TEXT CLASSIFICATION: COMPARATIVE ANALYSIS OF DIFFERENT DEEP NEURAL NETWORK ARCHITECTURES

Shawn Varghese Rajan (ID: 200220854)
Jibin Thomas (ID: 200224070)

1. Motivation

Automated text classification is considered a vital method to manage and process a vast number of documents in digital forms that are widespread and continuously increasing. In general, text classification plays an important role in information extraction and summarization, text retrieval, and question-answering. Within text classification, sentiment analysis has grown to be one of the most active research areas in Natural Language Processing. This proliferation is due to the fact that opinions are central to almost all human activities and are key influencers of our behaviors. Sentiment Analysis is the process of computationally identifying and categorizing opinions expressed in a piece of text, especially to identify the writer's attitude towards a particular topic or product.

Deep learning has emerged as a powerful machine learning technique for sentiment analysis as it is able to learn the intricate patterns in text data and produce state of the art prediction results.

The comparison of the three major DNN architectures will help us identify the specific use-cases and situations where each architecture outperforms the others.

2. Problem Statement

In this project, we are expected to develop and compare three different deep neural network architectures for the task of text classification. The architectures selected for comparison are Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) and Hierarchical Attention Network (HAN). We are also expected to use word vectors generated by GloVe as an underlying data model

3. Proposed Solution

A classification technique (or classifier) is a systematic approach to building classification models from an input dataset. Each technique employs a learning algorithm to identify a model that best fits the relationship between the attribute set and class label of the input data.

Our proposed solution consists of the following key steps

1. Dataset preparation and preprocessing
2. Feature Engineering
3. Model building
4. Hyperparameter tuning

Dataset Preparation and preprocessing:

In this project, as we are doing sentiment analysis, we have decided to classify reviews based on their sentiment. The two datasets which we have used for this purpose are IMDB movie reviews and Yelp restaurant reviews.

The following data preprocessing steps were performed on each dataset

1. Remove punctuations
2. Split reviews into sentences and tokens(words)
3. Identifying and removing stopwords
4. Setting up parameters for feature engineering

Punctuations were removed from the raw text data using regular expressions. The cleaned string was split into sentences and tokens using the tokenizer module from the NLTK library. Instead of using a standard stopwords list (e.g. stopwords list from NLTK library), we created a custom stopwords list to ensure that words which are most common to that problem are removed from the corpus. The maximum words in the vocabulary and maximum number of words in a review were set at 20,000 and 1,000 respectively. Reviews where the number of words were less than 1,000 were padded to ensure consistent word length across reviews.

Feature Engineering:

In order to train a neural network model, we need to convert the processed text data and words into a numerical representation. Though different methods (e.g. bag of words, tf-idf) exist to achieve this, word embeddings are one of the most popular methods where each word is represented with a real-valued vector. These vectors are generated such that they capture the semantic and syntactic similarity, relation with other words, etc. In this project, we utilize the already existing 100-dimensional GloVe word embeddings.

Model building:

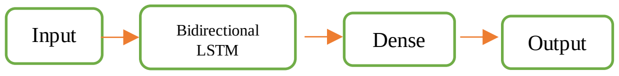
Convolutional Neural Network (CNN) – The pictorial form of the architecture is given in Fig 1.

CNNs are considered good at extracting local and position-invariant features and therefore should ideally perform well on Sentiment Classification tasks.

The key disadvantage of CNN is in classifying reviews where the sentiment is determined by the entire sentence or a long-range semantic dependency rather than some local key-phrases. In such cases, an architecture like RNN is needed that handles long sequences correctly. The RNN encodes the entire word sequence of the long review, making it hard for single negative key phrases to play a significant role in the final representation.

Recurrent Neural Network - A recurrent neural network (RNN) is an extension of a conventional feedforward neural network, which is able to handle a variable-length sequence input. The RNN handles the variable-length sequence by having a recurrent hidden state whose activation at each time is dependent on that of the previous time.

In our RNN implementation, we used LSTM (Long Short-term Memory), as traditional RNNs are difficult to train to capture long term dependencies due to the vanishing gradient issue. The LSTM architecture for sentiment analysis is as follows:



The pictorial form of the architecture is given in Fig 2.

Hierarchical Attention Network - HAN has two distinctive characteristics: (i) it has a hierarchical structure that mirrors the hierarchical structure of documents; (ii) it has two levels of attention mechanisms applied at the word and sentence-level, enabling it to attend differentially to more and less important content when constructing the document representation. This is ideal for applications like sentiment analysis where few words or sentences usually determine the overall sentiment of the review. Our HAN has been implemented based on Yang et al. There are two attention layers, at word level and sentence level, to reward words and sentences that are clues to correctly classify a review. The pictorial form of the architecture is given in Fig 3.

Hyperparameter tuning:

Hyperparameter optimization is a big part of deep learning as neural networks are notoriously difficult to configure and there are a lot of parameters that need to be set. We used the grid search capability from the scikit-learn library to tune the hyperparameters of Keras deep learning models.

Hyperparameter tuning was performed on the CNN model on IMDB dataset. Number of filters (filters = 8, 16, 32) and number of neurons (neurons = 8, 16, 32) in the dense layer are the parameters which were tuned using the grid search method. Based on the grid search method, the optimal values for the filters and neurons were 32 and 32 respectively.

4. Results

The neural network models were run on test data to understand their performance on previously unseen data. In both the datasets, HAN closely followed by RNN emerged as the best models based on test data classification accuracy.

Dataset	**CNN**	**RNN**	**HAN**
IMDB	81.7%	86.2%	87.2%
Yelp	88.8%	92.8%	93.2%

Table 1: Test data classification accuracy

For IMDB review dataset, the HAN model outperformed the RNN and CNN by 1% and 5.5% respectively. For Yelp reviews, the HAN model outperformed the RNN and CNN by 0.4% and 4.4% respectively. The accuracy and loss function graphs for each model are provided from Fig 7 – Fig 18. The relatively poor performance of CNN may be attributed to its inability in classifying reviews where the sentiment is determined by the entire sentence or a long-range semantic dependency rather than some local key-phrases. This is not an issue in RNN and HAN where the sequence is taken into consideration resulting in lesser chances of classifying a review based on single key phrases. Based on the comparative study, we can conclude that RNN and HAN suits the task of sentiment analysis better than CNN.

Dataset	#Classes	#Reviews	Avg #sentences	Max #sentences	Avg #words	Max #words	Vocabulary
IMDB	2	50,000	12	120	230	2,469	121,364
Yelp	2	194,540	9	108	128	1,005	175,067

Table 2: Data Statistics for the IMDB reviews and Yelp reviews dataset

5. Description of the datasets

We used two datasets to compare the performance of the three DNN architectures: IMDB reviews and Yelp reviews

IMDB reviews: This dataset contains movie reviews along with their associated binary sentiment polarity labels. The core dataset contains 50,000 reviews split evenly into 25k train and 25k test sets. The overall distribution of labels is balanced (25k pos and 25k neg). Special processing steps were taken while initially preparing the dataset to ensure it can serve as a benchmark for sentiment classification. In the labeled train/test sets, a negative review has a score <= 4 out of 10, and a positive review has a score >= 7 out of 10. Thus, reviews with more neutral ratings are not included in the train/test sets.

Yelp reviews: This dataset contains restaurant reviews by customers and their ratings. The ratings initially ranged from 1 to 5. Similar to the process followed in the creation of the IMDB dataset, a negative review has a score of <=2 out of 5 and a positive review has a score >=4 out of 5. Thus, reviews with ratings of 3 (neutral) are not included in the train/test sets. Almost 80% of the yelp restaurant reviews fell into the positive category (original rating >=4). This led to class imbalance and poor performance of the neural network models on data. In order to combat this, the class weights were adjusted while fitting the model. The loss generated from negative category were given 4 times more importance than the positive category to ensure that the models are able to unearth the intricate patterns in the data.

Table 2 provides basic data statistics for the two datasets

The datasets were split into training, validation and test in the ratio of 64%:16%:20%. Validation data was used to identify the optimal hyperparameters for the model.

Below table provides the number of reviews in the training, validation and test for both the datasets

Dataset	Training	Validation	Test
IMDB	20,000	5,000	25,000
Yelp	124,533	31,133	38,874

Table 3: # reviews in training, validation and test data

6. Architecture

The CNN architecture had the input data being fed into a Conv1D layer with 32 filters and kernel size equal to 4. The output of the Conv 1D layer was further fed into a Dropout layer (p= 0.2) to prevent overfitting of the training data. The output of the dropout layer was fed into a Maxpooling layer to reduce the dimensionality of the input representation. The output of the Maxpooling layer was flattened prior to sending it into a fully connected (Dense) layer of 32 neurons and ReLu activation. The output of the Dense layer is fed into a single neuron dense layer with sigmoid activation. The pictorial representation of the architecture is given in Fig 1.

For our RNN implementation, we use bidirectional LSTMs. Unidirectional LSTMs only preserves information of the past because the only inputs it has seen are from the past. Using bidirectional will run the inputs in two ways, one from past to future and one from future to past. This enables the bidirectional LSTMs to preserve information from the past and future and hence provide better contextual information compared to unidirectional LSTMs. The pictorial representation of the architecture is given in Fig 2.

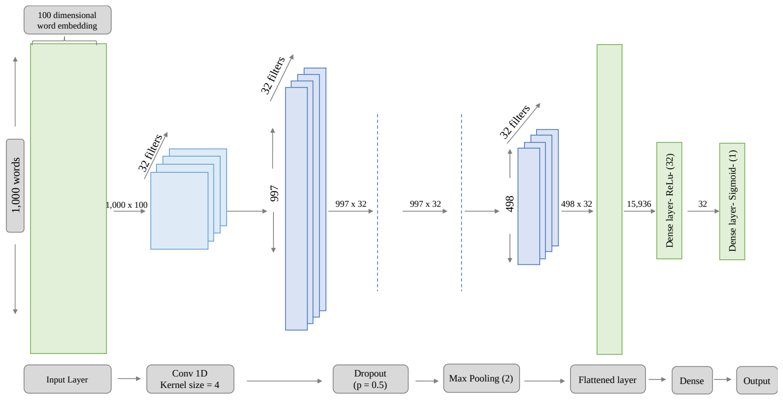


Fig 1. Convolutional Neural Network architecture

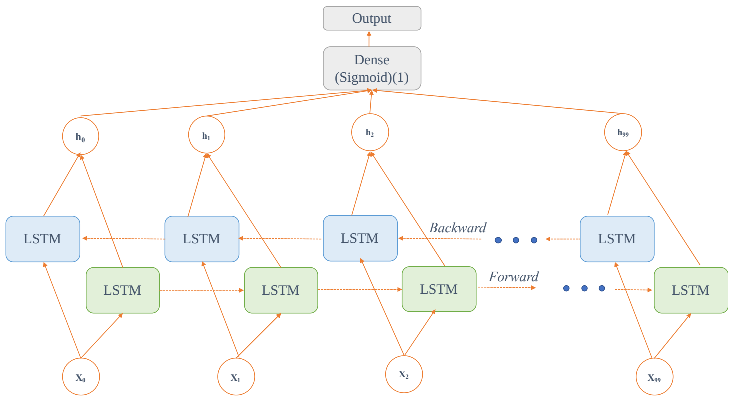


Fig 2. Recurrent Neural Network architecture

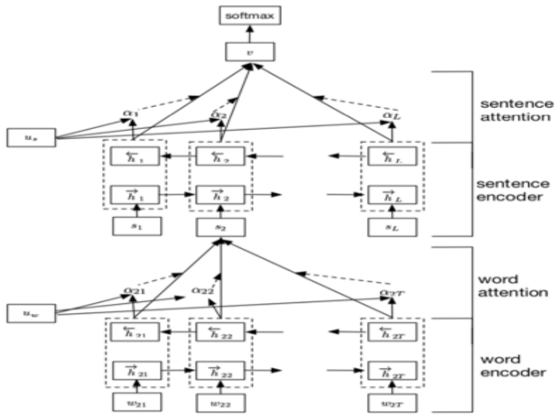
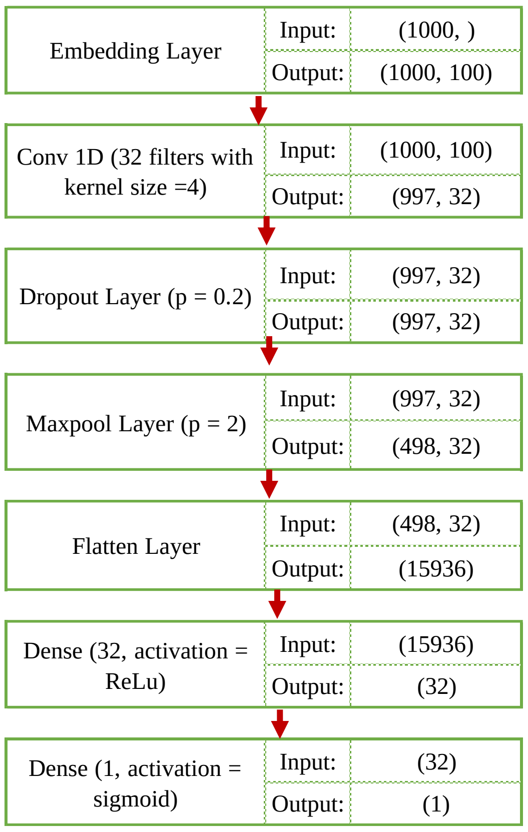


Fig 3. Hierarchical Attention Network architecture

For our HAN implementation, we referenced the architecture provided in the paper 'Hierarchical Attention Networks for Document Classification' by Yang et.al. The architecture captures the hierarchical structure of documents and it has two levels of attention mechanisms applied at the word and sentence-level, enabling it to attend differentially to more and less important content when constructing the document representation. The pictorial representation of the architecture is given in Fig 3.

7. Architecture Hyperparameter table

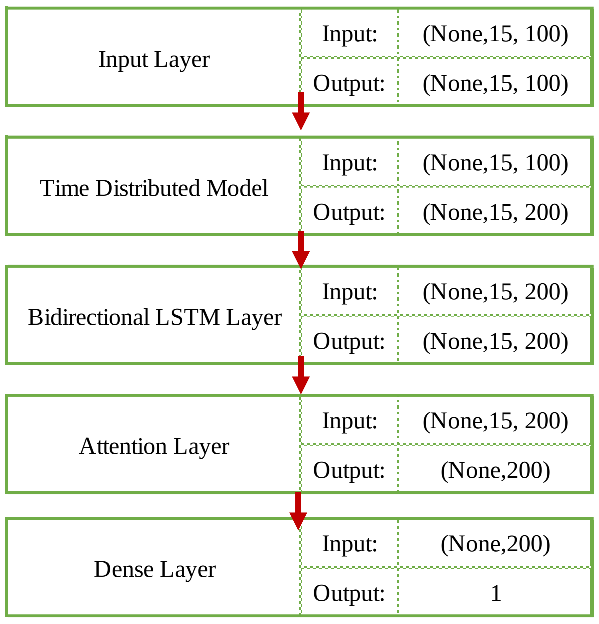
Convolutional Neural Network



Layer		
Embedding Layer	Input:	(1000,)
	Output:	(1000, 100)
Conv 1D (32 filters with kernel size =4)	Input:	(1000, 100)
	Output:	(997, 32)
Dropout Layer (p = 0.2)	Input:	(997, 32)
	Output:	(997, 32)
Maxpool Layer (p = 2)	Input:	(997, 32)
	Output:	(498, 32)
Flatten Layer	Input:	(498, 32)
	Output:	(15936)
Dense (32, activation = ReLu)	Input:	(15936)
	Output:	(32)
Dense (1, activation = sigmoid)	Input:	(32)
	Output:	(1)

Fig 4. CNN Hyperparameters

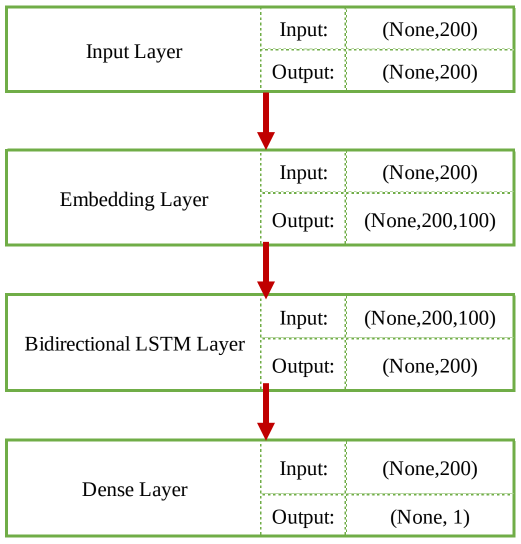
Hierarchical Attention Network



Layer		
Input Layer	Input:	(None,15, 100)
	Output:	(None,15, 100)
Time Distributed Model	Input:	(None,15, 100)
	Output:	(None,15, 200)
Bidirectional LSTM Layer	Input:	(None,15, 200)
	Output:	(None,15, 200)
Attention Layer	Input:	(None,15, 200)
	Output:	(None,200)
Dense Layer	Input:	(None,200)
	Output:	1

Fig 6. HAN Hyperparameters

Recurrent Neural Network



Layer		
Input Layer	Input:	(None,200)
	Output:	(None,200)
Embedding Layer	Input:	(None,200)
	Output:	(None,200,100)
Bidirectional LSTM Layer	Input:	(None,200,100)
	Output:	(None,200)
Dense Layer	Input:	(None,200)
	Output:	(None, 1)

Fig 5. RNN Hyperparameters

8. Training and Validation Accuracy over Epochs

IMDB reviews dataset

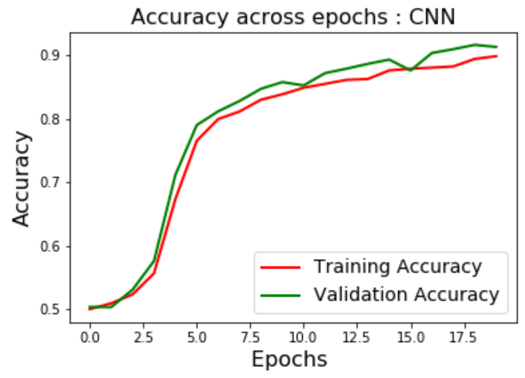


Fig 7. CNN on IMDB reviews: Accuracy over Epochs

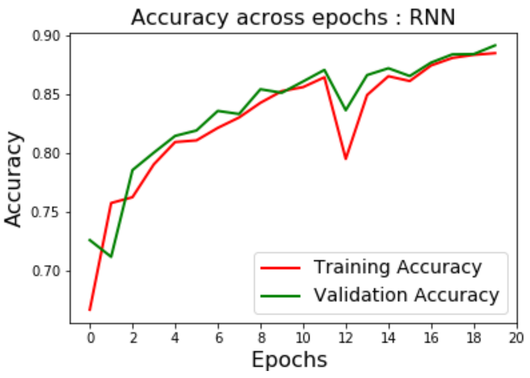


Fig 8. RNN on IMDB reviews: Accuracy over Epochs

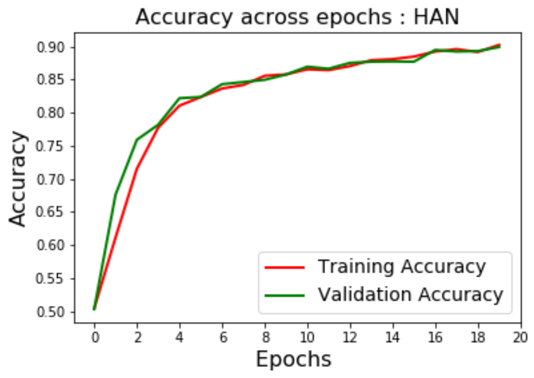


Fig 9. HAN on IMDB reviews: Accuracy over Epochs

Yelp Reviews dataset

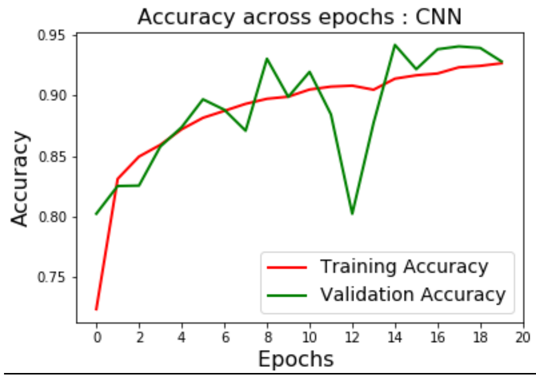


Fig 10. CNN on Yelp reviews: Accuracy over Epochs

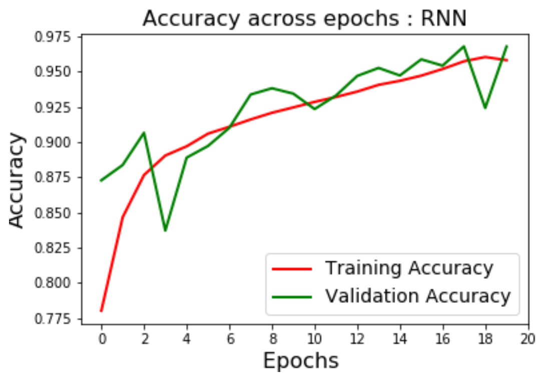


Fig 11. RNN on Yelp reviews: Accuracy over Epochs

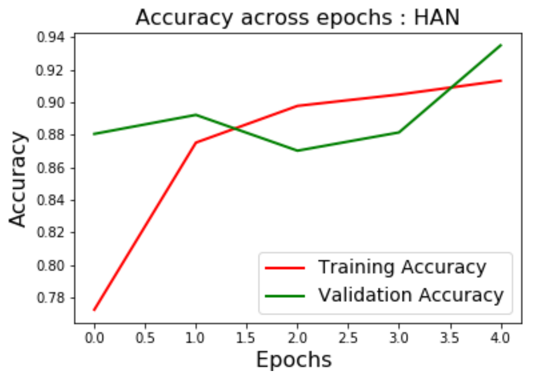


Fig 12. HAN on Yelp reviews: Accuracy over Epochs

9. Training and Validation Loss over Epochs

IMDB reviews dataset

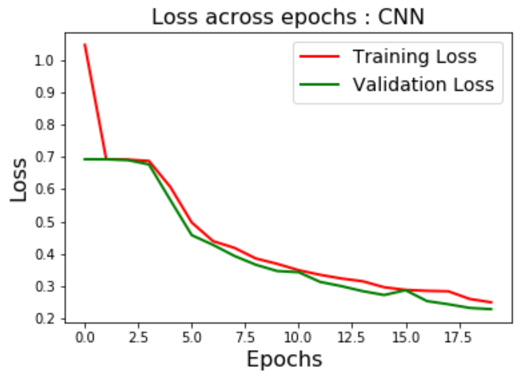


Fig 13. CNN on IMDB reviews: Loss over Epochs

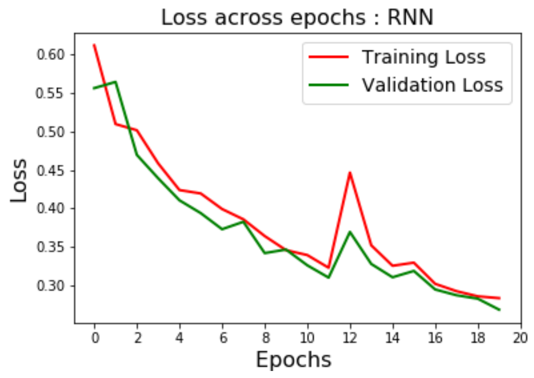


Fig 14. RNN on IMDB reviews: Loss over Epochs

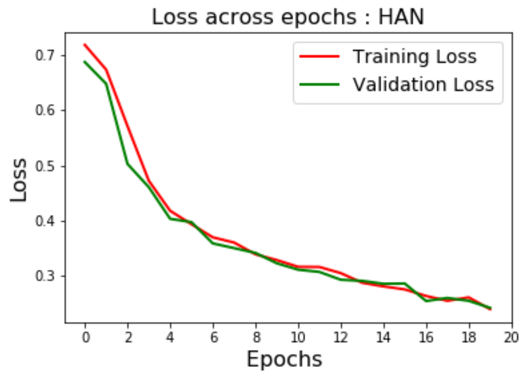


Fig 15. HAN on IMDB reviews: Loss over Epochs

Yelp Reviews dataset

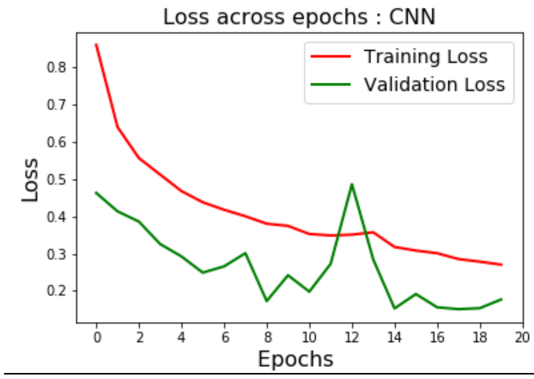


Fig 16. CNN on Yelp reviews: Loss over Epochs

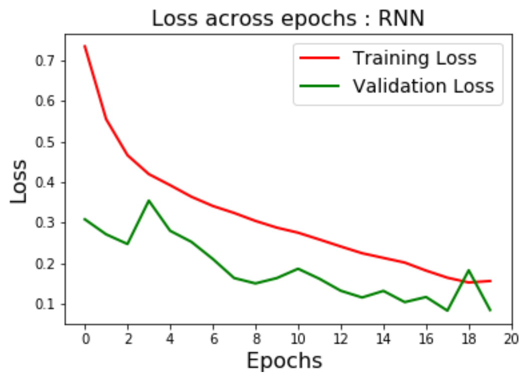


Fig 17. RNN on Yelp reviews: Loss over Epochs

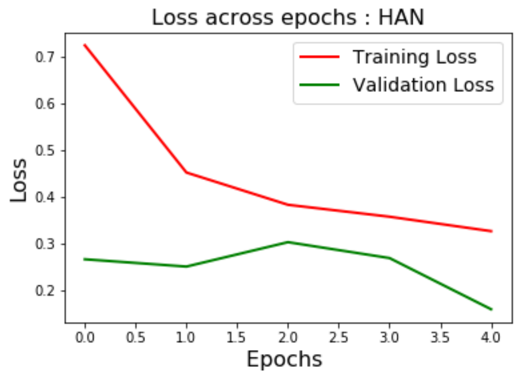


Fig 18. HAN on Yelp reviews: Loss over Epochs

10. Time per Epochs

<u>IMDB reviews dataset</u>

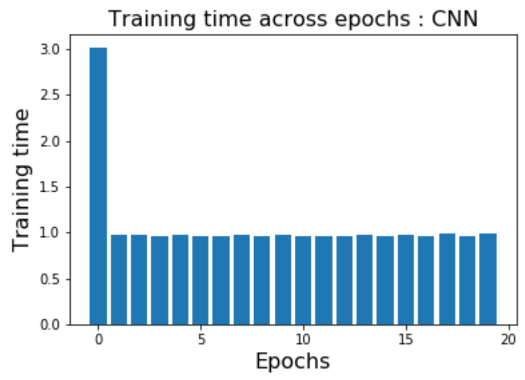


Fig 19. CNN on IMDB: Training time over Epochs

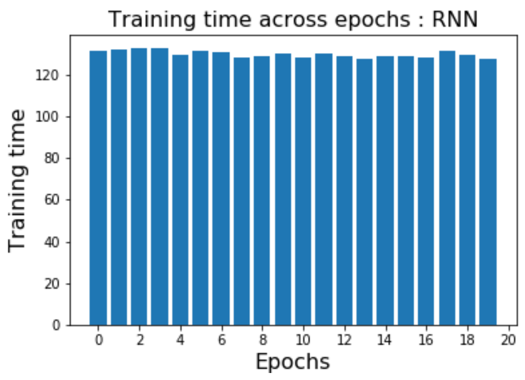


Fig 20. RNN on IMDB: Training time over Epochs

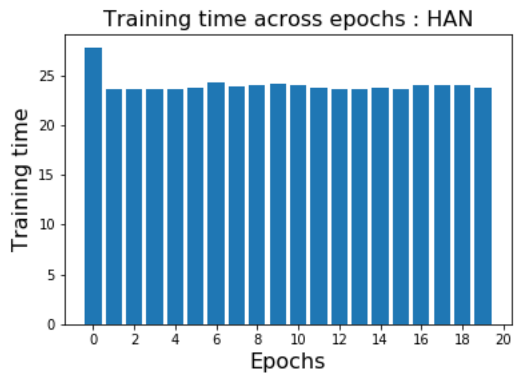


Fig 21. HAN on IMDB: Training time over Epochs

<u>Yelp Reviews dataset</u>

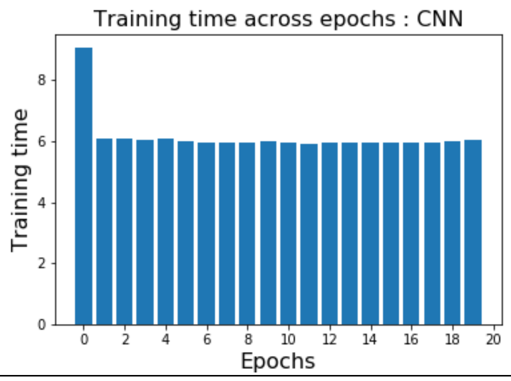


Fig 22. CNN on Yelp: Training time over Epochs

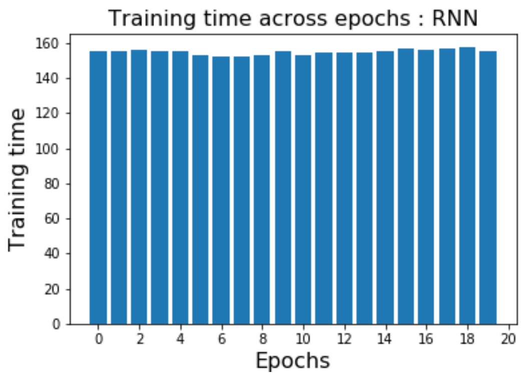


Fig 23. RNN on Yelp: Training time over Epochs

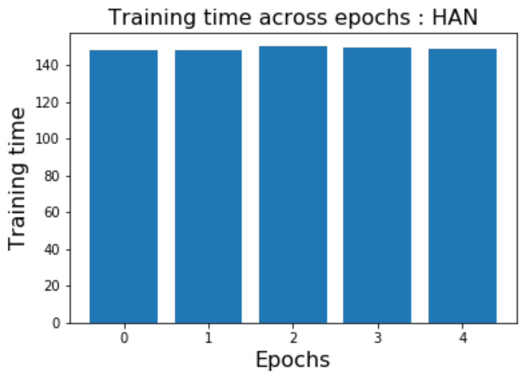


Fig 24. HAN on Yelp: Training time over Epochs

11. References

1. Hierarchical Attention Networks for Document Classification – Yang et. al
2. A Hierarchical Neural Autoencoder for Paragraphs and Documents – Jiwei Li, Minh-Thang Luong, Dan Jurafsky
3. Medium.com - Report on Text Classification using CNN, RNN & HAN -
4. Machine Learning Mastery - How to Grid Search Hyperparameters for Deep Learning Models in Python With Keras
5. Machine Learning Mastery – How to use Embedding Layers for Deep Learning with Keras
6. Richard’s Deep Learning Blog – Text Classification – Hierarchical Attention Network
7. Colah’s blog – Understanding LSTM Networks