Comparing CNN's, RNN's, and Transformers on Sentiment Analysis of Movie Reviews

Jayden Ong **Amey Dhamgunde**

Abstract

The transformer architecture is a relatively new and powerful ML tool that uses attention mechanisms to process and 'learn from' extremely large sequences of data, which has made it increasingly popular for NLP tasks such as sentiment analysis, language translation, and text classification. Transformers are replacing the previously dominant CNN/RNN architectures which excelled at images or NLP classification style tasks. This paper explores and compares the performance of these architectures by training 4 models: CNN, RNN, CNN+RNN, and transformer in a closed environment to evaluate their differences, and whether CNN/RNN models are still viable. It is found that CNN and RNN based models achieve testing accuracies of 89%, 3% higher than the transformer at 86%, with less than half the parameters and a fraction of the training time. [1]

Introduction

Attention-based transformers are a relatively new, state of the art machine learning architecture which are able to use self-attention mechanisms to put more focus on different parts of the input as well as recognize extremely long distance relationships in input sequences. As a result, they are being adopted for many NLP tasks such as sentiment analysis, language translation, and text classification.

While attention-based transformers are becoming increasingly popular in natural language processing tasks, both CNNs and RNNs could still outperform transformers in sentiment analysis. One potential advantage of CNNs is the ability to capture local patterns/features in the input data by applying a series of convolutional filters to the input data, allowing it to extract features at different levels of abstraction. This makes it particularly effective for tasks where local features are important, such as text classification based on individual words or phrases. RNNs are well-suited to NLP tasks for the same reason as transformers - they maintain an internal state that can be updated with each new input, allowing them to incorporate information from previous inputs into the current prediction. This makes them effective for sequential tasks such as sentiment analysis or language translation, where the meaning or classification of a word or phrase can depend on its context.

In contrast, multi-headed attention-based transformers operate on the entire sequence of input data simultaneously, allowing them to attend to all parts of the input at once. While this is hugely beneficial for tasks where global context is important, and on very large datasets, it may not be as effective for tasks where local dependencies and temporal information are crucial, with smaller datasets.

This paper aims to test the performance of four models: CNN, RNN, CNN + RNN, and Transformer which will be trained and analyzed with the aim of determining whether or not transformer models outperform convolutional and recurrent neural networks on a sentiment analysis classification task.

[1] `https://github.com/ameydhamgunde/nn_sentiment_analysis`

Preprint. Under review.

Related Works

The inspiration behind sentiment analysis was classifying the tone of product reviews as either positive, negative, or neutral. In the research paper “Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews” published by Kushal Dave, Steve Lawrence, and David M. Pennock, they introduced the concept of applying machine learning algorithms to this task. To classify reviews, their model used n-grams to identify features (phrases) that were most commonly associated with positive or negative reviews. The frequency of such phrases was used to calculate a score and determine a class.

Recurrent neural networks were not widely applicable to sentiment analysis until 2014 when a paper was published called,"Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank". The authors introduced the concept of Recursive Neural Tensor Networks which was trained on a Sentiment Treebank. The Sentiment Treebank was the first corpus that contained fully labeled parse trees that allowed models to understand the effect of composition on the sentiment of a phrase. They managed to surpass the current benchmark of 80.7% by 5%.

Transformers were not widely used for sentiment analysis until Google released BERT in 2018. BERT (Bidirectional Encoder Representations from Transformers) is a transformer-encoder model pretrained on large amounts of text. This allows it to better understand the intricate relationships between words which in theory, makes it perfectly applicable to sentiment analysis. As of now, transformers are the most popular choice for NLP tasks due to their ability to capture extremely long-term dependencies which is something that RNN’s cannot do.

Methodology

To evaluate each model (CNN, RNN, CNN+RNN, and Transformer) on a semantic classification task, we construct an architecture for each model using similar pre/post-processing techniques to most effectively isolate each architecture’s performance, and conduct a hyperparameter sweep using grid searching. Using the IMDb dataset consisting of movie reviews accompanied by a ’positive’ or ’negative’ label, the input strings are preprocessed into integer sequences using a tokenizer with a vocabulary size of 2000. Each input sequence is then padded to a maximum length of 200. Finally, the dataset is split into an 80/10/10 train/validation/test split, using a batch size of 250.

To isolate each technology’s performance as effectively as possible, each model initially takes input as padded, tokenized sequences, passing it through an embedding layer, mapping each token to a 60-dimensional vector. After applying the unique model architectures to the embedded input, the intermediate output is ’post-processed’ - it is globally max-pooled to find the strengths of the various identified sentiments across the timestamps, after which a fully connected layer is applied with a softmax used to get the probability of the input belonging in either the positive or negative class. During training, cross-entropy loss was used in place of the softmax with the Adam optimizer and weight decay of 5e-4. Dropout layers are used to promote robust training and generalization. Training curves can be found in the appendix, and all models can be found on github.

CNN

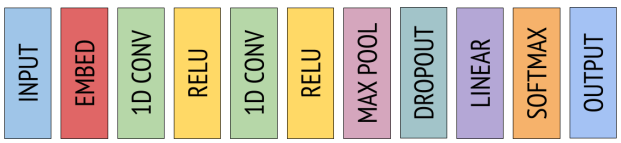

Figure 1: CNN Architecture

As seen in figure 1, the embedded input is transposed to allow for 1D convolution to occur along the sequence length, with 60 input channels and double the hidden size output channels. After the first convolution, the nonlinear activation function ReLU is applied. For the second convolution layer, the process is repeated where the number of input channels double the hidden size, and the number of output channels the hidden size. Finally, the convolved input is post-procssed.

Kernel Size	Learning Rate	Hidden Units	Conv Layers
[2, 3, 4]	[1e-3, 5e-4, 2e-4]	[32, 64, 128]	[1, 2, 3]

The most important hyperparameters to consider for this model are the kernel size # layers, learning rate, and number of hidden units. Larger kernel sizes and # layers will increase the size of the receptive field for hidden unit, but a kernel size too large hinders a CNN's ability to recognize patterns. A small learning rate causes the model to converge very slowly and a learning rate too large could cause the model to diverge. Finally, too many hidden units could cause the model to overfit and too few hidden units could cause the model to underfit. Another important hyperparameter that was considered was the number of epochs - the more epochs, the higher the risk of overfitting. To combat this, we used checkpoints to retrieve the model state before overfitting due to overtraining occurs.

Using checkpoints to save the model state with the lowest validation loss across all hyperparameter combinations and epochs, a testing accuracy of 89.08% was obtained by using a learning rate of 1e-3, kernel size of 4, and 128 hidden units. [2]

RNN

Figure 2: RNN Architecture

Figure 2 shows that the embedded input is processed using a Gated Recurrent Unit (GRU) layer with a hidden number of GRU cells, and then post-processed as described previously. The max-pooling layer is applied to the output of GRU along the sequence length to aid in semantic classification - we are looking for the "strongest" sentiments identified in the sequence by GRU.

Cell	Learning Rate	Hidden Units	RNN Layers
[Elman, GRU, LSTM]	[1e-3, 5e-4, 2e-4]	[32, 64, 128]	[1, 2, 3]

The hyperparameter sweep for this model consisted of varying the parameters above. The cell type determines the complexity of training as well as the model's affinity of retaining long-term information. and the combination of # hidden units and # layers controls the expressive capacity of the model; too much expressive capacity results in poor generalization.

A test accuracy of 88.62% was achieved using checkpoints, with a learning rate of 5e-4 and 64 hidden units in 1 layer of GRU cells. [3]

CNN + RNN

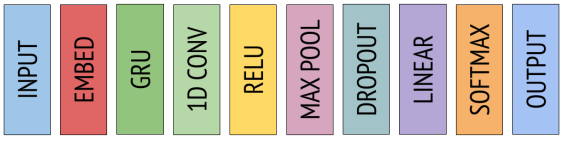

Figure 3: CNN+RNN Architecture

As seen in figure 3, this model uses a combination of the two previously discussed architectures: the embedded input is first processed in a GRU layer with a hidden number of GRU cells (60 $\rightarrow$ hidden size). Next, the intermediate output is transposed for 1D convolution along the sequence length (hidden size $\rightarrow$ hidden size), after which it is post-processed.

[2] Figure 5: Training Loss, Validation Loss, Training Accuracy, and Validation Accuracy for CNN
[3] Figure 6: Training Loss, Validation Loss, Training Accuracy, and Validation Accuracy for RNN

Learning Rate	Hidden Units	Conv Layers	GRU Layers
[1e-3, 5e-4, 2e-4]	[32, 64, 128]	[1, 2, 3]	[1, 2, 3]

The above hyperparameters were grid searched to find the best performing model. The combinations of hidden units/conv layers/GRU layers controls the model capacity - again, too low means the model can't classify and too high means the model generalizes poorly.

The optimal hyperparameter settings produced a test accuracy of 89.62% was achieved using a learning rate of 5e-4 and 64 hidden units in 1 layer of GRU cells, and 1 convolution layer. [4]

Transformers

Figure 4: Transformer Architecture

The final model uses the state-of-the-art transformer encoder architecture with self-attention, feed-forward, and normalization layers, which is represented by the 'encoder' block in figure 7. This transformer model is heavily inspired by Google's BERT, but smaller (6 layers instead of 12, smaller feedforward network, smaller input sequence size) due to compute resource limitations. As seen in figure 4, the embedded input is processed by the 6 encoding layers (represented as a TransformerEncoder in PyTorch) with a feedforward dimension of a hidden size and dropout enabled. Similar to BERT, the output of the encoder layers is pooled, then passed into a fully connected layer with ReLU activation, at which point it is post-processed without dropout or weight decay (since the encoders have normalization layers) to get the output as class probabilities.

Learning Rate	Feedforward Dim.	# Layers	# Heads	Activation Fcn.
[5e-4, 2e-4, 1e-4]	[64, 128, 256]	[1, 3, 5, 6]	[2, 4, 5, 6]	[ReLU, GeLU, Tanh]

A grid search was used to find the optimal hyperparameter combination from above. The feedforward dimension, # layers and # heads again represent the model capacity, but unlike the previous architectures, transformers generalize well even with large expressive capacity (provided an adequate dataset size). Here, the limiting factors were the limited compute resources, relatively small dataset size, and time limitations for training on Colab.

The best performing model achieved a test accuracy of 86.38% with a learning rate of 1e-4, a feedforward dimension of 256, 6 layers, 6 attention heads, and ReLU activation. [5]

Conclusion

In this paper, multiple NN architectures were put to the test in an NLP semantic classification task on the IMDb dataset with binary outputs (however, any of the models could be easily generalized to n classes). It was found that the CNN, RNN, and CNN+RNN models outperformed the transformer model in terms of accuracy on the test set, while sporting 60% fewer parameters and much faster training times, showing that there are tasks where CNN's and RNN's outperform the SotA transformer.

Model	CNN	RNN	CNN+RNN	Transformer
Test Acc	89.08%	88.62%	89.62%	86.38%
# Params	167,938	144,322	165,186	399,278
Train Time	76.7684s	61.0522s	51.4050s	1952.5625s

Some next steps include using a frozen pretrained transformer model, such as BERT with different decoder architectures (nonlinear fully connected, convolutional, etc) to analyze its performance on the dataset in comparison to the 4 models explored in this paper.

[4] Figure 7: Training Loss, Validation Loss, Training Accuracy, and Validation Accuracy for CNN+RNN

[5] Figure 8: Training Loss, Validation Loss, Training Accuracy, and Validation Accuracy for transformer

References

[1] Dave, Kushal et al. “Mining the peanut gallery: opinion extraction and semantic classification of product reviews.” The Web Conference (2003).

[2] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. In Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002), pages 79–86. Association for Computational Linguistics.

[3] Bo Pang; Lillian Lee, Opinion Mining and Sentiment Analysis , now, 2008.

[4] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Appendix

Figure 5: Training Loss, Validation Loss, Training Accuracy, and Validation Accuracy for CNN

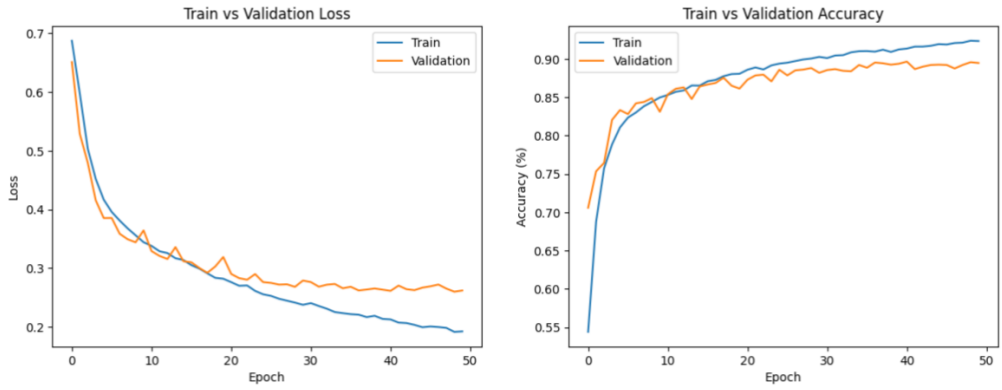

Figure 6: Training Loss, Validation Loss, Training Accuracy, and Validation Accuracy for RNN

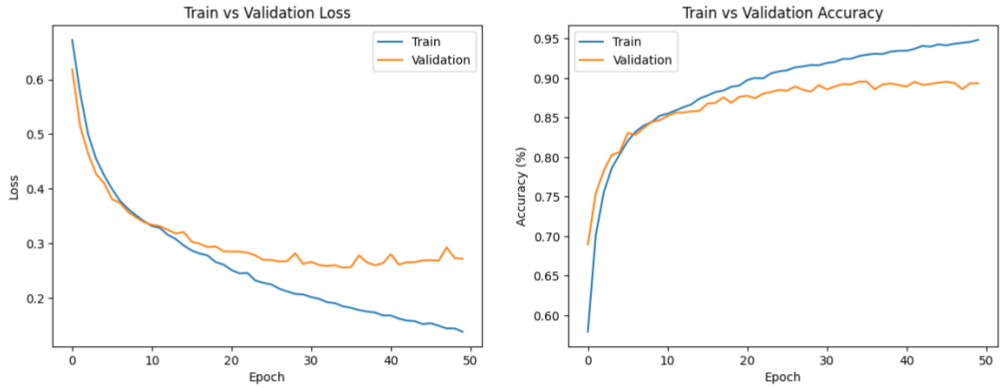

Figure 7: Training Loss, Validation Loss, Training Accuracy, and Validation Accuracy for CNN+RNN

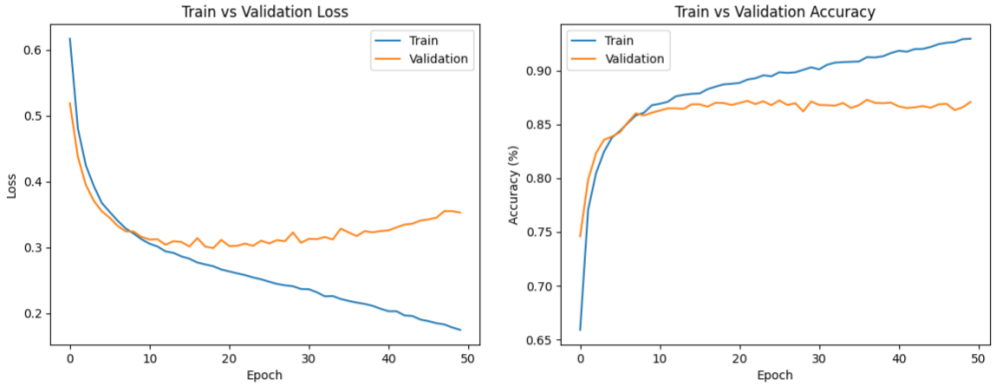

Figure 8: Training Loss, Validation Loss, Training Accuracy, and Validation Accuracy for Transformer