Deep CNN-LSTM with Combined Kernels from Multiple Branches for IMDb Review Sentiment Analysis

Alec Yenter
Department of Computer Science
California State University
Fullerton, California 92831
alecyenter(at)csu.fullerton.edu

Abhishek Verma
Department of Computer Science
New Jersey City University
Jersey City, NJ 07305
av56(at)njit.edu

Abstract— **Deep learning neural networks have made significant progress in the area of image and video analysis. This success of neural networks can be directed towards improvements in textual sentiment classification. In this paper, we describe a novel approach to sentiment analysis through the use of combined kernel from multiple branches of convolutional neural network (CNN) with Long Short-term Memory (LSTM) layers. Our combination of CNN and LSTM schemes produces a model with the highest reported accuracy on the Internet Movie Database (IMDb) review sentiment dataset.**

Additionally, we present multiple architecture variations of our proposed model to illustrate our attempts to increase accuracy while minimizing overfitting. We experiment with numerous regularization techniques, network structures, and kernel sizes to create five high-performing models for comparison. These models are capable of predicting the sentiment polarity of reviews from the IMDb dataset with accuracy above 89%. Firstly, the accuracy of our best performing proposed model surpasses the previously published models and secondly it vastly improves upon the baseline CNN+LSTM model. The capability of the combined kernel from multiple branches of CNN based LSTM architecture could also be lucrative towards other datasets for sentiment analysis or simply text classification. Furthermore, the proposed model has the potential in machine learning in video and audio.

Keywords—IMDb; sentiment analysis; text classification; neural network; CNN; LSTM;

I. Introduction

With an ever increasing production of films, there is an excess of reviews and opinions of movies. Analysis of these sentiments becomes extremely useful to helping people organize and choose movies. Classification of review sentiment helps extract meaningful information from opinions and is a critical portion of analysis.

There are different tiers to sentiment analysis: document view, sentence view, and aspect view [9] [25]. This paper focuses on a document view where the entirety of each review is classified as positive or negative. Additionally, learning can be supervised or unsupervised. Since our chosen IMDb review sentiment dataset has labels, this paper utilizes supervised machine learning.

Neural networks have become a popular approach to a variety of problems that require a complex solution, such as computer vision and speech recognition. GoogLeNet [10], ResNet [12], and VGG-16 [13] are all powerful deep networks that have made leaps of progress in image/video classification. The further advancement of the aforementioned models in [14], [15], and [16] prove that there is still improvement possible in deep neural networks. This paper builds a deep neural network towards text classification in sentiment analysis. While textual classification can be processed on a letter level [4], our proposed model centers on word level learning.

Our novel network successfully combines several useful ideas from various networks. A portion of the proposed network is inspired from the GoogLeNet Inception model in [10]. This portion uses multiple branches of convolution and further adds the recurrent neural layers. The resulting model outperforms other machine learning models on the same movie dataset.

The rest of this paper is organized into six main parts. Section II presents the basic sections of the network and reviews related literature in the field of text classification. Section III defines the details of the IMDb review sentiment dataset. Section IV presents the detailed structure of the model layer by layer. Section V describes the specific configuration of the model that results in the highest accuracy. Section VI discusses the results and challenges. Section VII concludes the paper.

II. Background

To build a context to the paper, the following sub sections review related literature of neural networks and text classification.

A. Deep Neural Networks (NN)

Groundbreaking progress in machine learning has been made through the use of deep neural networks. The ability for a neural network to imitate the brain's processes is a valuable method for problems in textual, visual, and other signals. The basics for neural networks are layers each with a specific number of nodes. Input data in the first layer and connections are given to the next layer. This is repeated until the final layer, where the output is produced. There are multiple types of networks and layers; in this paper, we will specifically use three layers: Fully-Connected Layers, Convolutional Layers, and LSTM layers.

1) Fully-Connected (Dense) Layers are simple layers where every neuron from the previous layer is connected to every neuron in the dense layer [11]. These layers can do basic learning while reshaping the input data, for instance reducing the output to one neuron.

2) Convolutional Neural Networks (CNN) are a specific type of neural network that can work well with spatial data. CNNs are particularly useful with images for tasks such as classification or segmentation. Convolutional layers use only certain connections from previous layer; specifically, local neurons are connected to the neurons of the next layer [11]. This method causes the layer to gain more of an understanding of the genreal view of the inputs.

3) Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network (RNN). RNN neurons have a connection to the previous neuron state in addition to the layer inputs. RNNs are particularly benefitial to data that is sequential or that can value a contextual view; therefore, RNNs can be impressive in text classification [19]. LSTMs are a form of RNNs where newer information in the neurons is more critical than older information [24]. LSTMs have been useful in deep learning on video footage in [17] and [18] because of films sequential nature. Similarly, they are useful on sequential text data, because, while LSTM can incorporate the context of a parargraph and/or a sentence, the most recent words hold the most weight on the current neuron state.

B. *Neural Network Text Classification*

When given textual data, a typical objective is to classify the information for analytical or statistical purposes. Text categorization is performed in [20], while [21] and [22] performed sentiment classification, all on social media datasets using neural networks. In [2] and [23] authors compared multiple machine learning techniques on the sentiment analysis of movie review. In [2] authors compared multiple *n*-gram machine learning approaches on the IMDb review sentiment dataset used in this paper. The data was preprocessed before being vectorized and fed into various configuration of machine learning algorithms. Many of these techniques in [2] reached high accuracies in the 80% with the best configuration "Unigram + Bigram + Trigram" reaching the maximum accuracy of 88.94%.

While there are many useful mathematical algorithms, this paper focuses on neural network approaches to textual classification. These approaches have been split into character-level classification and word-level classification for the following discussion.

1) Character-Level Text Classification is a newer approach that focuses on the letters of the text. This approach has the benefit of avoiding the need for a dictionary or an understanding of the language, but instead defines an albhabet for the data. [4] explored the use of character-level convolutional neural networks for classification of textual documents. The use of the sequence of letters as a signal sequence produced comparable results to other methods such as word-level classification. [4] developed a neural network of nine layers: six convolutional layers followed by three fully-connected layers. These layers were accompanied my max-pooling layers to help the nework handle the nine-layered depth.

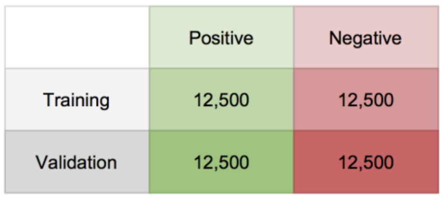
In [6] authors built onto character-level text classification by using a CNN to produce a form of word embedding from the character inputs of a word. For each word, letter embeddings are concatenated along axis 1, convoluted through multiple kernels, and finally pooled and concatenated into a flat layer. The last layer produces a representation of the word that is fed through a highway network to a LSTM network for classification. The benefit of this hybrid method is that no dictionary is needed, yet the network learns on a word-level basis that shows a human-like understanding of the language.

2) Word-Level Text Classification is a more traditional method to text classfication with neural networks. Authors in [5] created a shallow CNN that classifies using multiple different kernels. In this network, a convolutional layer will look at *n* words at a time when applying the filters and pooling-over-time. The network would use multiple kernel sizes and concatenate the results; therefore, the network finds context from the *n* number of words nearby the word. The final connected layer uses this information for classification. This network is simple but effective in textual classification.

Authors in [1] use IMDb review data with a new LSTM neural network to classify sentiment of the review. The review data contained no neutral data. The dictionary was limited to the top 2000 most used words and each review sequence was capped at 100 words and padded with zeros if less than the max. The proposed LSTM layer is a biologically-inspired additive version of a traditional LSTM that produced higher loss stability, but lower accuracy. The best accuracy achieved between both LSTM models was still under 85%. The use of an LSTM on textual data gives better contextual view of words than a CNN.

III. Description of Dataset

The ACL Internet Movie Database (IMDb) dataset used was created in [3] for learning word vectors. The dataset consists of 100,000 textual reviews of movies; half (50,000) of the reviews are for testing and have no label. The other (50,000) reviews are paired with a label of 0 or 1 to represent negative and positive sentiment, respectively. These labels were linearly mapped from the IMDb's star rating system where reviewers can rate a movie a certain number of stars from 1 to 10. Fig. 1 illustrates the organization of the dataset. The reviews with labels are split in half; each set has 12,500 positive reviews and 12,500 negative reviews to keep the data balanced.



	Positive	Negative
Training	12,500	12,500
Validation	12,500	12,500

Figure 1. Shows the split of dataset [3].

There are at most 30 reviews for any one movie. The mean number of words per review is 234.76 with a standard

Positive Example:

Although this was obviously a low-budget production, the performances and the songs in this movie are worth seeing. One of Walken's few musical roles to date. (he is a marvelous dancer and singer and he demonstrates his acrobatic skills as well - watch for the cartwheel!) Also starring Jason Connery. A great children's story and very likable characters.

Negative Example:

Not only is it a disgustingly made low-budget bad-acted movie, but the plot itself is just STUPID!!!
A mystic man that eats women? (And by the looks, not virgin ones)
Ridiculous!!! If you´ve got nothing better to do (like sleeping) you should watch this. Yeah right.

Figure 2. Example of two reviews from the IMDb dataset [3].

deviation of 172.91 words. Certain punctuation and symbols are included, such as "?", "!", and ":(". Collectively, the dataset contained 88,585 different words across all reviews. Fig. 2 gives two examples of movie reviews in the dataset: one positive and one negative.

The deep learning library Keras [26] provides a simple import method to retrieve these reviews in a preprocessed format. The function grabs the reviews and encodes them into a sequence of word indices according to a dictionary of the *D* most frequently used words in the dataset, where *D* is given. For further flexibility, the indices are ordered by the frequency of the words, such that the word mapped to index 2 is the 2nd most common word in the dataset. Index 0 is reserved for unknown words that are not in the dictionary. The reviews are then padded to fit a desired maximum sequence length; longer reviews are truncated and shorter reviews are padded with zeros. As a final step, the first layer of the neural network converts the indices to embeddings of *E* dimension.

For this paper's experiments, the dictionary was capped at 5,000 words with a maximum padded sequence length of 500 words. The indices were embedded into 32 dimensions.

IV. Proposed Combined Kernels from Multi-Branch CNN with LSTM Model

The proposed method in this paper utilizes a CNN and a LSTM on word-level classification of the IMDb review sentiment dataset. The method combines versions of the networks from [5] and [1]; novelty of the proposed network lies in having combined kernels through multiple branches that accept the data and perform convolution. The output of the CNN branches is fed into an LSTM before being concatenated and sent to a fully-connected layer in order to produce a single, final output. The network is trained and tested in mini-batches between 16 and 128. Embedding

The first layer of the network accepts the input reviews as a sequence of indices and embeds each word into a vector of a specific size *e*. (e.g. a vector of 500 word indices embedded at 32 becomes 500 vectors of length 32). The embedding layer is a matrix of trainable weights that, through matrix multiplication, produce the vectors for each word index. Therefore, during training, the embedding layer improves upon the embeddings of each word.

A. Convolution

The output of the embedding layer is given to each branch of *b* branches. Each branch starts with a 1-dimensional convolution layer of kernel size *c* specific to that branch. The kernel during 1-dimensional convolution is of shape kernel size by embedding size (*c* x *e*). The kernel will then perform convolution on whole words instead of the typical 2-dimensional convolution that would filter partial widths and cut words into pieces. The layer produces multiple outputs with the use of multiple filters *f*.

The purpose of the CNN layer is to view word combinations of the kernel size *c*. The result is an understanding of words when used with other words. For example, when $c=3$, the layer views 3 words at a time and, therefore, establishing a sense of 3-word combinations. This layer's output has a shape of input height by filters (*words* by *f*).

B. Activation

Each branch applies a rectified linear unit (ReLU) activation of the CNN layer's output; this layer replaces any negative outputs with zero. The ReLU layer is used in order to introduce non-linearity into the network. The output of this layer is the same shape as the input shape.

C. Max Pooling

Each branch undergoes 1-dimensional max pooling following ReLU activation; this layer converts each kernel size of the input into a single output of the maximum observed number. The result is a reduced, down-sampled version of the input. The purpose of the layer is to reduce overfitting, while allowing for further processing. Similar to the CNN layer, the 1-dimensional max pooling kernel shape is fit to the width of the data, so that the parameter kernel size *p* implies a kernel shape of *p* by data width. This technique allows pooling to be applied with the understanding that data is composed of whole words. The output of this layer is a reduction in height according to kernel size *p* (input height ÷ *p*).

D. Dropout

After max pooling, each branch goes through a dropout layer; this layer randomly sets a portion of the inputs to 0. The dropout is applied to specified *d* fraction of the inputs. This layer serves to prevent overfitting and generalize the network to not focus on specific pieces of input. The output shape is equivalent to the input shape.

E. Batch Normalization

The next layer for each branch is batch normalization; this layer simply normalizes the distribution for each batch after dropout. The purpose of batch normalization is to reduce internal covariate shift and therefore lead to convergence at a faster rate. The output of this layer holds the same shape as the input.

F. LSTM

The final layer for each branch is a LSTM layer with a specified number of units *l*. The LSTM is used because of the

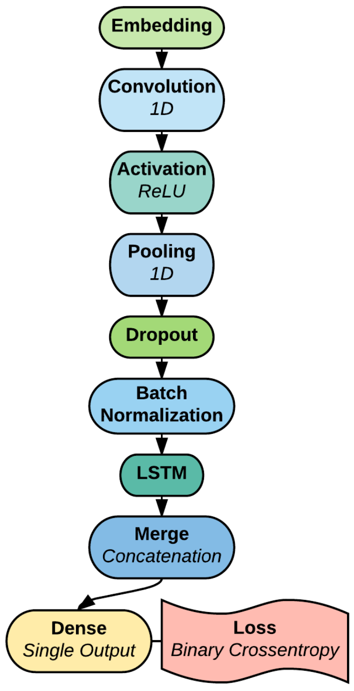


Figure 3. Diagram of basic structure of CNN based LSTM network. We use this in our experiments as a baseline model to compare with our novel model.

nature of sequential data. The layer's persistence allows knowledge of previous input (convoluted word combinations) to influence subsequent input. The output has a length of the number of units *l*.

G. *Concatenation*

The branches are finally merged together through concatenation. The LSTM layers' outputs are combined together in an array. The output shape of this layer is equal to the summation of the output of all the branches (l x b) .

H. *Dense*

The last layer is a fully-connected layer from the concatenated input to a single output. The layer is followed by a simple sigmoid activation function to conform the output between 0 and 1. The final yield is a single output.

I. *Loss Function and Optimizer*

The network is compiled with a binary cross entropy loss function; this loss calculates loss with two classes (0 and 1). For this paper's purposes, 0 represents negative sentiment and 1 represents positive sentiment. The loss is calculated on the single and final output of the dense layer.

The network is also compiled with an optimizer; Adam, RMSprop, and Stochastic Gradient Descent (SGD) were the optimizers used for testing during the experiments. Each optimizer was used with varying learning rates and learning rate decay parameters. Figure 3 gives a visualization of the baseline model and fig. 4 gives a visualization of our best performing proposed network.

V. Experimental Setup

The hardware used for experimenting different networks was a dual Intel Xeon E5-2690 v3 2.60GHz processors (24 physical cores / 48 logical cores) with a single NVIDIA GeForce GTX TITAN X GPU with 12 GB of VRAM and a 256 GB RAM. The machine ran Keras 2.0.4 [26] on Ubuntu 14.04.5 using TensorFlow 1.1.0 [27] backend on the GPU.

Many experiments were attempted to determine the appropriate parameters and network structure. The IMDb review sentiment dataset was used for all experiments. The dataset was preprocessed to a dictionary size of 5,000 words with a zero-padded maximum sequence of 500 words per a review; anymore data became insignificant to the networks objective. The best network used a batch size of 128. The following sections establish the best parameters approximated by the experiments.

A. *Embedding*

An embedding size of 32 best fit the dataset. Attempts to use GoogleNews word2vec pre-trained embeddings from [7] and Wikipedia fastText pre-trained embeddings from [8] (as untrainable word embeddings) proved to have no positive effect. Therefore we do not include it in our proposed model.

B. *Convolution and Activation in Multiple Branches*

The largest alterations were determined by the number of branches to be used; this was directly tied to the kernel sizes used. The optimal 1-dimensional kernel sizes were three, five, seven, and nine; therefore, four branches were setup. Viewing words in these kernel sizes proved to extract the most critical information and produce higher accuracy. The optimal number of filters was found to be 128. Additionally, the convolutional layer was accompanied by a ridge regression (*l2*) kernel regularizer to help prevent overfitting. The *l2* parameter was set to 0.01.

The ReLU activation layer proved crucial to reaching a higher accuracy. This layer has no parameters.

C. *Max Pooling, Dropout, and Batch Normalization*

Although max pooling assisted with the chief issue of overfitting, large pooling sizes proved to decrease accuracy. The optimum kernel size during the experiments was 2; reducing the height of the input by half.

The dropout layer was recognized to be the best option to reduce overfitting. The dropout was set at a rate of 0.5 to force other weights to help generalize the network. While ensuring convergence, this method led to higher accuracy and a better understanding of the data.

Batch normalization was added to the network to help with overfitting with the LSTM network. This layer has no parameters.

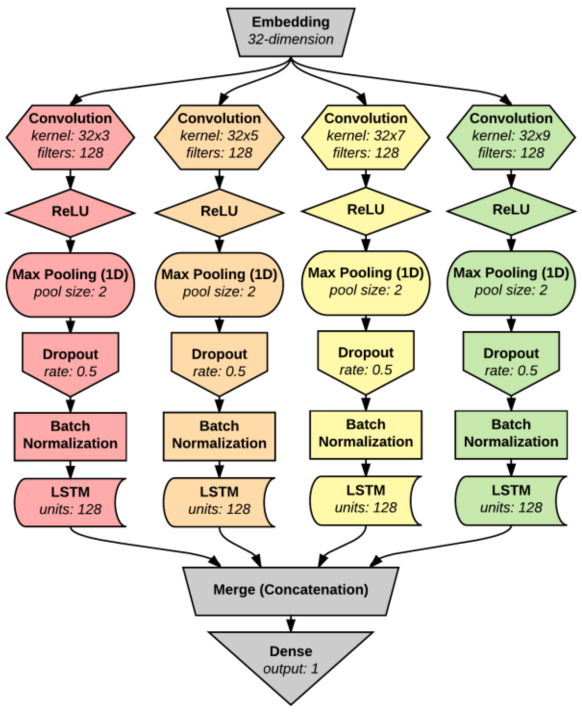


Figure 4. Diagram of the best performing proposed model (Model_63).

D. *LSTM, Concatenation, Dense Layer, and Optimizer*

Each branch's LSTM layer has 128 units. Any less or more units reduce accuracy or increase overfitting. The merging layer and dense layer have no tuning parameters. While the optimizer did not have a profound effect, RMSprop showed the best results. The learning rate was increased to 0.01 and the learning rate decay was set to 0.1. These parameters reached the highest overall accuracy. Fig. 4 illustrates the entirety of the network.

VI. Results and Analysis

The best performing proposed network reached 89.5% accuracy. This accuracy was achieved despite the main challenge of overfitting. Many models and variations were tested to attempt to reduce overfitting and generalize the learning. Figure 5 shows performance of best proposed model. Figure 6 and table 6 show the performance of various models.

A. *Accuracy*

The achieved accuracy surpassed other models' results on the same IMDb review sentiment dataset. Both the traditional and the proposed LSTM models in [1] plateaued at an accuracy just over 80%, with a maximum under 85%. The accuracy in [1] was used as a baseline accuracy for a simple, lone LSTM model. Our model's performance can also be compared to other machine learning methods. Authors in [3] proposed a hybrid supervised-unsupervised model when they created the IMDb review sentiment dataset. Their highest reported accuracy was 88.89% from their full model with additional unlabeled reviews and bag of words vectors. Authors in [2] experimented with multiple variation of machine learning models on the IMDb dataset. The best accuracy achieved in [2] was 88.94% from their combined Unigram-Bigram-Trigram model. From reviewing these other machine learning models, it is clear that our proposed method outperforms prior work.

B. *Overfitting*

During the experiments, overfitting became the primary issue. Within a few epochs, 80% was easily achieved in most networks. However, the networks would begin to memorize the data and overfit. Figure 6 shows the accuracy comparison of five variants of proposed models from epoch 3 to epoch 10. Except for model_63 the rest of the models suffered from performance degradation and overfitting after epoch 4.

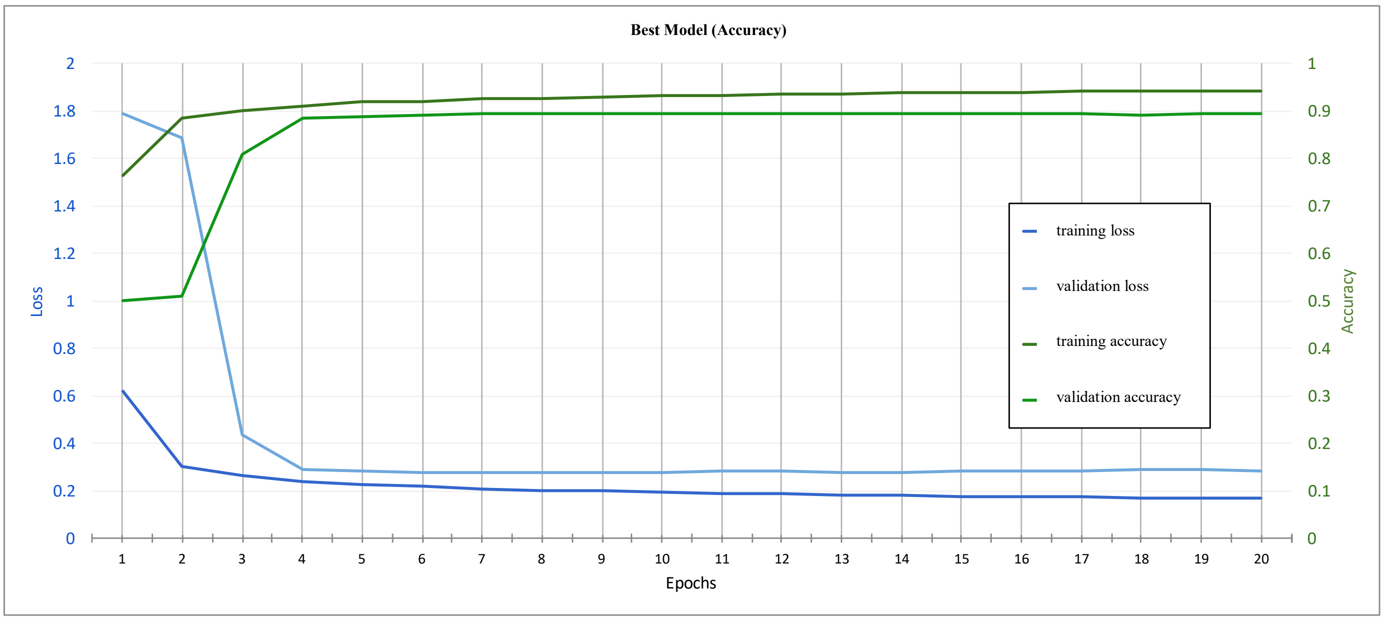


Figure 5. Graph of accuracy of top performing proposed model (Model_63).

Table 1. Table comparing the structure, parameters, and highest accuracy of the top 5 performing proposed models in the experiments, as well as the base model.

Proposed Models	Convolution			Activation	Max Pooling	Branch Dropout	Batch Normalization	LSTM	Merge Dropout	Optimizer			Accuracy
	Branches / Kernel Sizes	*Filters*	*Kernel Regulaizer*	*Type*	*Pool Size*	*Rate*	*Present*	*Units*	*Rate*	*Type*	*Learning Rate*	*Learning Rate Decay*	*Maximum*
Model_08	3/4/5	32	None	ReLU	2	0	yes	100	0	Adam	0.001	0	0.8922
Model_16	3/4/5	32	L2(0.01)	ReLU	2	0.5	yes	100	0	Adam	0.001	0	0.8934
Model_43	2/3/4/5/6/7	128	L2(0.01)	ReLU	2	0	yes	128	0.8	Adam	0.001	0	0.8914
Model_57	3/5/7/9	128	L2(0.01)	ReLU	2	0.5	yes	128	0	RMSprop	0.001	0	0.8936
Model_63	3/5/7/9	128	L2(0.01)	ReLU	2	0.5	yes	128	0	RMSprop	0.01	0.1	**0.895**
Base Model	5	64	None	ReLU	4	0.25	no	70	0	Adam	0.001	0	0.8498

1) Layers were able to help recude overfitting; these included Max Pooling, Dropout, and Normalization. By reducing dimensionality, max pooling assisted with generalizing the learning process; max pooling performed best at a size of 2 (1-dimensionally) across all networks. Dropout layers were assessed at different locations in the network. They were found to be most helpful after max pooling and before batch normalization or, in some cases, after concatenation of all the layers. Dropout was typically found to be helpful at a rate higher than or equal to 0.5. Batch normalization was found to always be helpful in both increasing accuracy and reducing overfitting.

2) Parameters of regularization, learning rate, and decay played a large part in reducing overfitting. Although any activity regularizer reduced accuracy with no other benefit, a ridge regression regularizer of the kernel decreased overfitting. The optimal kernel regularizer was L2(0.01). Changing the learning rate from its default did not typicallly help unless accompanied by a learning rate decay. The best performing model utilized a higher learning rate of 0.01 with a learning rate decay of 0.1. As shown in Fig. 5, this combination caused the accuracy percent to jump to the high 80's and then stay there with little alterations. Without the decay, the training accuracy would drop down to below 85% because of overfitting. The loss also did not fluctuate too much after reaching its lowest.

3) Depth had to be kept low in order to keep the network from decaying. The deeper the network, the greater the likelihood of overfitting. Since the data was more simple than images, deeper networks were not able to extract anymore useful information. Attempts to add additional dense or convolutional layers were fruitless.

C. Comparison of Other Experimented Models

Table 1 shows the comparisons between the proposed 5 top networks structure and parameters. Fig. 6 graphs and compares the accuracy of the top 5 models. Model_63 is the best performing proposed network; its use of learning rate decay resulted in the least overfitting.

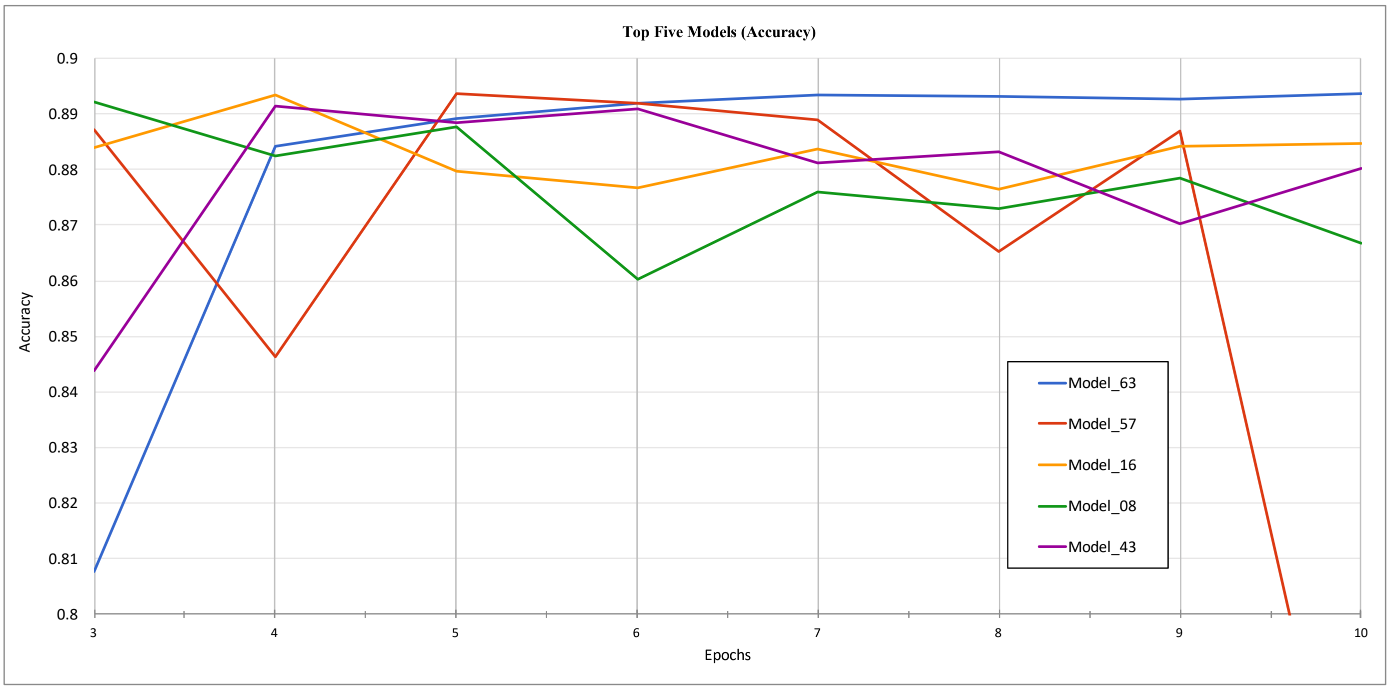


Figure 6. Accuracy graph of the top 5 performing proposed models.

VII. Conclusion and Future Work

This paper shows the effectiveness of the novel combined kernel from multi-branch convolution with LSTM network on the IMDb review sentiment dataset. It is evident that different convolution branches are able to pull significant information from textual information in a shallow network. Additionally, LSTM layers are able to use the information to dig deeper into classifying the reviews. The accuracy of our best performing proposed model surpasses the previously published models and vastly improves upon the baseline CNN+LSTM model. While overfitting was a challenge for most versions of the model, proper layers and parameters were able to reduce the deterioration and reach new accuracy highs on the dataset.

One area of investigating that could be beneficial is the use of different types of word representations. While the embedding was used, other methods, such as bag of words and TF-IDF, could be mixed in to produce different results. While it was established that GoogleNews embeddings from [7] and Wiki FastText embeddings from [8] did not add to the accuracy, other pre-trained models may have better effect. There may also be other pre-processing techniques that could raise the efficacy of the model.

References

[1] L. Rahman, N. Mohammed, and A. Kalam Al Azad, "A new LSTM model by introducing biological cell state." *In Electrical Engineering and Information Communication Technology (ICEEICT), 2016 3rd International Conference on*, pp. 1-6, 2016.

[2] A. Tripathy, A. Agrawal, and S. Kumar Rath, "Classification of sentiment reviews using n-gram machine learning approach." *Expert Systems with Applications*, vol. 57, pp. 117-126, 2016.

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

[4] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification." *In Advances in neural information processing systems*, pp. 649-657, 2015.

[5] Y. Kim, "Convolutional neural networks for sentence classification," arXiv preprint arXiv:1408.5882, 2014.

[6] Y. Kim, Y. Jernite, D. Sontag, and A.M. Rush, "Character-aware neural language models," *In Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[7] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality." *In Advances in neural information processing systems*, pp. 3111-3119, 2013.

[8] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," arXiv preprint arXiv:1607.04606, 2016.

[9] R. Feldman, "Techniques and applications for sentiment analysis," *Communications of the ACM*, vol. 56, no. 4, pp. 82-89, 2013.

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

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, 1998.

[12] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” arXiv preprint arXiv:1602.07261, 2016.

[13] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” arXiv preprint arXiv:1409.1556, 2014.

[14] A. Verma and Y. Liu, "Hybrid Deep Learning Ensemble Model for Improved Large-Scale Car Recognition," *IEEE Smart World Congress*, San Francisco, CA, 2017. (to appear)

[15] H. Vo and A. Verma, "New Deep Neural Nets for Fine-Grained Diabetic Retinopathy Recognition on Hybrid Color Space," *the 12th IEEE International Symposium on Multimedia*, Dec. 11-13, 2016, San Jose, CA, USA.

[16] H. Al-Barazanchi, H. Qassim, and A. Verma, "Novel CNN Architecture with Residual Learning and Deep Supervision for Large-Scale Scene Image Categorization," *the 7th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, Oct. 20-22, 2016, New York, NY, USA.

[17] K. Zhang, W.-L. Chao, F. Sha, and K. Grauman, “Video summarization with long short-term memory,” *In European Conference on Computer Vision*, pp. 766–782, Springer, 2016.

[18] J. R. Medel and A. Savakis, “Anomaly detection in video using predictive convolutional long short-term x networks,” arXiv preprint arXiv:1612.00390.

[19] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent Convolutional Neural Networks for Text Classification," *In AAAI*, vol. 333, pp. 2267-2273, 2015.

[20] F. Sebastiani, "Machine learning in automated text categorization," *ACM computing surveys (CSUR)*, vol. 34, no. 1, pp 1-47, 2002.

[21] L. Chen, C. Liu, and H. Chiu, "A neural network based approach for sentiment classification in the blogosphere," *Journal of Informetrics*, vol. 5, no. 2, pp. 313-322, 2011.

[22] A. Severyn, and A. Moschitti, "Unitn: Training deep convolutional neural network for twitter sentiment classification," *In Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), Association for Computational Linguistics*, Denver, Colorado, pp. 464-469. 2015.

[23] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: sentiment classification using machine learning techniques," *In Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, vol. 10, pp. 79-86, 2002.

[24] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[25] J. Mir and M. Usman, "An effective model for aspect based opinion mining for social reviews," *2015 Tenth International Conference on Digital Information Management (ICDIM)*, Jeju, pp. 49-56, 2015.

[26] Keras. (2017). Retrieved June 19, 2017, from keras.io

[27] Tensorflow (2017). Retrieved May 20, 2017, from tensorflow.org