CSC 591 - Capstone Project

Topic 4 : Comparative Analysis of CNN, RNN and HAN for Text Classification with GloVe Data Model

Rachit Shah (rshah25)
Sourabh Sandanshi (ssandan)

BI Use Case

- Text Classification is an example of supervised learning on a labelled dataset containing text documents into predefined categories.

- For this project, We have chosen to do News category classification - given news articles, classify them into appropriate predefined category.

Datasets

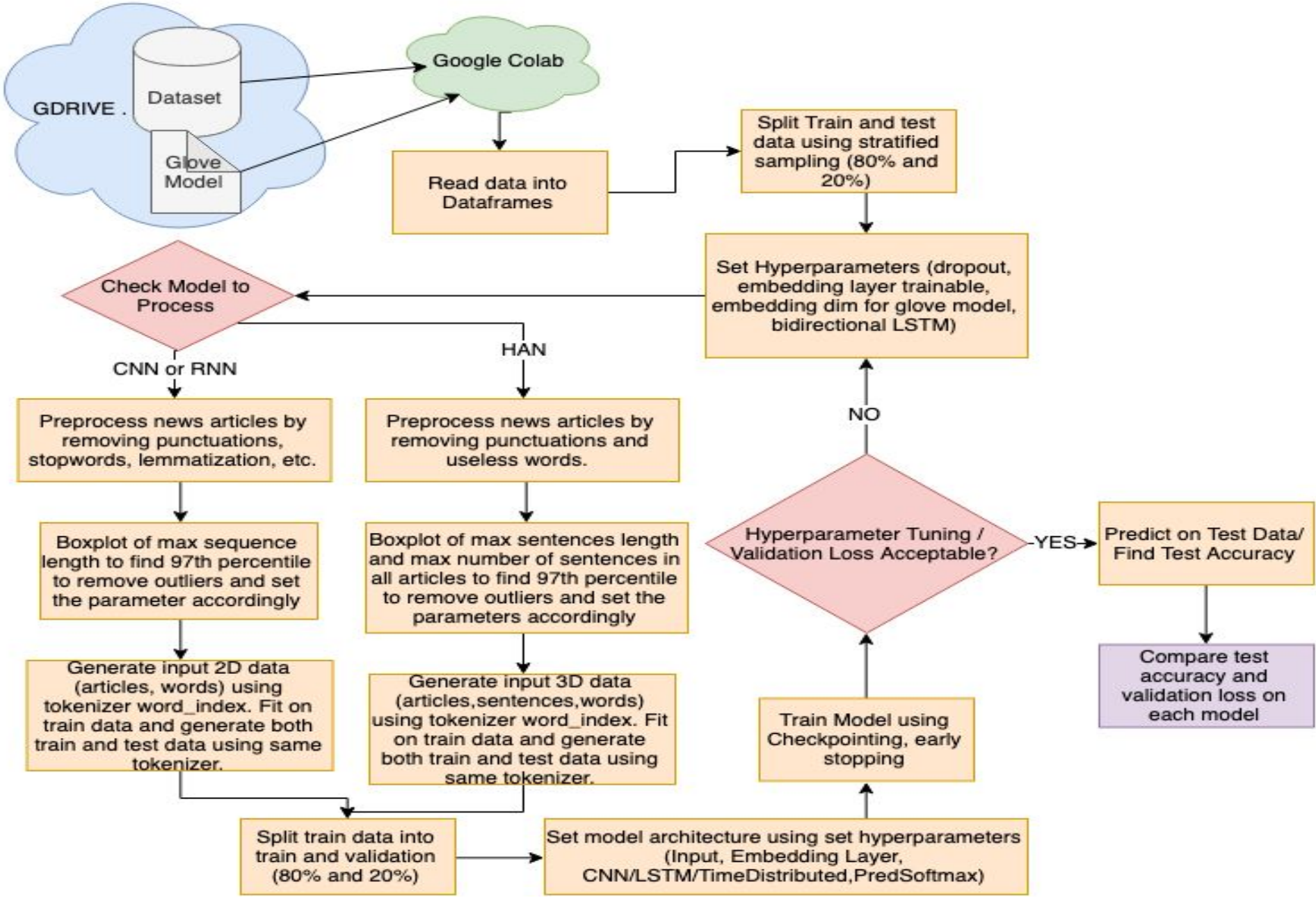
- We used two datasets to compare the architectures. This would help us find out what architecture is suitable given a dataset size and number of classes.
- First, we used the BBC news articles dataset from University College Dublin's Insight project. The dataset consists of 2225 documents from the BBC news website corresponding to stories in five topical categories (business, entertainment, politics, sports, tech) from 2004-2005
- Next we used the The 20 newsgroups dataset provided in scikit-learn comprising 19063 newsgroup posts on 20 topics

Dataset split

- The BBC dataset is divided as Train (64 %), Validation (16%), Test(20%) through stratified sampling.
- The 20 Newsgroup dataset was already divided into train/Validation and Test sets. With the following division - Train (48 %), Validation (12%), Test(40%)

Set	Train	Validation	Test
Dataset 1 (BBC)	1424	356	445
Dataset 2(20 Newsgroup)	9051	2263	7532

Solution Framework


GDRIVE .
Dataset
Glove Model
Google Colab
Read data into Dataframes
Split Train and test data using stratified sampling (80% and 20%)
Set Hyperparameters (dropout, embedding layer trainable, embedding dim for glove model, bidirectional LSTM)
Check Model to Process
CNN or RNN
HAN
Preprocess news articles by removing punctuations, stopwords, lemmatization, etc.
Preprocess news articles by removing punctuations and useless words.
Boxplot of max sequence length to find 97th percentile to remove outliers and set the parameter accordingly
Boxplot of max sentences length and max number of sentences in all articles to find 97th percentile to remove outliers and set the parameters accordingly
Generate input 2D data (articles, words) using tokenizer word_index. Fit on train data and generate both train and test data using same tokenizer.
Generate input 3D data (articles,sentences,words) using tokenizer word_index. Fit on train data and generate both train and test data using same tokenizer.
Split train data into train and validation (80% and 20%)
Set model architecture using set hyperparameters (Input, Embedding Layer, CNN/LSTM/TimeDistributed,PredSoftmax)
Train Model using Checkpointing, early stopping
Hyperparameter Tuning / Validation Loss Acceptable?
NO
YES
Predict on Test Data/ Find Test Accuracy
Compare test accuracy and validation loss on each model

Model Architecture

GloVe: Global Vectors for Word Representation

- GloVe is an unsupervised learning algorithm for obtaining vector representations for words.

- Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

- We used the Wikipedia 2014 + Gigaword 5 pre-trained vectors (6B tokens, 400K vocab, uncased, 50d, 100d, 200d, & 300d vectors)

Convolutional Neural Network

- Initially used for Image processing and computer vision applications
- The network consists of series of convolutions (extracting high level features from pixels of images) and pooling (downsampling to capture representative values).
- In context of Text classification, pixels can be replaces by words.
- We try to extract and understand the pattern on sentences which help us classify them.

Convolutional Neural Network

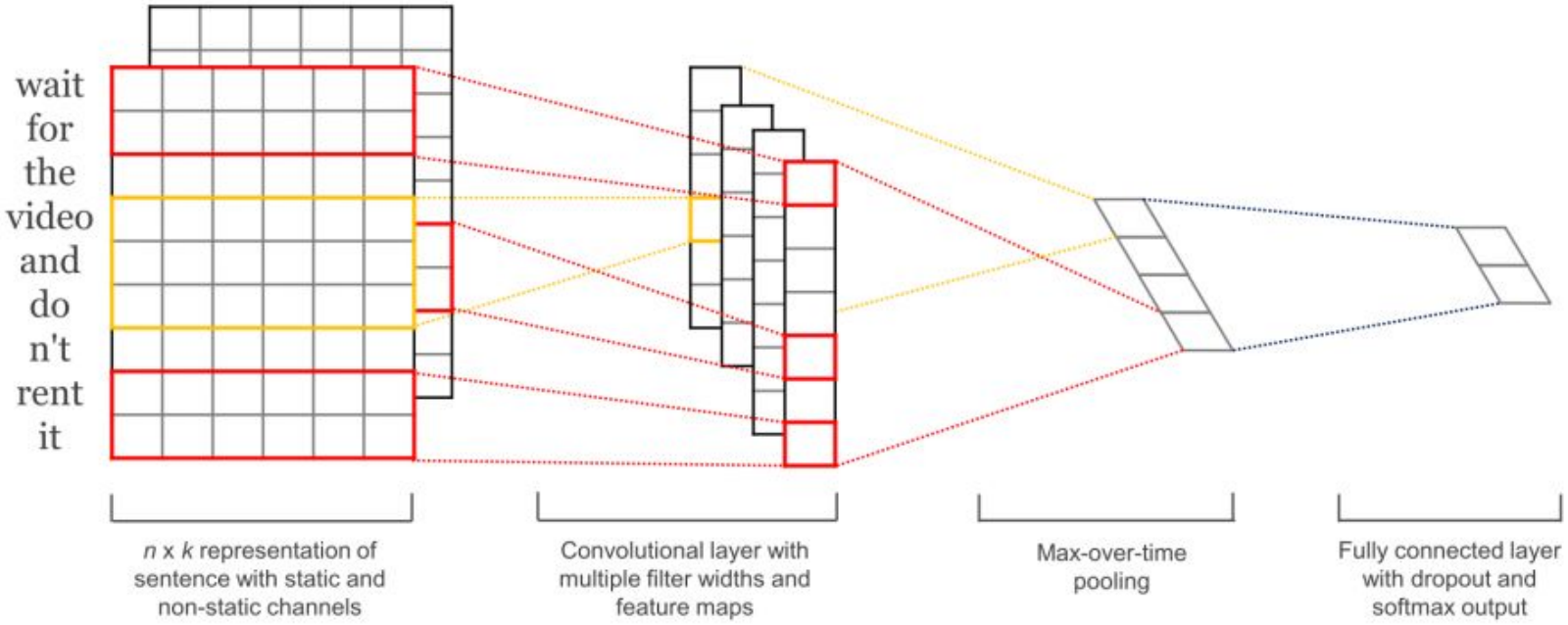
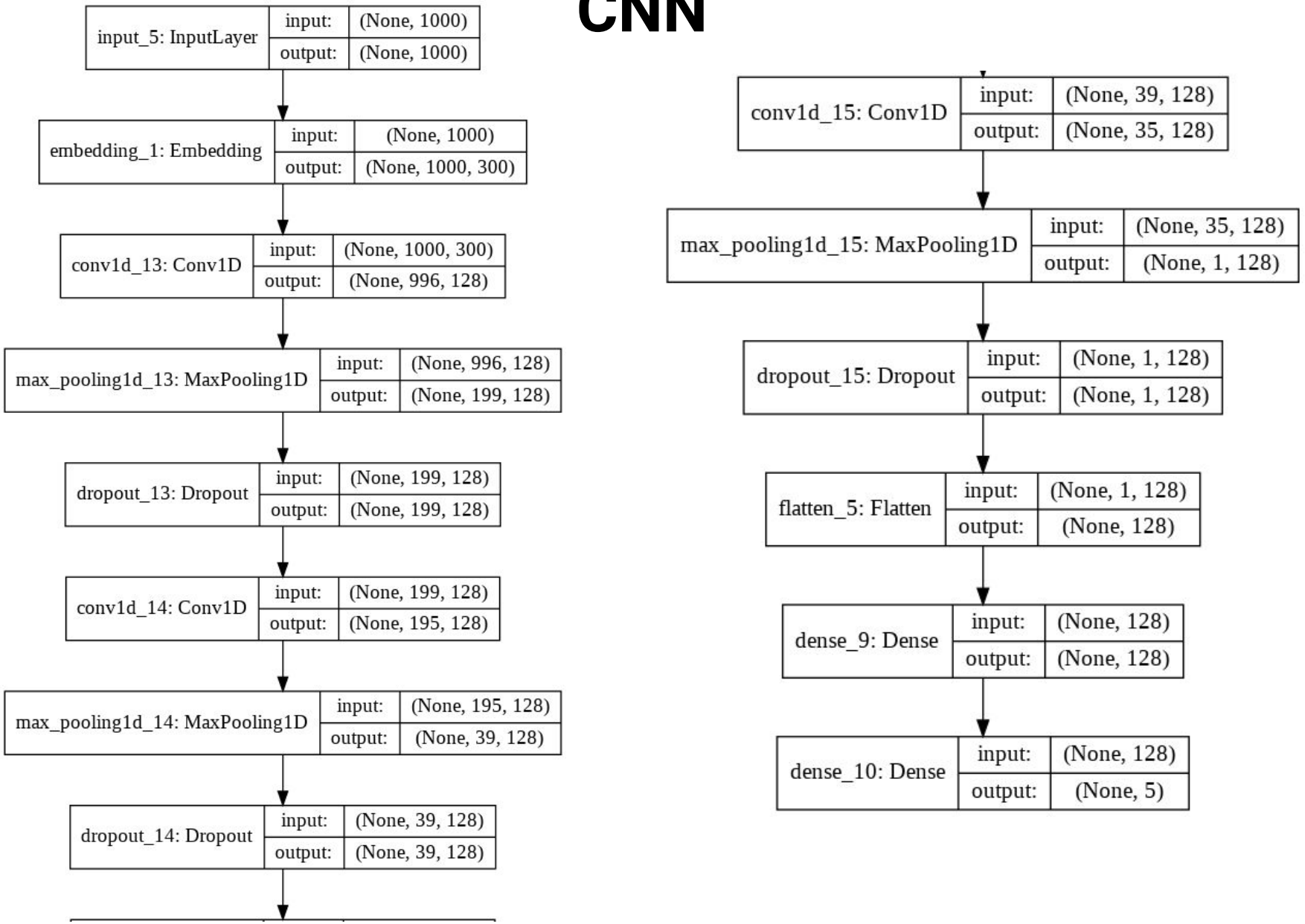


Image Reference : http://www.wildml.com/2015/12/implementing-a-cnn-for-text-classification-in-tensorflow/

CNN


input_5: InputLayer
input: (None, 1000)
output: (None, 1000)
embedding_1: Embedding
input: (None, 1000)
output: (None, 1000, 300)
conv1d_13: Conv1D
input: (None, 1000, 300)
output: (None, 996, 128)
max_pooling1d_13: MaxPooling1D
input: (None, 996, 128)
output: (None, 199, 128)
dropout_13: Dropout
input: (None, 199, 128)
output: (None, 199, 128)
conv1d_14: Conv1D
input: (None, 199, 128)
output: (None, 195, 128)
max_pooling1d_14: MaxPooling1D
input: (None, 195, 128)
output: (None, 39, 128)
dropout_14: Dropout
input: (None, 39, 128)
output: (None, 39, 128)
conv1d_15: Conv1D
input: (None, 39, 128)
output: (None, 35, 128)
max_pooling1d_15: MaxPooling1D
input: (None, 35, 128)
output: (None, 1, 128)
dropout_15: Dropout
input: (None, 1, 128)
output: (None, 1, 128)
flatten_5: Flatten
input: (None, 1, 128)
output: (None, 128)
dense_9: Dense
input: (None, 128)
output: (None, 128)
dense_10: Dense
input: (None, 128)
output: (None, 5)

Recurrent Neural Network

- RNN is a sequence of neural network blocks that are linked to each other like a chain. Each one is passing a message to a successor.
- They are networks with loops in them, allowing information to persist.
- Long Short Term Memory networks, usually just called “LSTMs” are a special kind of RNN, capable of learning long-term dependencies.

Recurrent Neural Network

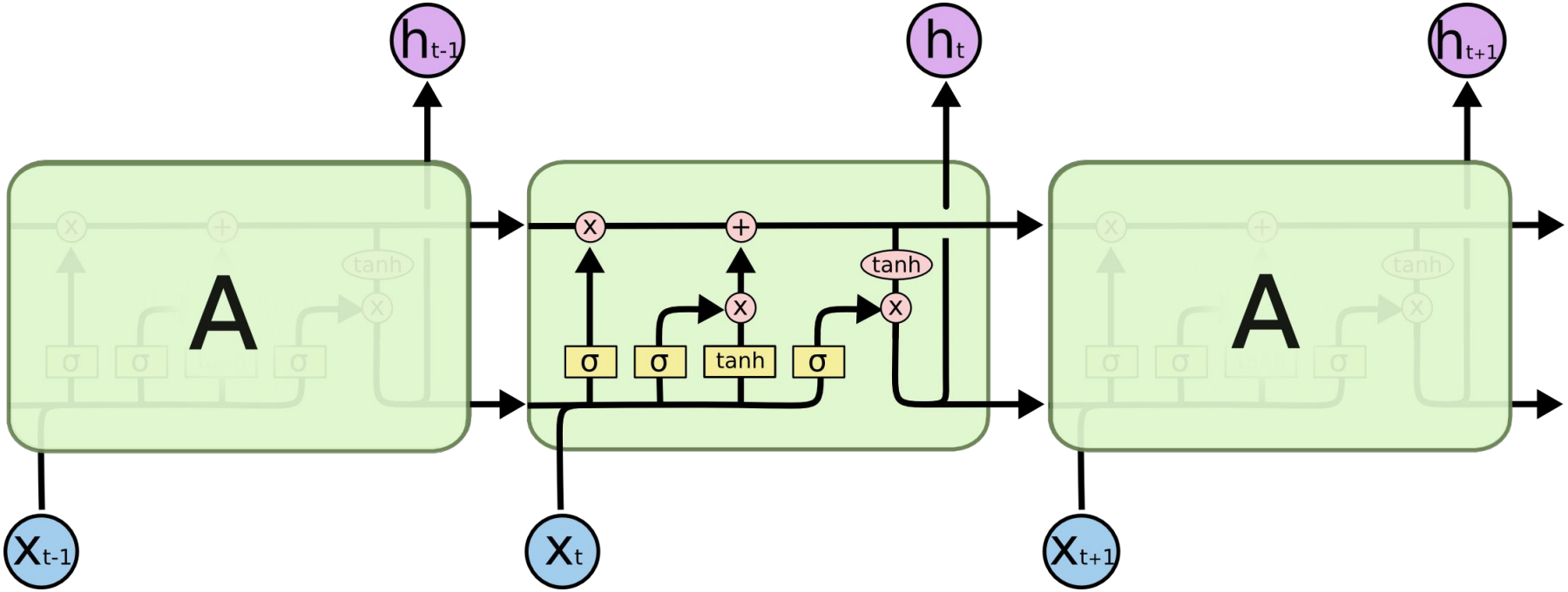
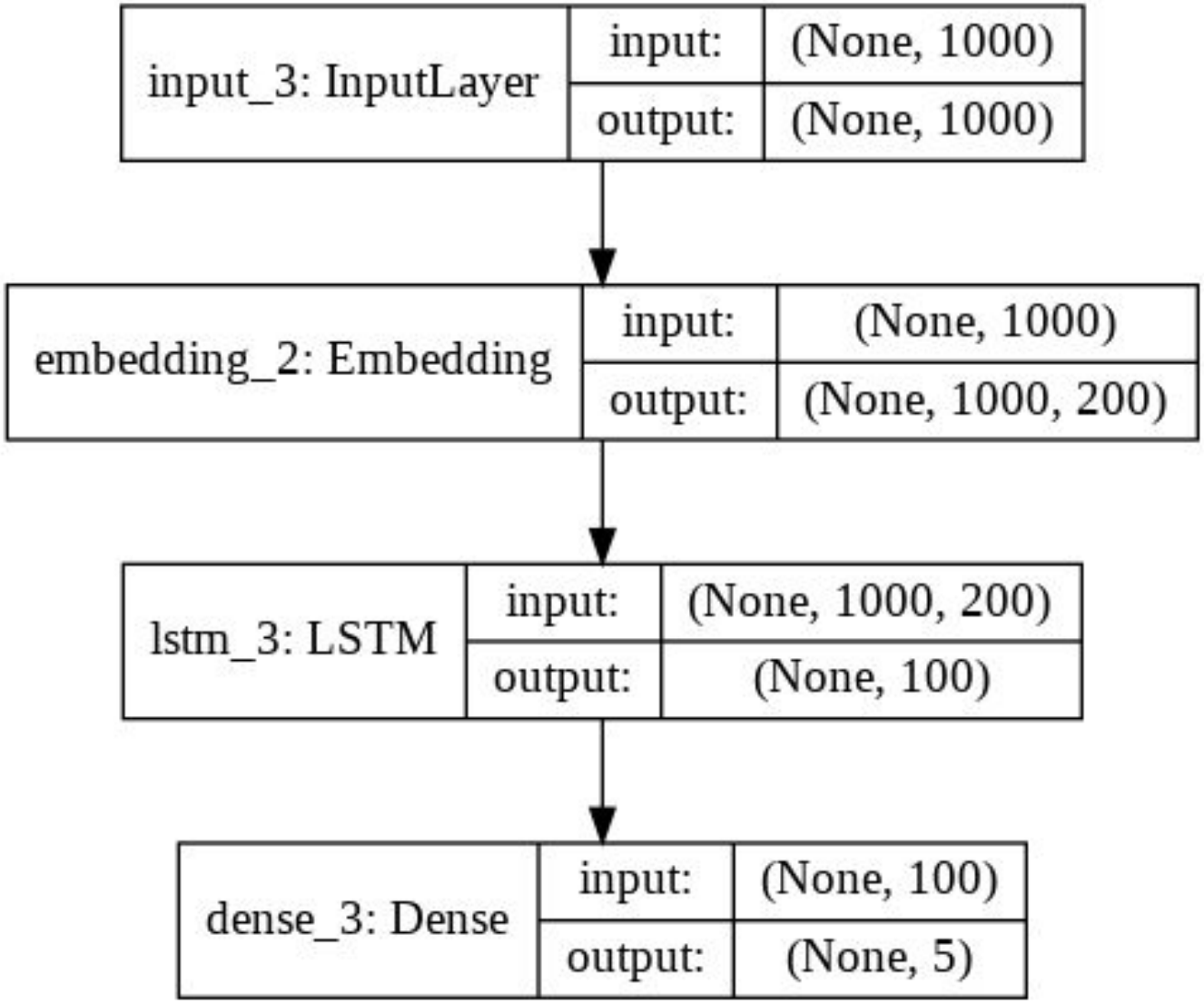


Image Reference : http://colah.github.io/posts/2015-08-Understanding-LSTMs/

The repeating module in an LSTM

RNN


input_3: InputLayer
input: (None, 1000)
output: (None, 1000)
embedding_2: Embedding
input: (None, 1000)
output: (None, 1000, 200)
lstm_3: LSTM
input: (None, 1000, 200)
output: (None, 100)
dense_3: Dense
input: (None, 100)
output: (None, 5)

Hierarchical Attention Network

- HAN has a hierarchical structure that mirrors the hierarchical structure of documents

- 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

Hierarchical Attention Network

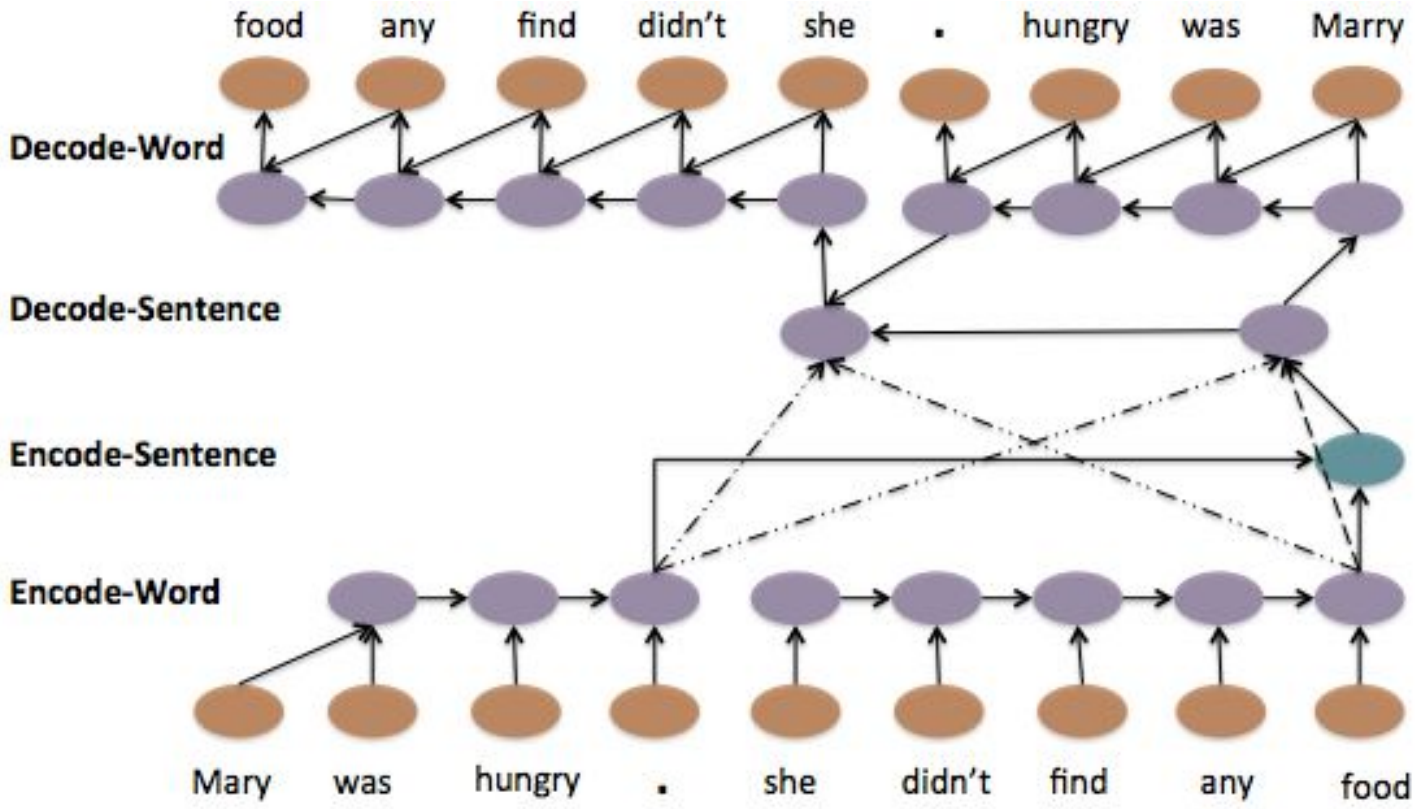
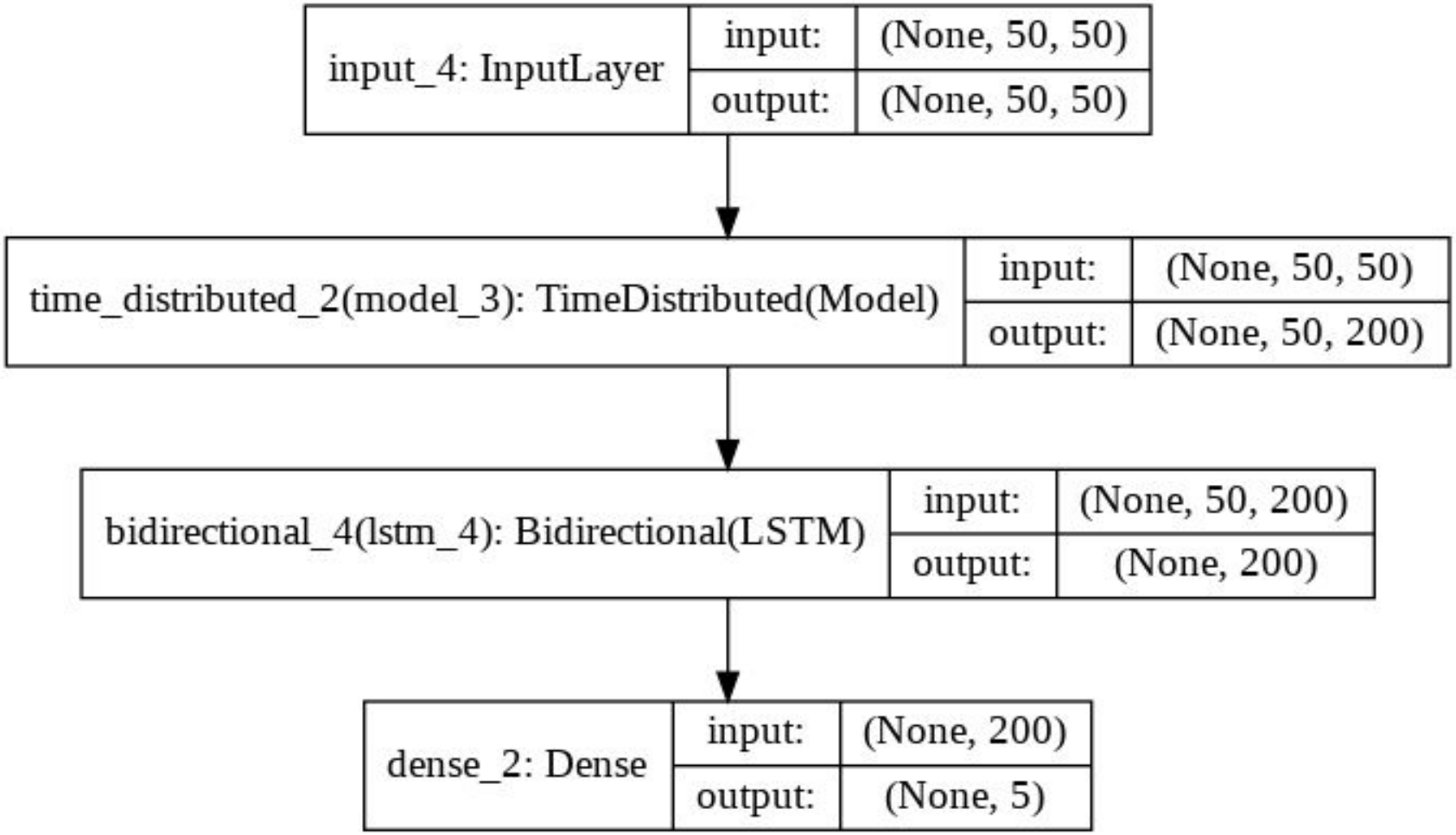


Image Reference :https://arxiv.org/pdf/1506.01057v2.pdf

Hierarchical Sequence to Sequence Model with Attention

HAN



The TimeDistributed layer is what allows the model to run a copy of the encoder to every sentence in the document.

Hyperparameter Tuning (Dataset 1)

Hyperparameter Tuning

We considered the following Hyperparameters to tune -

1. Embedding Layer Trainable
2. Embedding Dimensions
3. Dropout
4. Bidirectional LSTM or Unidirectional LSTM (for RNN)
5. MAX_SEQUENCE_LENGTH
6. MAX_SENTENCES
7. MAX_SENT_LENGTH

CNN

CNN			
Embedding Trainable	**Embedding Dim**	**Dropout**	**Validation Loss**
TRUE	100D	0.2	0.04462
FALSE	100D	0.2	0.06694
TRUE	300D	0.2	0.03009
TRUE	300D	0.4	**0.01323**
TRUE	300D	0.5	0.02558

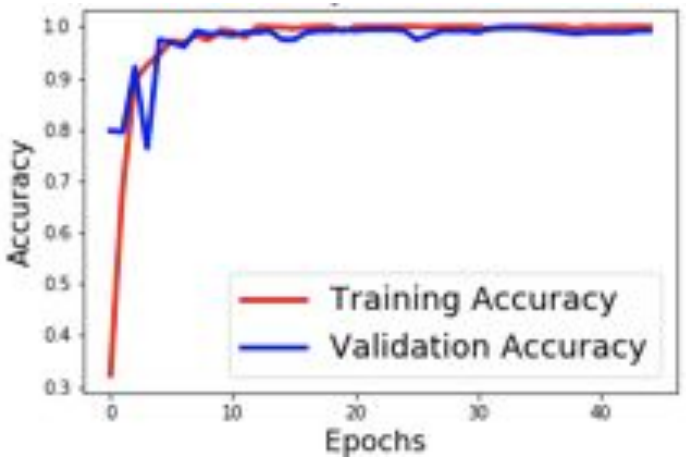
RNN

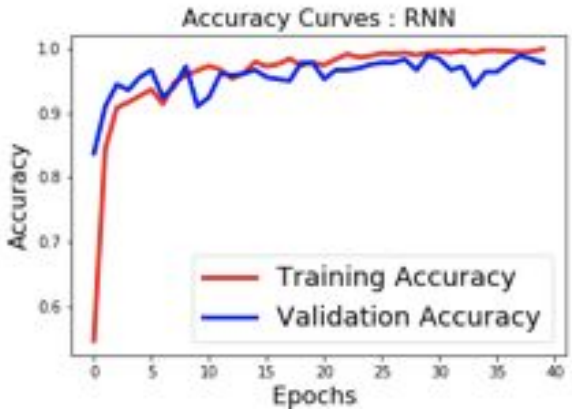
RNN				
Embedding Trainable	**Embedding Dim**	**Dropout**	**Bidirectional**	**Validation Loss**
TRUE	100D	0.2	FALSE	0.10813
FALSE	100D	0.2	FALSE	0.1482
TRUE	300D	0.2	FALSE	0.13059
TRUE	200D	0.2	FALSE	0.05499
TRUE	200D	0.4	FALSE	0.05803
TRUE	200D	0.2	TRUE	0.06209
TRUE	200D	0.4	TRUE	0.0731
TRUE	200D	0.3	FALSE	**0.04299**

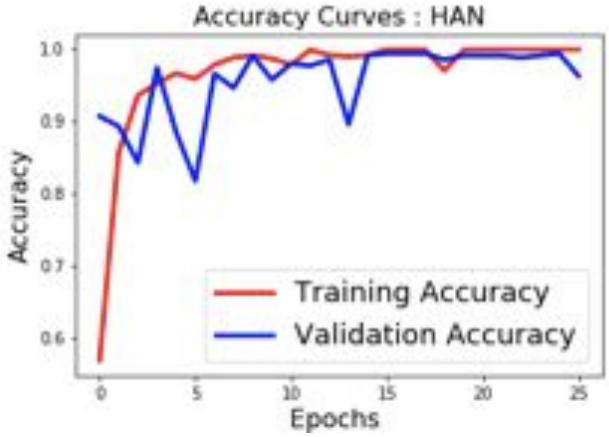
HAN

HAN			
Embedding Trainable	**Embedding Dim**	**Dropout**	**Validation Loss**
TRUE	100D	0.2	0.03365
FALSE	100D	0.2	0.06856
TRUE	300D	0.2	0.02858
TRUE	300D	0.4	0.03002
TRUE	300D	0.3	0.02612

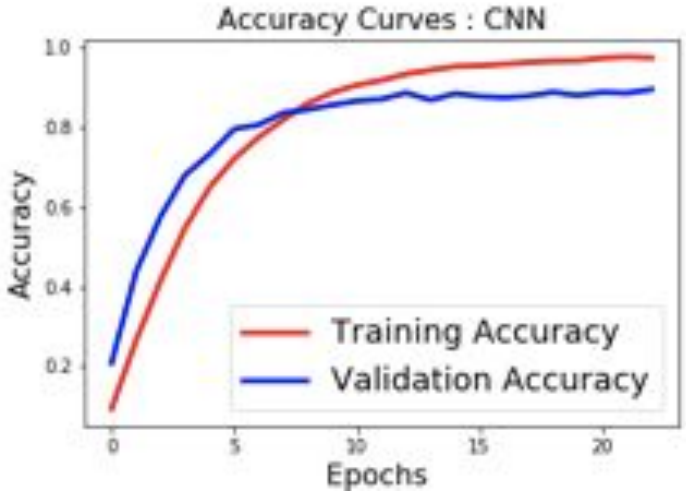
Accuracy Plots on Dataset 1

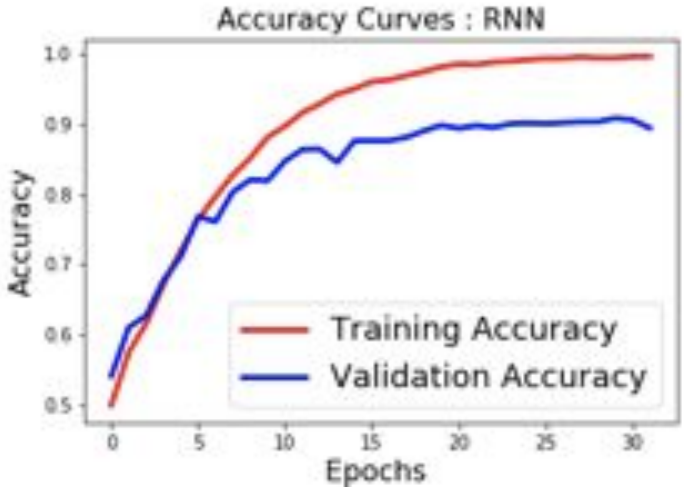


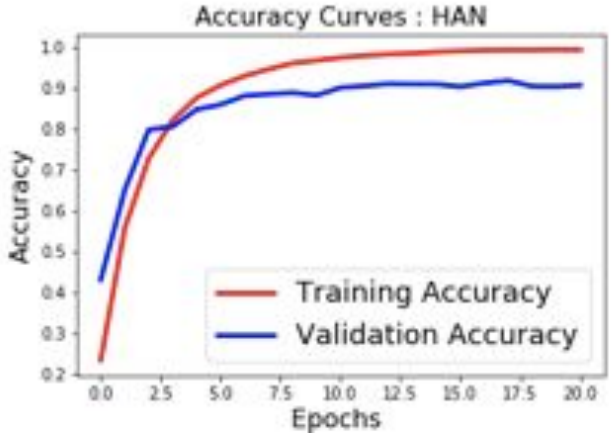




Accuracy Plots on Dataset 2







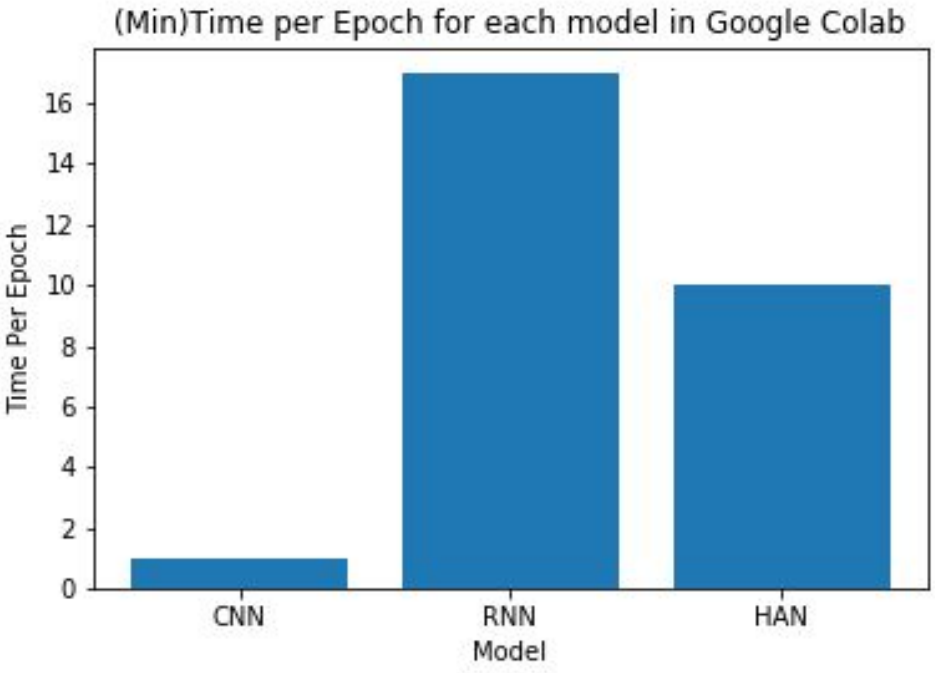
Best Parameters (Dataset 1)

Model	CNN	RNN	HAN
Validation Loss	0.01323	0.04299	0.02612
Test Accuracy	96.63	95.73	97.07
Embedding Dimension	300D	200D	300D
Dropout	0.4	0.3	0.3
Bidirectional LSTM	N/A	FALSE	TRUE
MAX_SEQUENCE_LENGTH (for CNN/RNN) or MAX_SENT_LENGTH (for HAN)	1000	1000	50
MAX_SENTS (for HAN)			50
MAX_NB_WORDS	20000	20000	20000
LSTM units		100	100
Optimizer	rmsprop	rmsprop	rmsprop
Activation (CNN)	relu	N/A	N/A

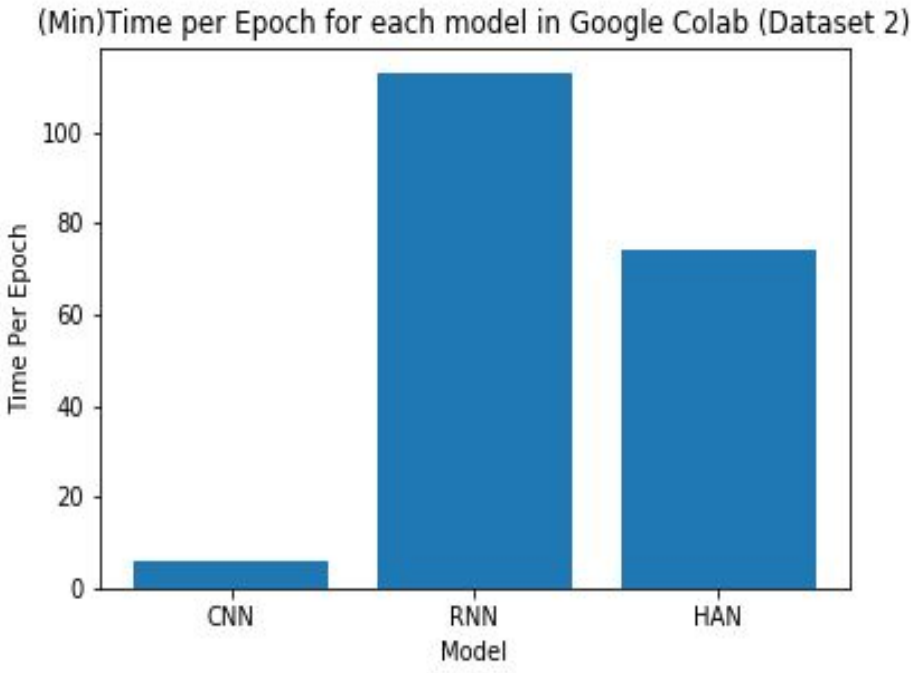
Best Parameters (Dataset 2)

Model	CNN	RNN	HAN
Validation Loss	0.48017	0.40511	0.36618
Test Accuracy	79.82	82.63	83.4
Dropout	0.5	0.3	0.3
Trainable	TRUE	TRUE	TRUE
Embedding DIM	300D	200D	300D
MAX_SEQ_LEN	1000	1000	N/A
MAX_SENT_LEN	N/A	N/A	15
MAX_SENT	N/A	N/A	143

Minimum Time per Epoch (seconds)



Dataset 1 (BBC)



Dataset 2 (20 Newsgroup)

Conclusion

- From our experiment, we found that while CNN had the lowest validation loss on dataset 1, the test accuracy of HAN was highest even though it had higher validation loss compared to CNN.

- We can see that the difference between the test accuracies is very less. Hence, if someone needs to train a model faster, they could choose CNN over HAN.

Conclusion (contd.)

- For dataset 2, we had considered a data with higher number of records and more classes. HAN performed the best on both validation loss and test accuracy while CNN performed the worst on both.

- This may have been because of larger dataset; HAN was able to retrieve a deeper understanding/context of the data.

- Overall, HAN performed consistently better for both types of datasets and it also took average time to train compared to CNN and RNN.