Text Classification using CNN, RNN and HAN

Authors: Tejas Gupta, Tushar Dahibhate, Shraddha Dhyade

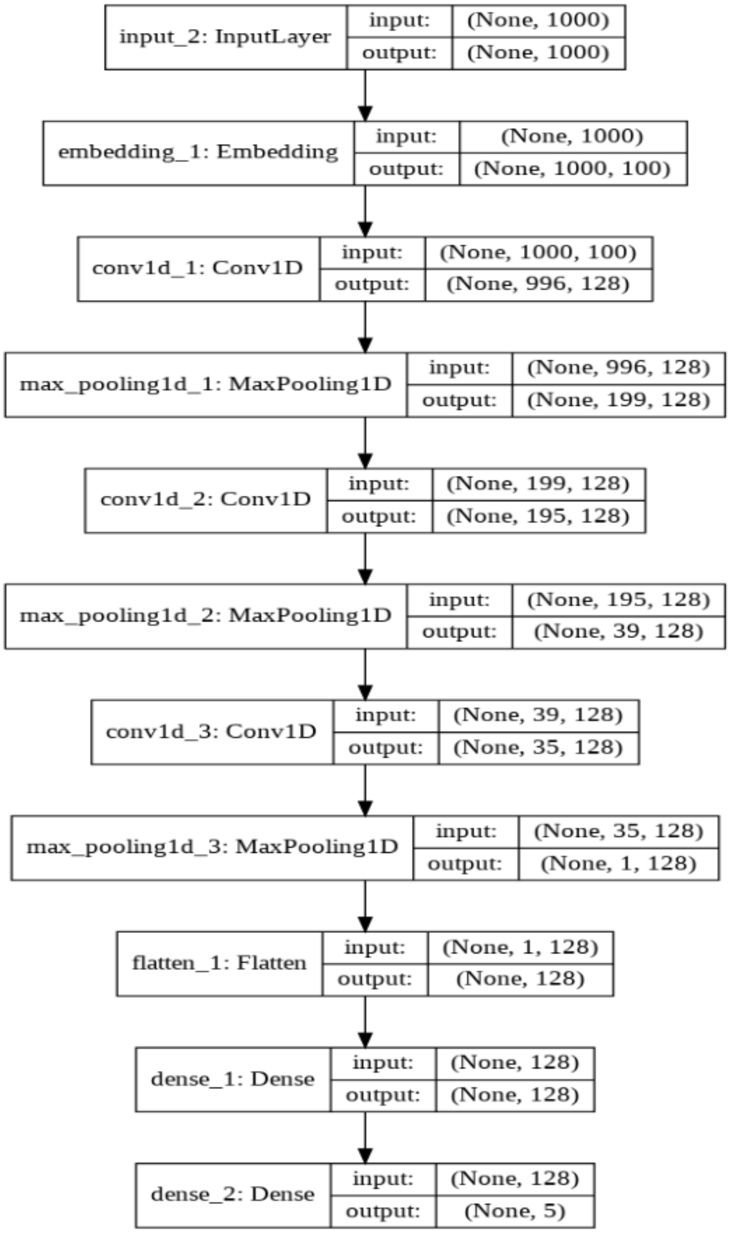
Text classification problem:

Using deep learning, we aim to perform a sentiment analysis of the yelp online reviews. The reviews are in the form of star ratings(1-5) and textual format. Thus, we use the pretrained GloVe vectors dataset and perform a sentiment analysis to later perform comparison of the aforementioned models.

Description of the data sets:

The yelp dataset consists of reviews with star ratings 1-5 and textual reviews and other information such as user id, review id usefulness of the review, business id, date, review id, user id and comments on the review, given by other users. However, the ratings and reviews are of prime importance to us in this analysis. The dataset has 10000 samples which were divided into train test and validation datasets as follows: The training set contains 6000 samples, validation data has 2000 samples and test data has 2000 samples.

Architectures: Visual Graphs



input_2: InputLayer | input: (None, 1000) | output: (None, 1000)

embedding_1: Embedding | input: (None, 1000) | output: (None, 1000, 100)

conv1d_1: Conv1D | input: (None, 1000, 100) | output: (None, 996, 128)

max_pooling1d_1: MaxPooling1D | input: (None, 996, 128) | output: (None, 199, 128)

conv1d_2: Conv1D | input: (None, 199, 128) | output: (None, 195, 128)

max_pooling1d_2: MaxPooling1D | input: (None, 195, 128) | output: (None, 39, 128)

conv1d_3: Conv1D | input: (None, 39, 128) | output: (None, 35, 128)

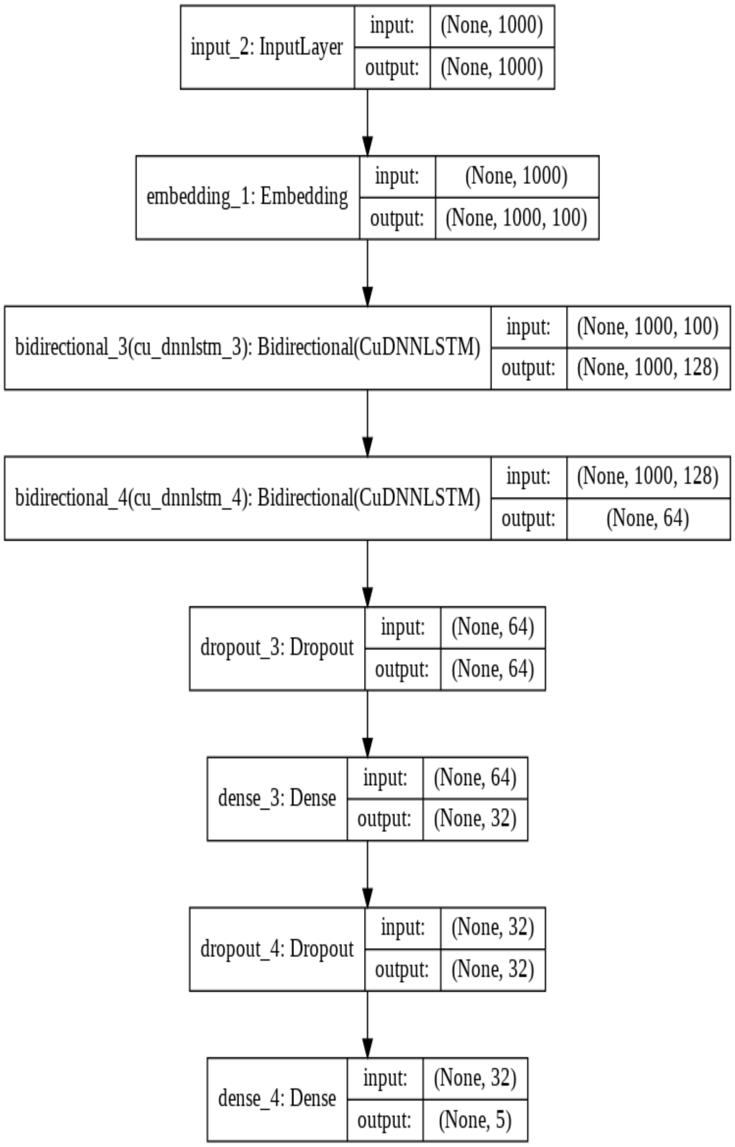
max_pooling1d_3: MaxPooling1D | input: (None, 35, 128) | output: (None, 1, 128)

flatten_1: Flatten | input: (None, 1, 128) | output: (None, 128)

dense_1: Dense | input: (None, 128) | output: (None, 128)

dense_2: Dense | input: (None, 128) | output: (None, 5)

CNN



input_2: InputLayer | input: (None, 1000) | output: (None, 1000)

embedding_1: Embedding | input: (None, 1000) | output: (None, 1000, 100)

bidirectional_3(cu_dnnlstm_3): Bidirectional(CuDNNLSTM) | input: (None, 1000, 100) | output: (None, 1000, 128)

bidirectional_4(cu_dnnlstm_4): Bidirectional(CuDNNLSTM) | input: (None, 1000, 128) | output: (None, 64)

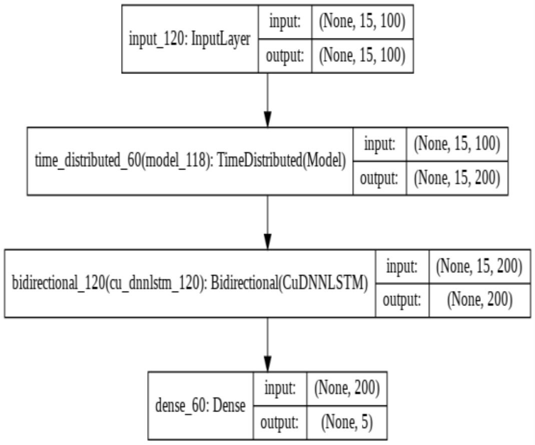
dropout_3: Dropout | input: (None, 64) | output: (None, 64)

dense_3: Dense | input: (None, 64) | output: (None, 32)

dropout_4: Dropout | input: (None, 32) | output: (None, 32)

dense_4: Dense | input: (None, 32) | output: (None, 5)

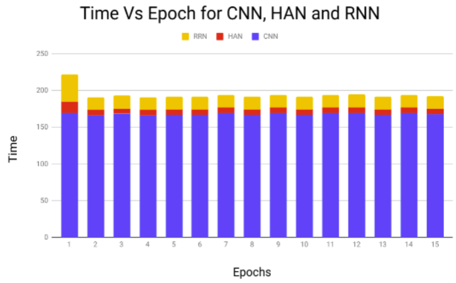
RNN



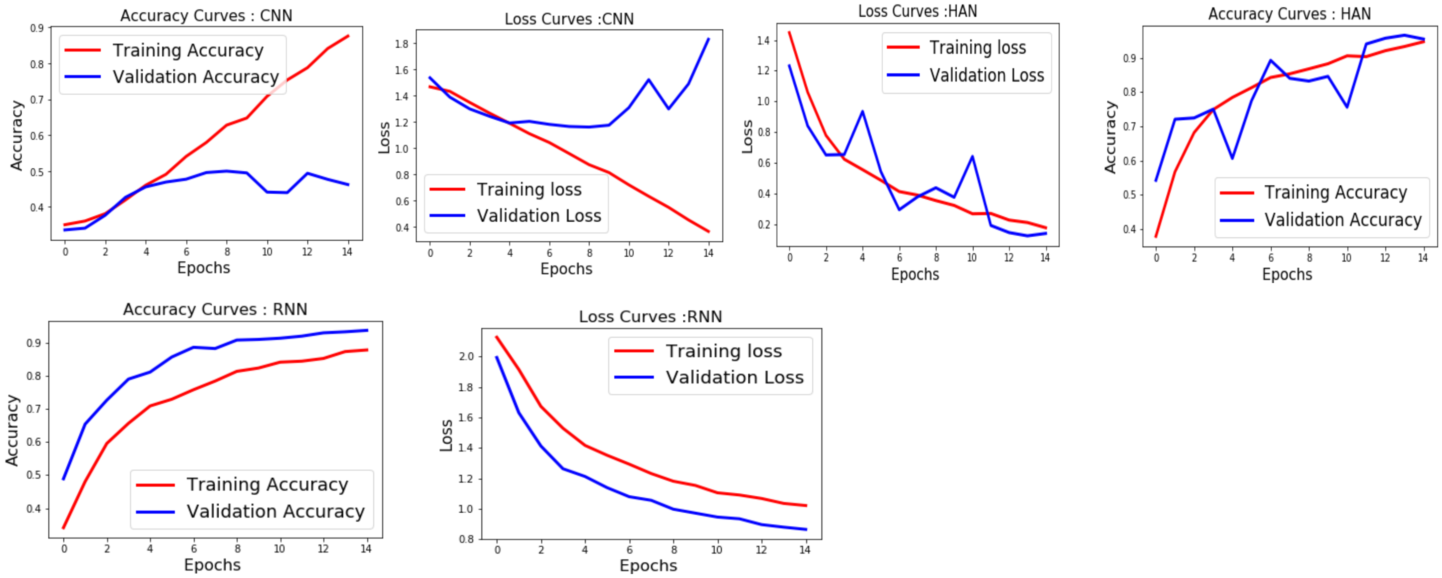
HAN

Architectures: Hyperparameter Table			
	CNN	RNN	HAN
Epochs	15	15	15
Batch Size	128	64	256
Learning rate	0.001	0.0001	0.001
L1 Regularization	default	0.01	0.00001

Time/Epochs graph:



Training and Validation Accuracy and Loss over Epochs:



Performance Analysis:

Test accuracy of 0.9355 was achieved in HAN model, while RNN gave 0.8529 accuracy and CNN gave 0.7319 accuracy. Thus, we can say that HAN performed better among the three.

Hyperparameter Tuning: Choices, rationale, observed impact on the model performance:

We tuned the model by tweaking following hyperparameters-

- **Learning rate-** When the model was overfitting, decreasing the learning rate would increase the performance while on underfitting, the increasing the learning rate would increase the model performance.
- **Regularization parameter–** decreasing the L1 regularization parameters improved the model performance
- **Batch size-** Increasing the batch size improved the model performance.