2023].
- Maas, Andrew L, et al. "Learning Word Vectors for Sentiment Analysis." The 49th Annual Meeting of the Association for Computational Linguistics Human Language Technologies: Held at the Portland Marriott Downtown Waterfront in Portland, Oregon, USA, June 19-24, 2011, ACL?, 2011.
- Y. Kim, "Convolutional Neural Networks for Sentence Classification," *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, doi: https://doi.org/10.3115/v1/d14-1181.
- MonkeyLearn Inc. "Text Classification: What It Is and Why It Matters." MonkeyLearn, monkeylearn.com/text-classification/.

Thank You! Any Questions?