Merging Recurrence and Inception-Like Convolution for Sentiment Analysis

Alex R. Kuefler
Department of Symbolic Systems
Stanford University
akuefler@stanford.edu

Abstract

Computer vision has driven many of the greatest advances in convolutional neural networks, a model family that has found only limited use for natural language processing. The inception module [2] of GoogleNet in particular attains high classification accuracy with few parameters. This project attempts to harness the insights of the inception module in a jointly convolutional and recurrent natural language model. A set of 18 networks built on the idea of combining convolution and recurrence are evaluated on a fine-grained (8 class) sentiment analysis task using the IMDB Large Movie Dataset [5].

1 Introduction

Within the deep learning model family, recurrent (RNN) and convolutional neural networks (CNN) have served as the two major workhorses for natural language processing (NLP) and computer vision tasks respectively. RNNs make use of feedback loops that propagate hidden layer activations through time, allowing these models to represent sequential data like text, where an input's semantics depend heavily on previous (and subsequent) inputs. Conversely, CNNs exploit the spatial structure of images by learning a set of filters that respond selectively to oriented gradients or colors within their receptive field.

Major breakthroughs in computer vision and the ImageNet challenge [1, 2, 3] have driven the development of many variations on the CNN architecture. Google introduces the "inception module" in particular, which achieved ImageNet state-of-the-art in 2014 with remarkably few parameters. CNNs are now showing some promise as sentence models [6, 7]. But despite the growing diversity of model architectures in computer vision, the limited applications of CNNs to NLP still largely resemble the classical architecture formulated by LeCun, Bottou, & Bengio [4].

This project explores ways in which the spatial-selectivity of CNNs may be integrated with the sequential processing of RNNs. A convolutional architecture loosely inspired by GoogleNet's inception module is introduced. Its outputs are merged with those of a gated RNN, allowing the model to respond selectively to important *n*-grams at different scales, while also capturing long-term dependencies in sentences. The model is evaluated on a fine-grained (8 class) sentiment analysis task using the IMDB movie review dataset [5].

1.2 Background

An early CNN model for sentence data is introduced in [6]. Their Time-Delay Neural Network (TDNN) uses a convolution operator to map input to output sequences. These output sequences undergo max-pooling, allowing the model to "flatten" sentences of variable lengths into a vector of maximally responding units. Because the dimensionality of this vector is constrained by model hyperparameters, the TDNN is readily stacked with fully connected layers. However, [7] observes that max pooling causes the model to forget word order, and is unable to distinguish between cases where the max feature appeared once, versus multiple times. Instead, they introduce k-max pooling. Given a CNN output sequence and integer k, the algorithm chooses the k maximally activated entries of the sequence, and returns a subsequence that preserves their order (but discards intermediate features). The "dynamic" variant of this operation treats k as a function of sentence length and network depth. The resulting Dynamic Convolutional Neural Network (DCNN) preserves many of the favorable properties of TDNN, while discarding less information. But despite DCNNs ability to preserve word order, there remain many nuances of sentence data they seem unprepared to handle. The semantics of many sentences depend not only on the order of key words, but also upon how keywords are scoped by modifiers or depend upon a long-term context.

Many RNNs appear better equipped to handle such context effects. Although the classic RNN formulation suffers from the vanishing gradient problem, limiting its memory to only a few time-steps, new architectures experiment with gating mechanisms capable of remembering information across longer sequences. The long short-term memory (LSTM) consists of a constant error carrousel that maintains a cell state, which selectively accepts, erases, or emits its stored value through elementwise multiplication by an input gate i_t, forget gate f_t, and output gate o_t at each timestep [8]. The end product is a hidden state vector h_t that potentially captures information from much earlier timesteps.

$$i_t = \sigma(W^i x_t + U^i h_{t-1}) \; ; \; f_t = \sigma(W^f x_t + U^f h_{t-1}) \; ; \; o_t = \sigma(W^f x_t + U^f h_{t-1})$$

$$\tilde{c}_t = tanh(W^c x_t + U^c h_{t-1}))$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

$$\boldsymbol{h_t = o_t \odot tanh\,(c_t)}$$

Figure 1. LSTM update equations [9].

The final major source of inspiration for this work was the Inception Module of [2]. Szegedy et al. observe that the performance of deep vision models can reliably be improved by increasing model size, but doing so increases the risk of overfitting and the need for greater computational resources.

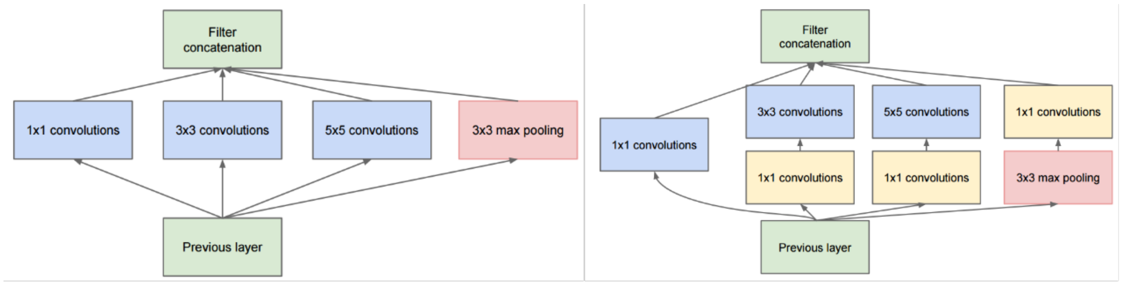

Figure 2. Idealized inception module (left) and pragmatic version (right). From [2].

The inception module introduces sparsity into neural networks by replacing fully-connected (FC) layers (consisting of many parameters) with parallel convolutions that act on the same input layer but at different scales (which save on parameters by sharing weights). They describe an ideal architecture (left on **Figure 2**), where convolution with 1x1 filters is intended to capture correlated features clustered in the same region of input images. Simultaneously, 3x3 and 5x5 convolutions respond to image patterns at slightly larger scales. The feature maps produced by all convolutions (in addition to the outputs of a pooling layer), are concatenated to form the output activations. The authors concede that this architecture may still be computationally expensive, and include a dimensionality reduction stage (using 1x1 convolutions) for their final design (right on **Figure 2**).

2 Approach

This project observes that the notion of sparse, parallel processing of inputs may have a nice analogue for sentiment analysis. N-gram models are widely used for NLP, where feature engineering often consists of selecting the length (*n*-value) for contiguous subsequences of sentence data [10]. Choosing a correct value may require domain knowledge or trial and error. By convolving sentences with a kernel that responds to contiguous subsequences of sentence inputs, the TDNN and DCNN can be interpreted as extracting *n*-grams from text.

Similarly, the proposed Language Inception Module (LIM) extracts *n*-grams of multiple scales from a sentence. But whereas the DCNN computes *n*-gram features of different sizes at different network depths, the proposed architecture computes feature maps for different *n*-grams on the same sentence input. As in TDNN, these features maps are max pooled into a standard length, which are concatenated to form the output vector, as in the inception module. The end result is a CNN resembling the idealized inception module, in that it performs automated feature extraction at multiple scales, in parallel.

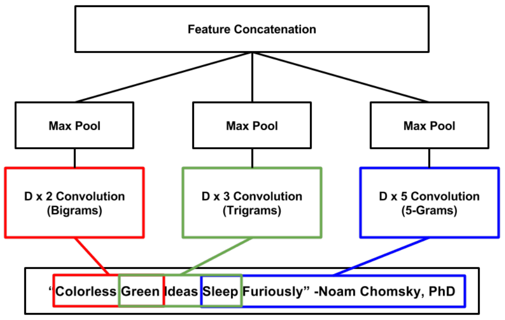
Feature Concatenation
Max Pool
Max Pool
Max Pool
D x 2 Convolution (Bigrams)
D x 3 Convolution (Trigrams)
D x 5 Convolution (5-Grams)
"Colorless Green Ideas Sleep Furiously" -Noam Chomsky, PhD

Figure 3. Language Inception Module.

Unlike DCNN, this first formulation performs simple max pooling, thus discarding word order and frequency information. I instead observe that meaning depending on a sentence's structure and syntax is more adequately modeled by an RNN, and that the job of a convolutional module is to learn *n*-gram features that contribute meaningfully to the minimization of the loss function. As such, the LIM is trained jointly with an RNN, rather than acting as a replacement.

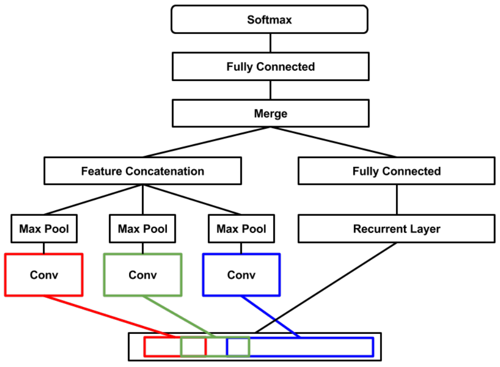

Figure 4. Completed network architecture with CNN (left), RNN (right), and classifier (top).

The resulting architecture features two "trunks" that process the same input sentence in parallel. The convolutional trunk extracts *n*-grams at multiple scales and max-pools to find the strongest responding subset of sentence subsequences (the number of which depends on the number of filters chosen, which is specified as a hyperparameter). The recurrent trunk uses gated units to model long-term dependencies between words in the sentence. The outputs of each trunk are concatenated or undergo elementwise multiplication to produce a feature vector upon which a classifier can be trained.

3 Experiments

Different variations upon this broad architecture were explored in three experiments. All tasks involve multi-class sentiment analysis on the IMDB dataset of [5], where the networks must learn to classify text from movie reviews into one of 8 possible star-ratings determined by reviewer's sentiment. Each minibatch of data seen by the network during training consists of the first 200 hundred tokens of each movie review, converted to a sequence of GloVe vectors [11]. The GloVe vectors were pre-trained from the Wikipedia 2014 and Gigaword 5 datasets and are 50-dimensions each. Tokenization was performed by the NLTK python library [12] and discarded punctuation and nonwords. All models were implemented in TensorFlow [13].

3.1 Dataset

The IMDB Large Movie Dataset includes 50,000 movie reviews written by users and labeled with low (i.e., star ratings of 1, 2, 3, or 4) or high (i.e., 7, 8, 9, 10) sentiment values. No single movie is reviewed more than 30 times, and in order to decorrelate the data, no movie reviewed in the training set is also reviewed in the test set.

The longest review in the dataset is 2,445 tokens long, whereas the shortest is only 6 tokens. At training time, all examples are standardized to be the first 200 tokens of the review. Reviews shorter than 200 tokens are padded with zero-vectors.

This dataset is well-studied in the domain of sentiment analysis. Wang and Manning [14] evaluate a zoo of different classifiers based on Naïve Bayes and Support Vector Machines (SVM) on popular sentiment datasets and report the state-of-the-art for the large movie dataset near 91.22% accuracy. However, they only evaluate binary classification accuracy. To my knowledge, there is no previously published state-of-the-art for the IMDB dataset based on the 8-way sentiment classification task described here. Given the difficulty of fine-grained sentiment analysis over binary classification, we can nevertheless treat Wang & Manning's figure as a reasonable upper bound for the performance of my models.

3.2 Experiment 1

The first experiment evaluated the LIM against an architecture including only bigram filters and one including only trigram filters in the convolutional trunk. A baseline, purely recurrent LSTM was also trained and evaluated.

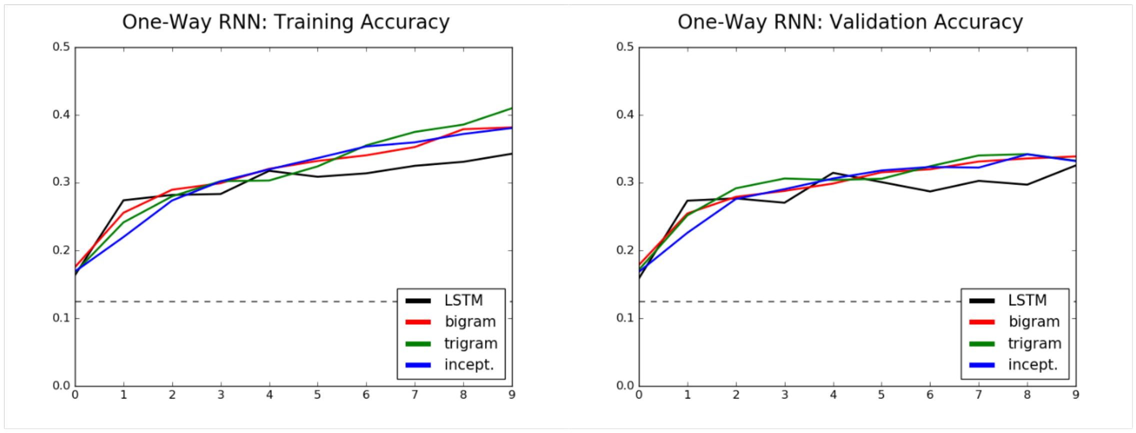

Figure 5. Learning and validation curves for all one-way RNN models. X-axis on 5-epoch intervals.

Each network's recurrent trunk was made up of a single LSTM layer, where outputs were extracted only from the last time-step (before zero-padding) for each of the variable length reviews. Elementwise multiplication was used to merge the outputs from the two trunks. Dropout was used to regularize the network and occurred after the nonlinearities of each FC layer. All models trained for 50 epochs, where training and validation accuracy was recorded every 5 epochs.

Learning Rate	1e-5
Exponential Decay	96% every 8 epochs
Dropout Probability	10%
Batch Size	20
Activation Functions	Exponential Linear Unit (ELU)
FC Layers in Classification Head	3
Units per FC layer	300
Total LSTM units	300
Total Convolutional Filters	300

Figure 6. Final hyperparameter settings chosen throughout all 3 experiments.

The results on the validation set indicated an advantage for the bigram-based model, but all networks including a convolutional trunk demonstrated a slight improvement in accuracy above the LSTM baseline.

One-Way	LSTM Only	Bigram-LSTM	Trigram-LSTM	LIM-LSTM
Train	34.27%	38.16%	40.98%	38.06%
Validation	32.54%	**33.86%**	33.24%	33.18%

Figure 7. Final training and validation accuracy for Experiment 1 models.

3.3 Experiment 2

The next study evaluated the performance of similar models that make use of bidirectional networks in the recurrent trunk. Bidirectional RNNs consist of two sub-networks that process sequences forward and in reverse order respectively, before combining their outputs into a single hidden representation. Although merging of the two networks typically occurs through concatenation, recent work in deep learning has demonstrated the effectiveness of simple elementwise multiplication in generating meaningful feature vectors from multiple sources [15]. Both methods for merging the networks are tested here.

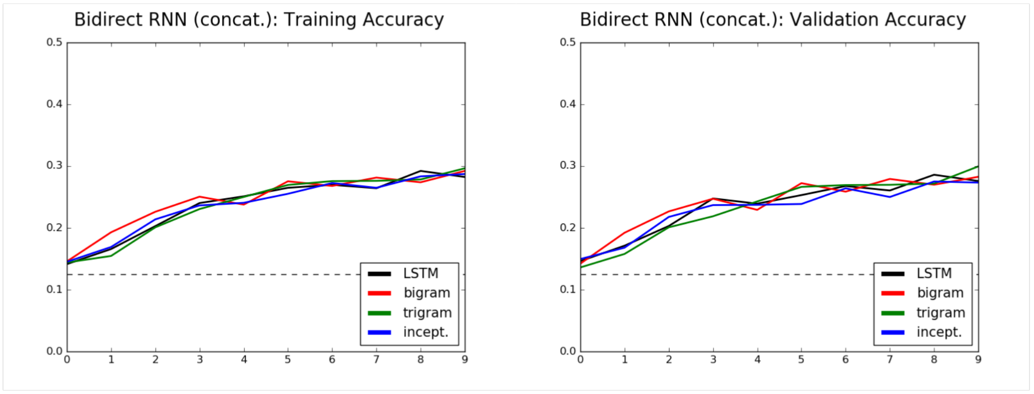

Figure 8. Learning and validation curves for concatenated models. X-axis on 5-epoch intervals.

In the condition making use of elementwise multiplication, the bigram and inception modules emerged as the winners, now by a more significant margin.

Elem. Mul.	LSTM Only	Bigram-LSTM	Trigram-LSTM	LIM-LSTM
Train	30.72%	33.19%	28.94%	33.58%
Validation	28.76%	**31.10%**	26.80%	**31.89%**

Figure 9. Final training and validation accuracy for multiplicative models.

Surprisingly, the opposite result was born out when concatenation was used. In these conditions, the bigram and inception modules performed closer to the level of the baseline, whereas the trigram model took the lead.

Concatenate	LSTM Only	Bigram-LSTM	Trigram-LSTM	LIM-LSTM
Train	28.24%	29.22%	29.64%	28.72%
Validation	27.58%	28.26%	**29.94%**	27.33%

Figure 10. Final training and validation accuracy for concatenated models.

Between the two merge strategies, elementwise multiplication seemed to produce better results altogether. However, it is worth noting that in order to produce results comparable to the first experiment, the same amount of training data (10,000 examples) was held fixed. Because both these models (and especially the one using concatenation) contain more parameters, yet train on the same amount of data for the same number of epochs, they may be underfitting.

3.3 Experiment 3

Given the close accuracies of the previous models, the final study sought to test how the convolutional networks performed in isolation of the recurrent trunk. A similar experiment to the previous 2 was repeated, but the models were trained for an additional 30 epochs each. Furthermore, the number of filters and hidden units were scaled between conditions.

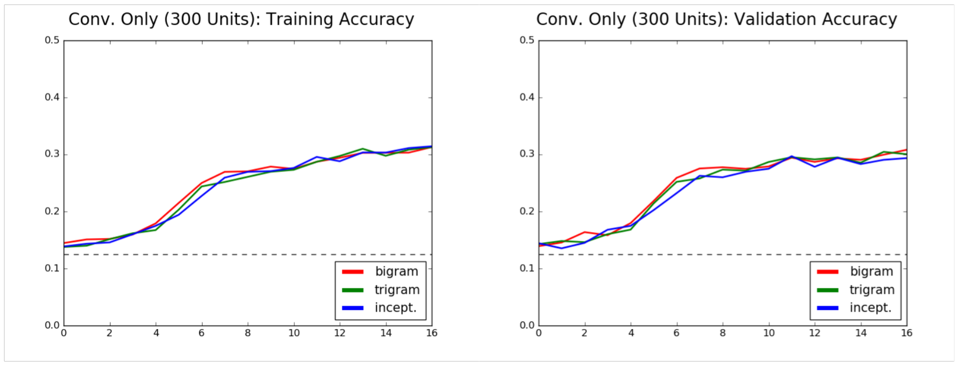

Figure 11. Learning and validation curves for small CNN models. X-axis on 5-epoch intervals.

In the first condition, the convolutional models featured 300 kernels in total (100 allocated to each extractor type in the case of the inception module) followed by 300-dimensional FC layers. In the second condition, this hyperparameter was set at 600.

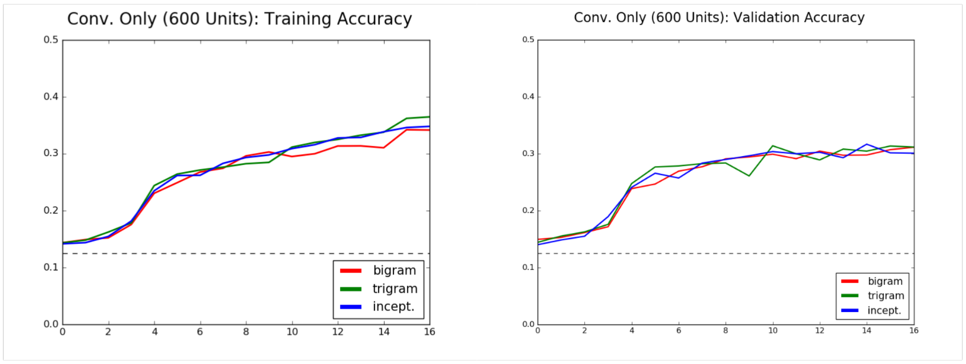

Figure 12. Learning and validation curves for larger CNN models. X-axis on 5-epoch intervals.

	Bigram300	Trigram300	LIM300	Bigram600	Trigram600	LIM600
Train	31.29%	31.28%	31.43%	34.17%	36.48%	34.81%
Validation	30.82%	30.04%	29.36%	**31.10%**	**31.14%**	30.06%

Figure 13. Final training and validation accuracy for CNN-only models

Again the bigram model appeared the most desirable, leading to greater accuracy with slightly less overfitting. Although the CNNs converged to similar levels of accuracy as the two-trunk architectures, they demonstrated an unusual learning pattern. Each model remained relatively close to chance accuracy for about 15 epochs before jumping to a more respectable range.

3.3 Test Set Results

Finally the 10 most promising models were evaluated on a held out test set consisting of 5000 examples. The accuracy and number of weights in each network is shown below.

Model	**Accuracy**	**Parameters**
One-Way LSTM	31.84%	694,508
One-Way Bigram-LSTM	**33.22%**	**815,108**
One-Way Inception-LSTM	32.62%	825,108
Elem. LSTM	29.48%	1,206,008
Elem. Bigram-LSTM	29.22%	1,326,608
Elem. Inception-LSTM	30.58%	1,336,608
Bigram 300	29.54%	303,608
Inception 300	28.34%	313,608
Bigram 600	29.84%	1,147,208
Inception 600	30.58%	1,167,208

Figure 14. Final test set accuracies and parameters for 10 best models.

These results largely bear out expectation established by the validation accuracies, with the bigram-LSTM model developed in experiment 1 attaining the best performance by a small margin. However, when the number of parameters is accounted for, the CNN-only models look desirable.

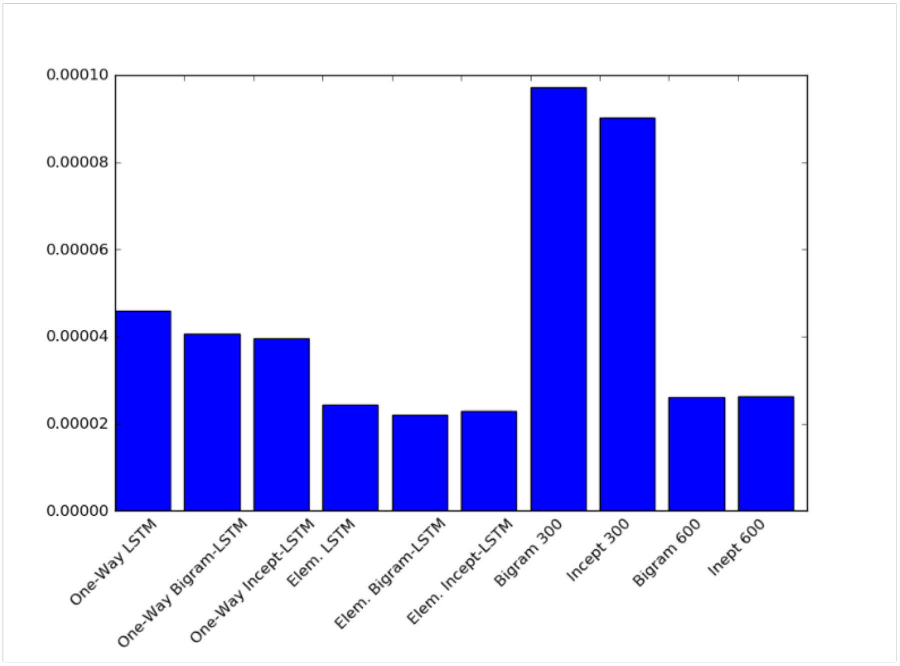

Figure 15. Accuracy per parameter of each model.

Figure 15 displays the accuracy-over-parameters (AOP) of each model. The important thing to note is that the CNNs had half the parameters of the smallest merged model, and an order of magnitude fewer parameters than the bidirectional networks, yet performed on par with them. As a result, each weight in the smaller CNNs can be thought of as accounting for about twice as much of the fraction of accuracy accounted for by weights in the One-Way LSTM.

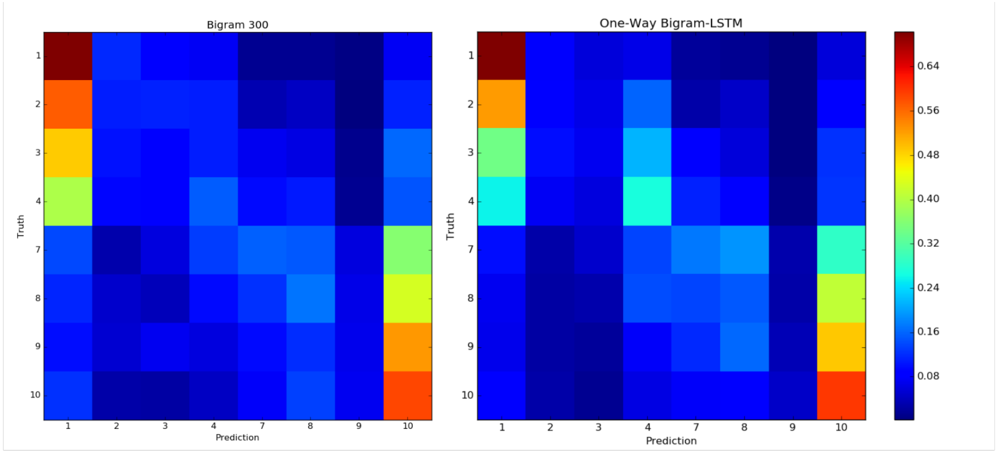

Figure 16. Confusion matrices for best model based on AOP (left) and best based on un-weighted accuracy (right).

The networks' patterns of errors are also informative. **Figure 16** displays the confusion matrices of the best model according to the AOP metric (left), and the best according to classification accuracy (right). Both models reasonably favor classifications on the two poles of the sentiment spectrum (1 vs. 10-star ratings), rarely confusing a negative sentiment review with a positive one. However, both models encounter difficulty classifying less polar reviews and tend to mistake a review's class with its closest pole.

4 Conclusion

This work evaluated 18 deep learning architectures on a novel, fine-grained sentiment classification task for the well-known IMDB Large Movie Dataset. Although I present a CNN module inspired by the computer vision model GoogleNet, all evaluations point to a simpler, bigram-based CNN as the best model for this task. This result accords with the findings of Wang & Manning [14], who similarly demonstrated that a modified SVM trained on bigrams outperformed more elaborate state-of-the-art models. Whether the power of bigram models depends on the statistics of this particular dataset would be an important question for follow-up work.

Interestingly, merging CNN and RNN into a single network also appeared to improve accuracy. Although the argument was made that the CNN models attain the greatest accuracy for their number of parameters, the combined architectures outperformed all other models on pure accuracy scores. I attribute this to the network's ability to learn *n*-grams significant to the objective function, while still modeling word-by-word sequences through recurrence.

In summary, although merged networks appear useful for sentiment classification, little empirical support has been found for the effectiveness of LIM in particular. However, different modifications may improve the proposed model. For example, just as [2] implemented a pragmatic inception module that differed from their original design, the LIM may benefit from input-pooling or 1x1 convolutions. Given the baselines established by [14], it may also be the case that the dominance of the bigram networks is an artifact of the IMDB dataset, and the LIM may benefit from testing in other domains.

5 References

[1] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

[2] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1-9).

[3] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. *arXiv preprint arXiv:1512.03385.*

[4] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324.

[5] Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011, June). Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1* (pp. 142-150). Association for Computational Linguistics.

[6] Collobert, R., & Weston, J. (2008, July). A unified architecture for natural language processing: Deep neural networks with multitask learning. In*Proceedings of the 25th international conference on Machine learning* (pp. 160-167). ACM.

[7] Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188.*

[8] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-178.

[9] Socher, R. (2016). Lecture 8: Recap, projects and Fancy Recurrent Neural Networks for Machine Translation. *CS 224: Deep Learning for NLP*. Stanford University.

[10] Jurafsky, D., & Martin, J. H. (2014). Chapter 4: N-grams. In *Speech and Language Processing*. Draft of September 1. Pearson.

[11] Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global Vectors for Word Representation. In *EMNLP* (Vol. 14, pp. 1532-1543).

[12] Bird, S. (2006, July). NLTK: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions* (pp. 69-72). Association for Computational Linguistics.

[13] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... & Ghemawat, S. (2016). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv preprint arXiv:1603.04416.*

[14] Wang, S., & Manning, C. D. (2012, July). Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2* (pp. 90-94). Association for Computational Linguistics.

[15] Oh, J., Guo, X., Lee, H., Lewis, R. L., & Singh, S. (2015). Action-conditional video prediction using deep networks in atari games. In *Advances in Neural Information Processing Systems* (pp. 2845-28