Text Classification: Comparative Analysis of Different Deep Neural Network Architectures

Shefali Agarwal
Computer Science
North Carolina State University
sdagarrwa@ncsu.edu

Suvodeep Majumder
Computer Science
North Carolina State University
smajumd3@ncsu.edu

Varsha Nagarajan
Computer Science
North Carolina State University
vnagara2@ncsu.edu

I. Introduction

Automated categorization of text documents into predefined categories are an important aspect of text classification. In recent years the increasing amount of available documents and huge cost of manually labelling and classifying them, has increased the need of automated machine learning based techniques to categorize documents. Similarly, Sentiment Analysis or opinion mining is also one of the most studied research areas. It can hugely benefit organizations as they are able to understand the emotions and opinions of the users with respect to their product and incorporate necessary changes to make the customer's experience better.

Although different techniques has been developed and employed for such analysis, Deep Neural Networks (DNN) have proven to be one of the most successful techniques among them. The goal of this project is to study and perform a comparative analysis of different DNN architectures such as Convolutional Neural Networks [1], Long Short-Term Memory networks [2] and Hierarchical Attention Networks [3] and further, analyze and evaluate the model's performance on the two selected text classification tasks mentioned above.

II. Datasets

For this study, we used 3 different datasets, one for text categorization and two for sentiment analysis. These datasets were made publicly available after being used for analysis in well established papers or as part of data challenges. Our rationale behind choosing these datasets was to pick data belonging to different domains and of varying sizes in order to properly assess the model's performance from various aspects i.e. (a) time required for training, (b) complexity of model and (c) performance in different domains.

A. *AG News*

The AG News dataset [4] consists of news articles from more than 2000 news sources containing 127,600 news articles divided into 4 different categories of articles. This dataset has been widely used in research and has been cited in over 50 research articles.

B. *Amazon Reviews*

The Amazon reviews dataset [5] contains more than 142 million reviews spanning over 18 years. For the purposes of this project and text classification, we work with a subset of this data and use only the Toys and Games reviews consisting of 167,597 reviews with 5 point ratings.

C. *IMDB Reviews*

The IMDB dataset [6] is a binary sentiment-analysis dataset consisting of 50,000 reviews from the Internet Movie Database, IMDB labeled as positive or negative. The dataset is balanced and contains an even number of positive and negative reviews. Only highly polarizing reviews are considered. A negative review has a score ≤ 4 out of 10, and a positive review has a score ≥ 7 out of 10. No more than 30 reviews are included per movie.

The datasets have been divided into training, validation and test sets for this study, and have been described in Table I.

TABLE I
Data size for Training, Validation and Test Sets

Data Set	Training Set	Validation Set	Test Set
AG News	96000	24000	7600
Amazon Review	107262	26816	33520
IMDB Review	20000	5000	25000

III. Data Pre-processing

The pre-processing part of an experiment involves conversion of the raw data to a data-mining-ready structure. For doing this, we used various natural-language based based methods to remove stop-words, special characters, numbers and converted most frequently used short hand texts to full length texts (i.e. I'll to I will, 'hv to have). By doing this, we have cleaned the texts such that the models get a better understanding of the sentence structures. This cleaned data was sent to the GloVe word embedding IV for getting the vector representation for each word, thus getting a multi-dimensional array for each sentence and each review.

IV. Word Embedding

We use the pre-trained GloVe model for word embeddings. The GloVe model was trained on the non-zero entries of a global word-word co-occurrence matrix, which tabulates how frequently words co-occur with one another in a given corpus. It was trained on a dataset of six billion tokens (words) with a

vocabulary of 400 thousand words. The GloVe has embedding vector sizes of 50, 100, 200 and 300 dimensions. In this project, we used embedding vectors of 100 dimensions.

V. Model Architecture

We explore the following three popular neural network architectures.

A. Convolutional Neural Networks:

We designed a sequential model comprising of an embedding layer where words are mapped to embedding vectors in a 100 dimensional space followed by two 1D convolutional layers with filter sizes of {128, 128} and kernel sizes of {5, 5}. The output of each convolutional layer is passed through a max-pooling layer with a pool size of 5 followed by a spatial dropout of 0.5. These sequences of layers are then followed by two-fully connected layers - the first dense layer vectorizes the flattened output into a 128-dimensional feature vector; and the last dense layer comprises of neurons equal to the number of classes. The loss is computed using categorical cross entropy loss for multi-class classification task and binary cross entropy loss for binary classification. Similarly, the activation for the output will be either softmax or sigmoid given multi-class and binary classification tasks respectively. The network uses rectified linear units (ReLU) as the activation function for each of the convolutional layers and Adam optimizer with a learning rate of 0.001 is used as the optimizer. Figure 1 summarizes our network architecture.

B. Long Short-Term Memory:

We designed a sequential model comprising of an embedding layer where words are mapped to embedding vectors in a 100 dimensional space followed by two LSTM layers of 100 units each. Each LSTM layer is followed by a dropout layer with dropout rate of 0.5 in order to avoid overfitting and to train a model that generalizes well. These sequences of layers are then followed by a fully connected layer which comprises of neurons equal to the number of classes. The loss is computed using categorical cross entropy loss for multi-class classification task and binary cross entropy loss for binary classification. Similarly the activation for the output will be either softmax or sigmoid for multi-class and binary classification tasks respectively. The network uses *tanh* activation for each of the LSTM layers and Adam optimizer with a learning rate of 0.001. Figure 2 summarizes our network architecture.

In order to build networks capable of capturing context from information presented before and after the occurrence of the word, for making accurate classifications of categories or sentiments, we also experiment with Bidirectional-LSTM layers to aggregate contextual information. Our motivation behind trying both these variant stems came from our understanding of the way humans interpret natural language. The architecture is essentially the same with the only difference being that the normal LSTM layers are now replaced with Bidirectional-LSTM layers. They might have implications of increased training time and probably improved accuracy. We try to analyze this in our experiments to evaluate the pros and cons of both architecture variants. Figure 3 summarizes the bidirectional-LSTM network architecture.

C. Hierarchical Attention Networks:

Our HAN model exhibits a hierarchical structure very similar to that of documents, wherein two levels of attention mechanisms (word-level and sentence-level) are applied, which enables it to identify the key representative and dictating features for document classification. It consists of a word sequence encoder, a word-level attention layer, a sentence encoder and a sentence-level attention layer. First, we use an embedding layer which maps the words into a 100-dimensional embedding space. These embedded vectors are then fed to a bidirectional GRU Layer (word sequence encoder) which comprises of 100 units. In this layer, the words are annotated to incorporate contextual information. Then we add an attention layer on top of this to extract only those words that are important to the meaning of the sentence and we aggregate the representation of those informative words to form a sentence vector. Now, a similar set of layers is added to extract document vectors. Here a bidirectional GRU Layer with 100 units serves as the sentence encoder and an attention layer is added on top of this to extract only those sentences that are meaningful representatives of the context, sentiment conveyed or the category of the document. This document vector encompasses key information that helps in document classification. The output of the last attention layer is fed into a fully connected layer which flattens the output into a class-dimensional vector with probabilities indicating the likelihood of the document belonging to a particular class. The loss is computed using categorical cross entropy loss for multi-class classification task and binary cross entropy loss for binary classification. Similarly, the activation for the output will be either softmax or sigmoid for multi-class and binary classification tasks. The network uses *tanh* activation for each of the Bidirectional GRU layers and Adam optimizer with a learning rate of 0.001. Figures 4 and 5 summarizes our network architecture.

VI. Model Training and Hyperparameter Selection

A. *Convolutional Neural Networks*

In order to improve the classification accuracy of CNNs, certain design choices were made. For reducing the complexity of the networks (number of trainable parameters) and to avoid over-fitting, we use pooling layers to downsample the spatial dimensions. Additionally, a spatial dropout with dropout rate of 0.5 is used to promote the independence between feature maps as in our case the adjacent frames within feature map are strongly correlated and hence a regular dropout was not so effective.

For training the model, we have used early stopping with a patience of 10 epochs. If the validation loss does not improve for 10 consecutive epochs, we halt the training. From Figures

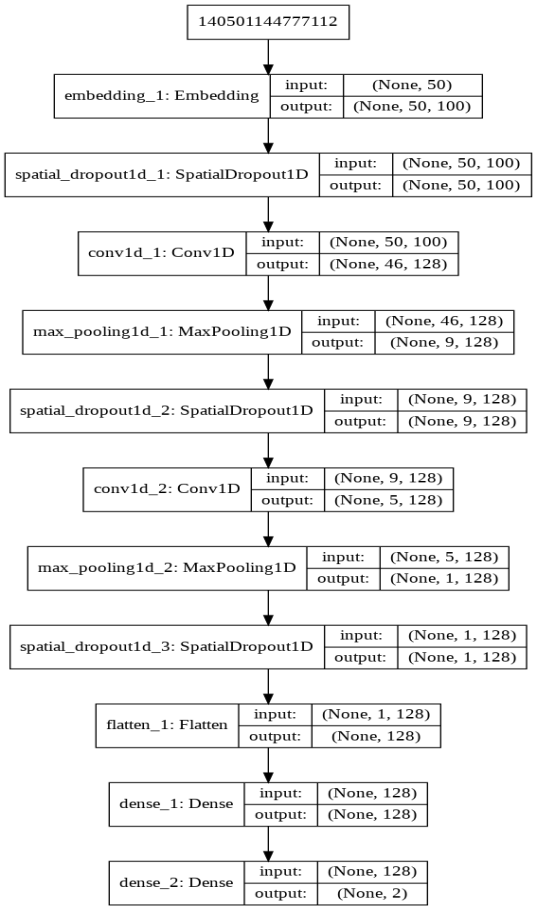

Fig. 1. CNN Architecture (for IMDB)

10 and 11, we see that the training converges after 20 epochs and post this, the model does not essentially learn anything and only overfits. We noticed similar numbers while training the model on different datasets and have used early stopping as a criterion to decide exactly when to stop training. We also experimented with a the number of Conv1d layers and 2 layers seemed optimum. Fewer layers were underfitting and more layers did not provide substantial benefits either. Other hyperparameters we tried experimenting with include, the learning rate, optimizer function, batch size and the number of epochs. In our experiments, we observed that the model was taking a lot of time to converge and the convergence in many runs were sub-optimal when working with stochastic gradient descent. Adam optimizer with a learning rate of 0.001 provided the best results. We also carefully investigated the optimal batch size so that the training is faster and the model does not make many frequent updates. Following this, a batch size of 64 provided a good performance. Table II provides a comprehensive list of the range of values tried for different hyperparameters along with the final values that were fixed.

B. Long Short-Term Memory

In order to access the performance of Long Short-Term memory (LSTM) networks for text classification, we exper-

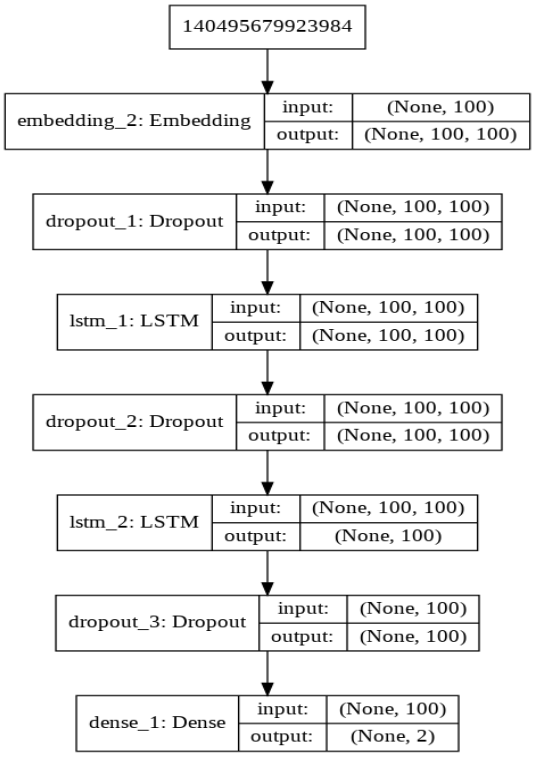

Fig. 2. LSTM Architecture (for IMDB)

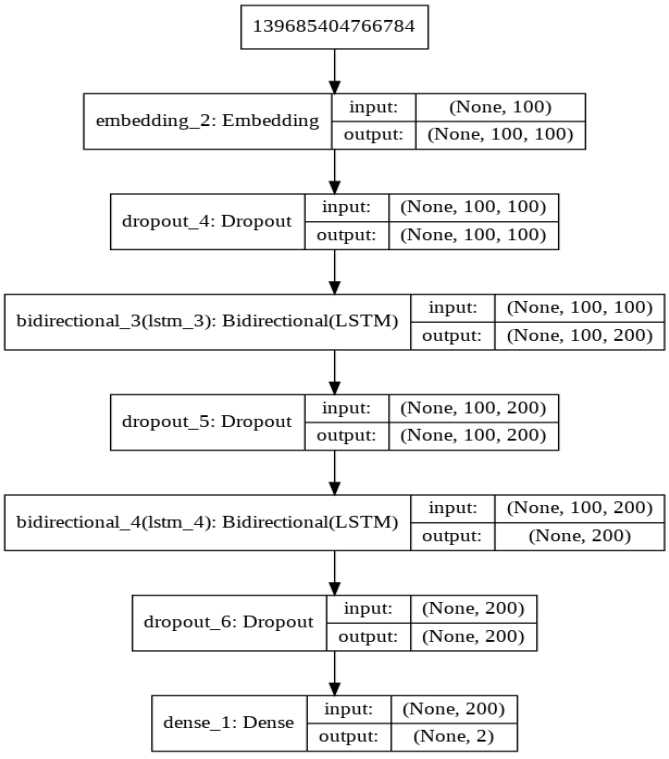

Fig. 3. BiDirectional LSTM Architecture (for IMDB)

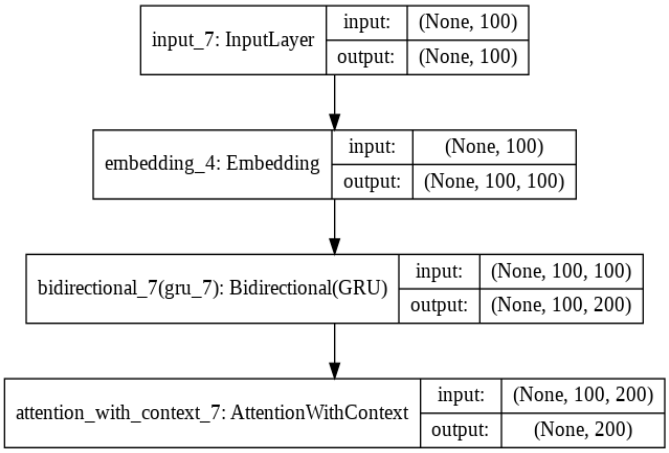

Fig. 4. HAN Architecture [Word Encoder + Attention] (for IMDB)

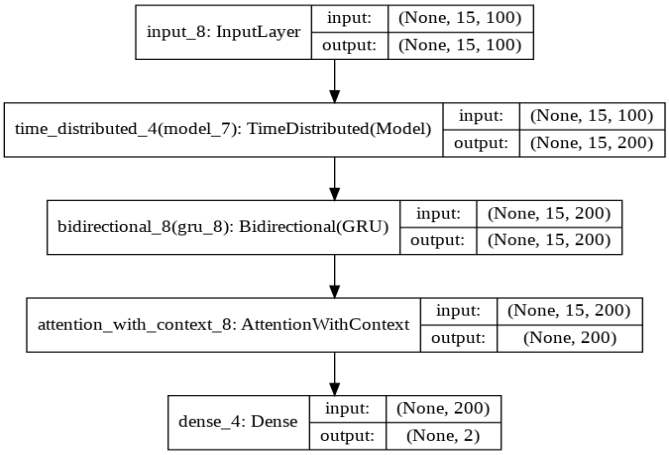

Fig. 5. HAN Architecture [Sentence Encoder + Attention] (for IMDB)

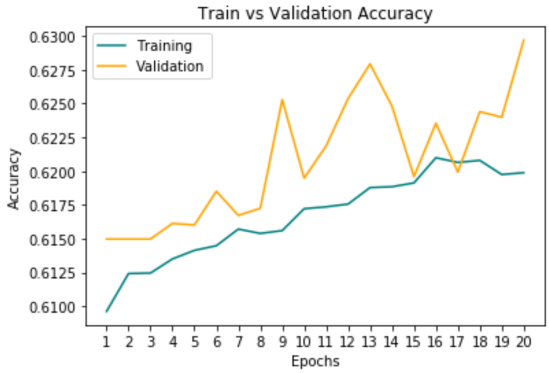

Fig. 6. Training and Validation Accuracy (CNN - Amazon)

imented with both unidirectional and bidirectional LSTM networks. As per the numbers in Tables III and 24, we note that in this particular text classification task, there is no substantial difference in the accuracy values from using either of the models. Further, there is additional training time required for bi-LSTM which is roughly more than double of

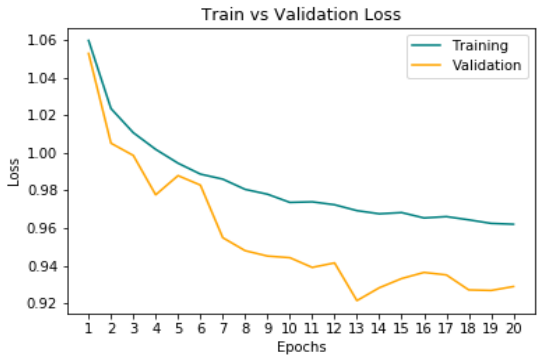

Fig. 7. Training and Validation Loss (CNN - Amazon)

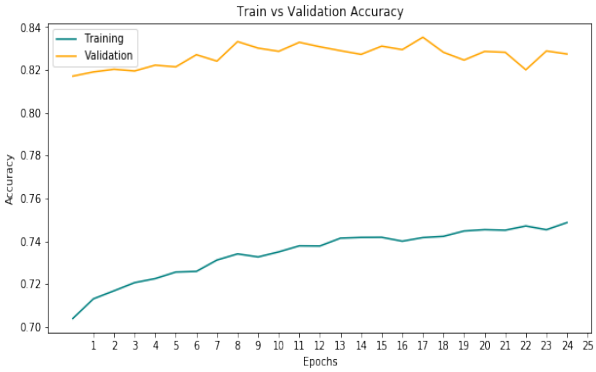

Fig. 8. Training and Validation Accuracy (CNN - AG News)

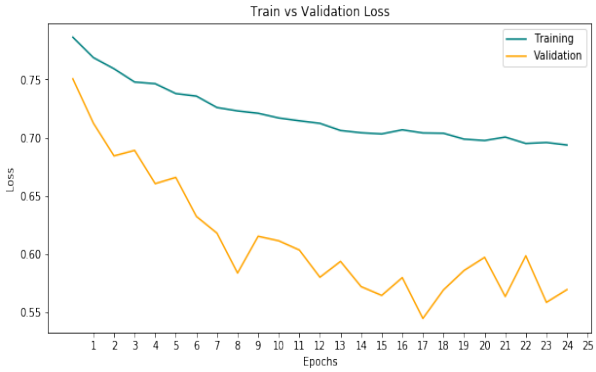

Fig. 9. Training and Validation Loss (CNN - AG News)

unidirectional LSTM. Thus, we see that both bi-directional and unidirectional LSTMs perform comparably. One possible reason for this observation could be the nature of task at hand. While working with reviews, we need to understand that these are reviews given by users and are usually short and crisp. In such short sentences, we believe that there is not so much of a need to worry about the context of words presenting themselves in the future. Moreover, we ensure that our network, the input sequence length and the dimensions we work with are sufficient to aggregate the context information by using a simpler model, thereby eliminating the need for bi-

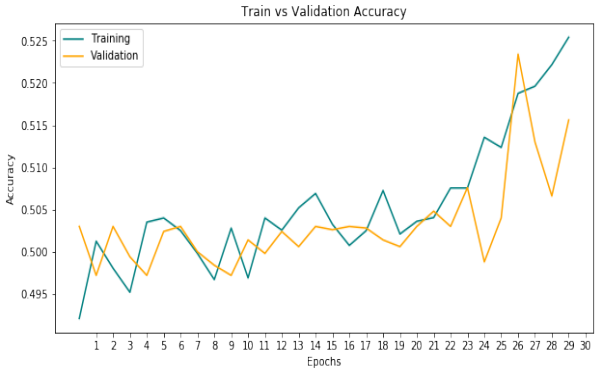

Fig. 10. Training and Validation Accuracy (CNN - IMDB)

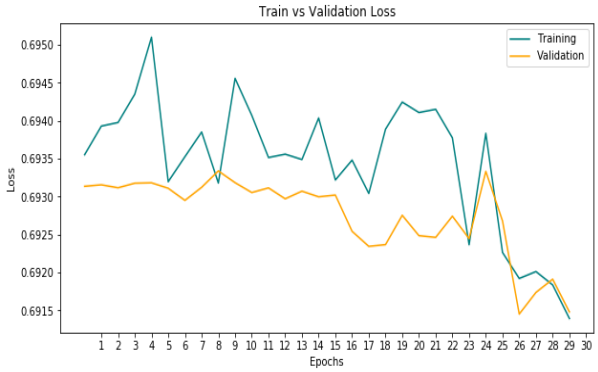

Fig. 11. Training and Validation Loss (CNN - IMDB)

LSTM networks. For training the model, we have used early

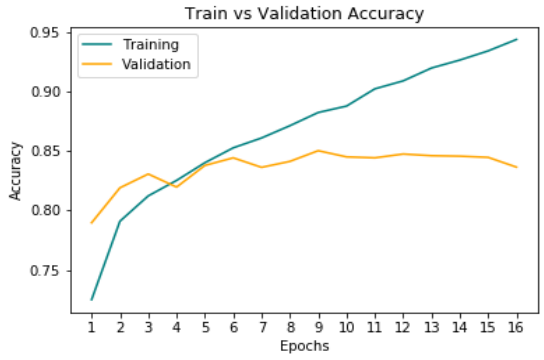

Fig. 12. Training and Validation Accuracy (LSTM - IMDB)

stopping with a patience of 10 epochs. From Figure 12 and 13, we see that the training converges after around 7 epochs and post this, the model does not essentially learn anything and only overfits. We noticed similar numbers while training the model on different datasets and have used early stopping as a criterion to decide exactly when to stop training. We also experimented with the number of LSTM layers and 2 layers seemed optimum. Fewer layers were underfitting and more layers did not offer substantial improvements for the increased training time. Other hyperparameters we tried experimenting with include, the learning rate, optimizer function, batch size and the number of epochs. With higher learning rates, the validation accuracy and losses over the epochs were pretty bumpy, indicating steep jumps across the local minima resulting in delayed and sometimes sub-optimal convergence. Adam optimizer with a learning rate of 0.001 seemed to work best in our case. As for the number of epochs, we trained our model for 50 epochs to note the trend of validation accuracy and losses and observed that beyond some 7-12 epochs (across different datasets), the model begins to over-fit and hence does not generalize well. We also carefully investigated the optimal batch size and fixed on a batch size of 64 that seemed to work well in our case. Table II provides a comprehensive list of the range of values tried for different hyperparameters along with the final values that were fixed.

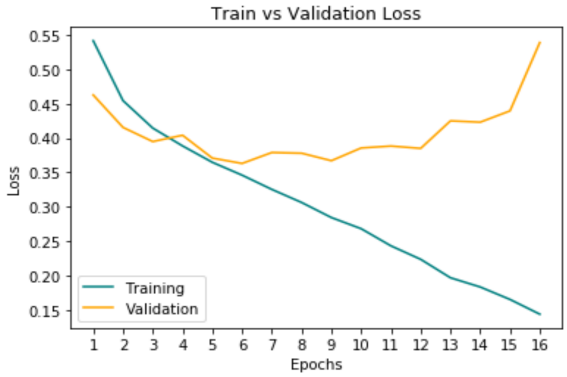

Fig. 13. Training and Validation Loss (LSTM - IMDB)

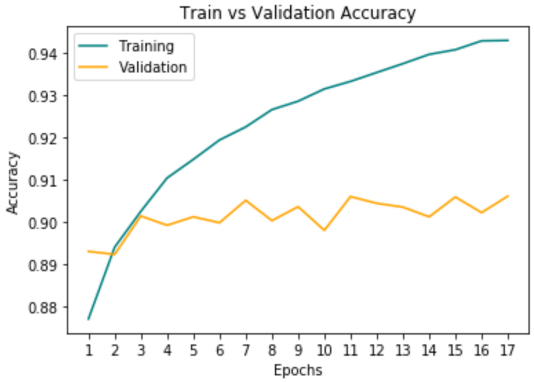

Fig. 14. Training and Validation Accuracy (LSTM - Ag-News)

C. *Hierarchical Attention Networks*

Just like the previous networks, for training the Hierarchical Attention Networks (HAN) model, we have used early stopping with a patience of 10 epochs. However, a quick look at Figure 18 shows that the training on IMDB dataset converges in around 2-3 epochs, post which the validation accuracy is pretty constant, indicating that the model is not learning anything new and is possibly overfitting. The reason behind

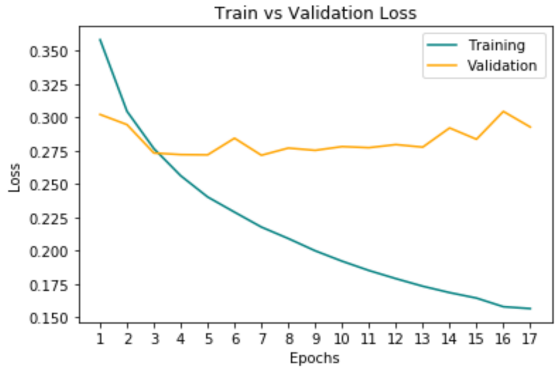

Fig. 15. Training and Validation Loss (LSTM - Ag-News)

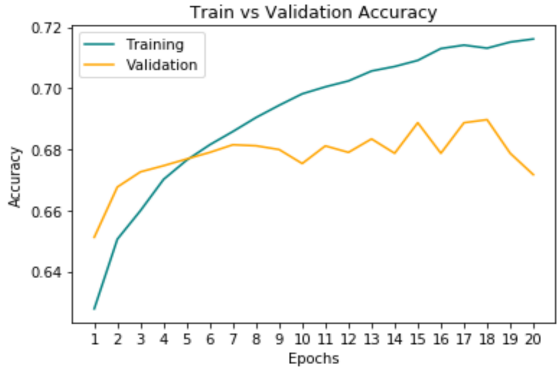

Fig. 16. Training and Validation Accuracy (LSTM - Amazon)

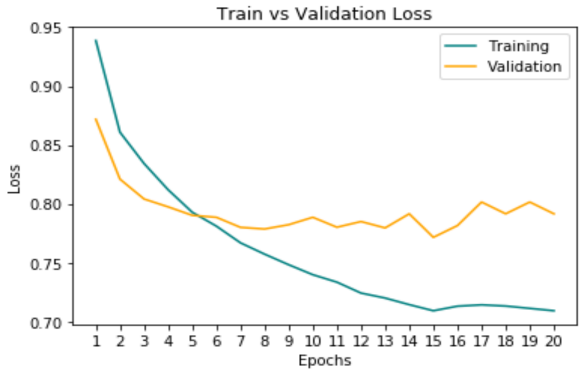

Fig. 17. Training and Validation Loss (LSTM - Amazon)

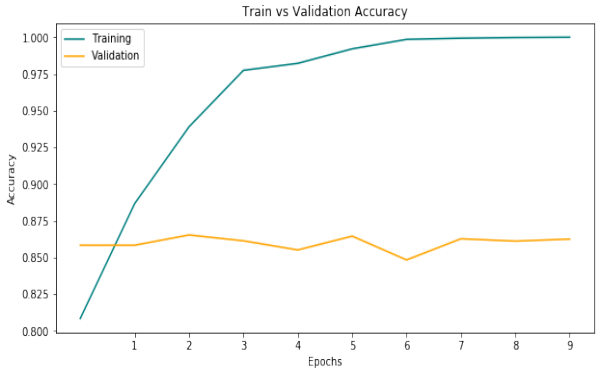

Fig. 18. Training and Validation Accuracy (HAN - IMDB)

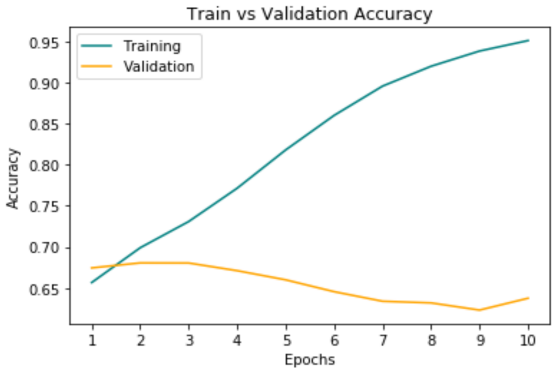

Fig. 19. Training and Validation Accuracy (HAN - Amazon)

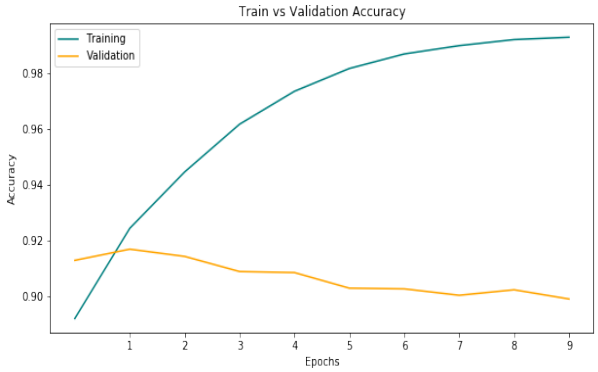

Fig. 20. Training and Validation Accuracy (HAN - Ag-News)

this could be the sheer complexity of the HAN model and relatively less training data. In order to understand this better, we trained our model on a subset of Amazon review dataset which is relatively larger than IMDB and Ag-News. From Figures 19 and 23 we observe that the validation loss starts increasing steeply beyond 7 epochs indicating over-fitting and so we stop our training at 7 epochs. Based on our observations, we conclude that for models as relatively complex as HAN, we require large amounts of training data and fairly extensive computational resources to conduct efficient training for accurate classifications.

We observed that HAN models in general took around 1 hour per epoch (wall clock time) when trained on NCSU ARC HPC cluster. Training on smaller dataset, while comparatively faster, did not give us much scope for hyperparameter tuning as the model starts overfitting very soon. Hence, in order to establish a middle ground and to select the best set of parameters for training, we performed a literature survey and used configurations that were reported to perform well for HAN models. Our model was trained for 20 epochs (with early stopping) using Adam optimizer with a learning rate of 0.001 and input sequences were fed to the model in batches

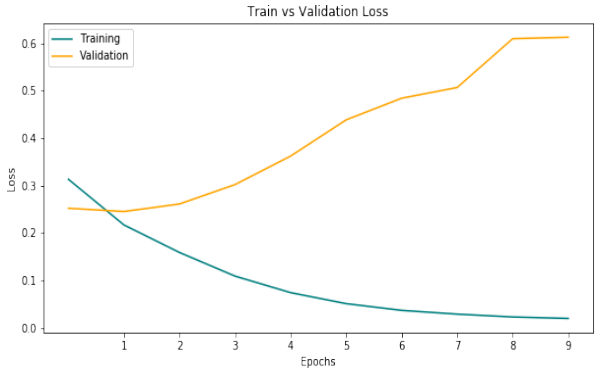

Fig. 21. Training and Validation Loss (HAN - Ag-News)

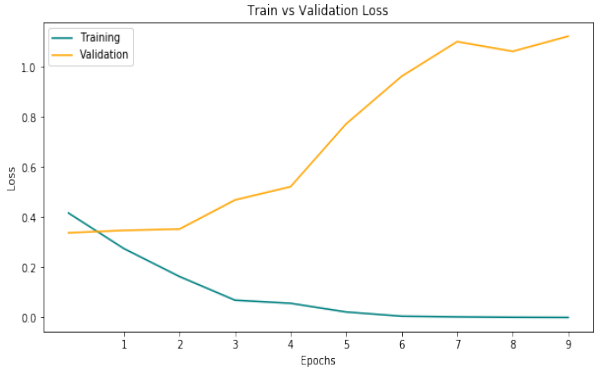

Fig. 22. Training and Validation Loss (HAN - IMDB)

of 64.

Additionally, for all the above deep learning architectures, we tried to set the embedding layer as trainable in order to observe the impact on training and corresponding accuracy. This would consequently increase the number of trainable parameters and subsequently the model training time. We did not observe any significant improvement in the accuracy scores to allow such increased training time. Hence we decided to not incorporate this approach.

VII. Results and Analysis

Table III shows the accuracy scores of our trained models for different datasets and Figure 24 shows the average model training time. We observe a clear pattern indicating that HANs are superior to LSTMs and LSTMs are superior to CNNs, when tasked with text classification. Such progressive increase in the accuracy values could be directly attributed to the nature of the models and we also note that these improvements are regardless of both the size of the dataset and the nature of the task (binary or multi-class classification). While LSTMs have been popularly used for such tasks owing to their effectiveness in modeling sequential data, we observe that attention mechanism together with exploration of hierarchical document structure clearly provides an improvement in the accuracy scores.

TABLE II
Hyper parameter Tuning Ranges

Hyperparameters	***Range***	***Final CNN /LSTM /HAN***
Learning Rate	0.1, 0.01, 0.001, 0.0001	0.001/0.001/0.001
Optimizer	RMSProp, SGD, Adam	Adam /Adam /Adam
Epochs	10, 20, 30, 50	20 /20 /20
Batch Size	16, 32, 64	64 /64 /64

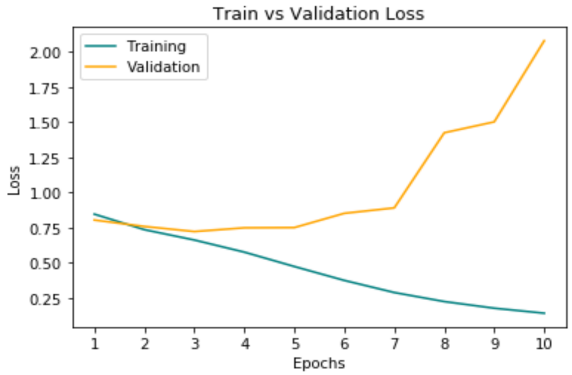

Fig. 23. Training and Validation Loss (HAN - Amazon)

One possible reason for the poor accuracy results of CNN based models could be that even though their convolutional and pooling layers help find strong local clues regarding class memberships, it fails to capture information that might be spread across the length of the document. In such cases, they often lose the local order of words and are incapable of setting context for the document.

TABLE III
Evaluation Results

	Ag-News	***IMDB***	***Amazon***
CNN	0.838	0.517	0.630
LSTM	0.887	0.849	0.672
Bi-LSTM	0.884	0.815	0.669
HAN	0.913	0.856	0.710

This drawback is overcome by LSTM networks because of their effectiveness in modeling sequential data and their cell states make it possible to capture and remember long-term dependencies (context information). We observe that they perform decently well on all our datasets with good accuracy scores. However, LSTMs do not try to identify important words that need to be attended to, for making classifications. Giving attention to the important words and using them to make classification decisions can definitely help the model perform better as the model will have a definitive latent representation to look for while making such decisions.

Thus, by employing both word-level and sentence-level attention, together with exploration of hierarchical document structure, HAN models provide the best results in this task and it is evident from the figures in Table III.

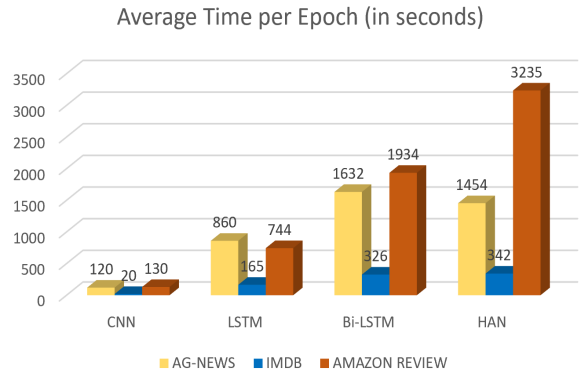

Fig. 24. Average Time Per Epoch

VIII. Conclusion

Our experiments and observations clearly indicate the effectiveness of the global word and sentence importance vectors used in HAN. By exploring the hierarchical document structure, and employing word-level and sentence-level attentions, we were able to achieve superior performance for text classification tasks across all datasets of varying sizes and classes (binary and multi-class). Due to limited computational resources, we were not able to complete our experiments with different genetic algorithms to identify optimal hyperparameters. As a future work, we would like to explore different genetic algorithms for more fine-grained hyperparameter tuning and would also like to evaluate the performance of BERT [7] which is the current state-of-the-art for text classification.

References

[1] Y. Kim, "Convolutional neural networks for sentence classification," *CoRR*, vol. abs/1408.5882, 2014.

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

[3] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1480–1489, 2016.

[4] X. Zhang, J. J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," *CoRR*, vol. abs/1509.01626, 2015.

[5] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel, "Image-based recommendations on styles and substitutes," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 43–52, ACM, 2015.

[6] 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-volume 1*, pp. 142–150, Association for Computational Linguistics, 2011.

[7] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018.