**Capstone Project:- Topic 3**

**Text Classification: A Comparative Analysis of Different Deep Neural Network Architectures**

**Team Member Unity IDs:**

pushende

pkponnap

ssmehend

**1. Task chosen:**

**Comparative analysis of RNN, CNN and HAN on Movie Review sentiment classification**

**2. Description:**

Sentiment analysis involves classifying the polarity of texts to determine whether the contents of the text is positive, negative, or neutral.

In the traditional approach to text classification, texts are treated as bags of words. This approach is simple but it ignores the context of words. Modern methods of text classification treat texts as sequences of words and use neural networks of different flavours to train classification models. In our project, we consider classifying reviews of movies as either positive or negative using three different recurrent neural network models, i e Convolutional Neural Networks(CNN), Recurrent Neural Networks(RNN) and Hierarchical Attention Networks(HAN).

**3. Dataset: IMDB reviews dataset**

The data was compiled by Andrew Maas and can be found here: <https://ai.stanford.edu/~amaas/data/sentiment/>

IMDB reviews dataset consists of 50,000 reviews split equally into train and test.

The train dataset and test dataset has 12.5k positive and 12.5k negative reviews each.

IMDb lets users rate movies on a scale from 1 to 10. The curator of the data labeled anything with ≤ 4 stars as negative and anything with ≥ 7 stars as positive. Reviews with 5 or 6 stars were left out.

**4. Our Approach**

* **We used Keras for the implementation of Neural Networks.**
* **Data Preprocessing**

The original reviews are stored in text files. Positive and negative reviews are stored in different folders. We extract the texts from the files and insert them into lists for further preprocessing of the texts.

Preprocessing steps include:

1. Labelling the positive texts with 1 and negative texts with 0
2. Removing punctuation and special characters
3. Splitting and tokenizing text.
4. Creating word to unique word index map
5. Replacing words with unique word indices.

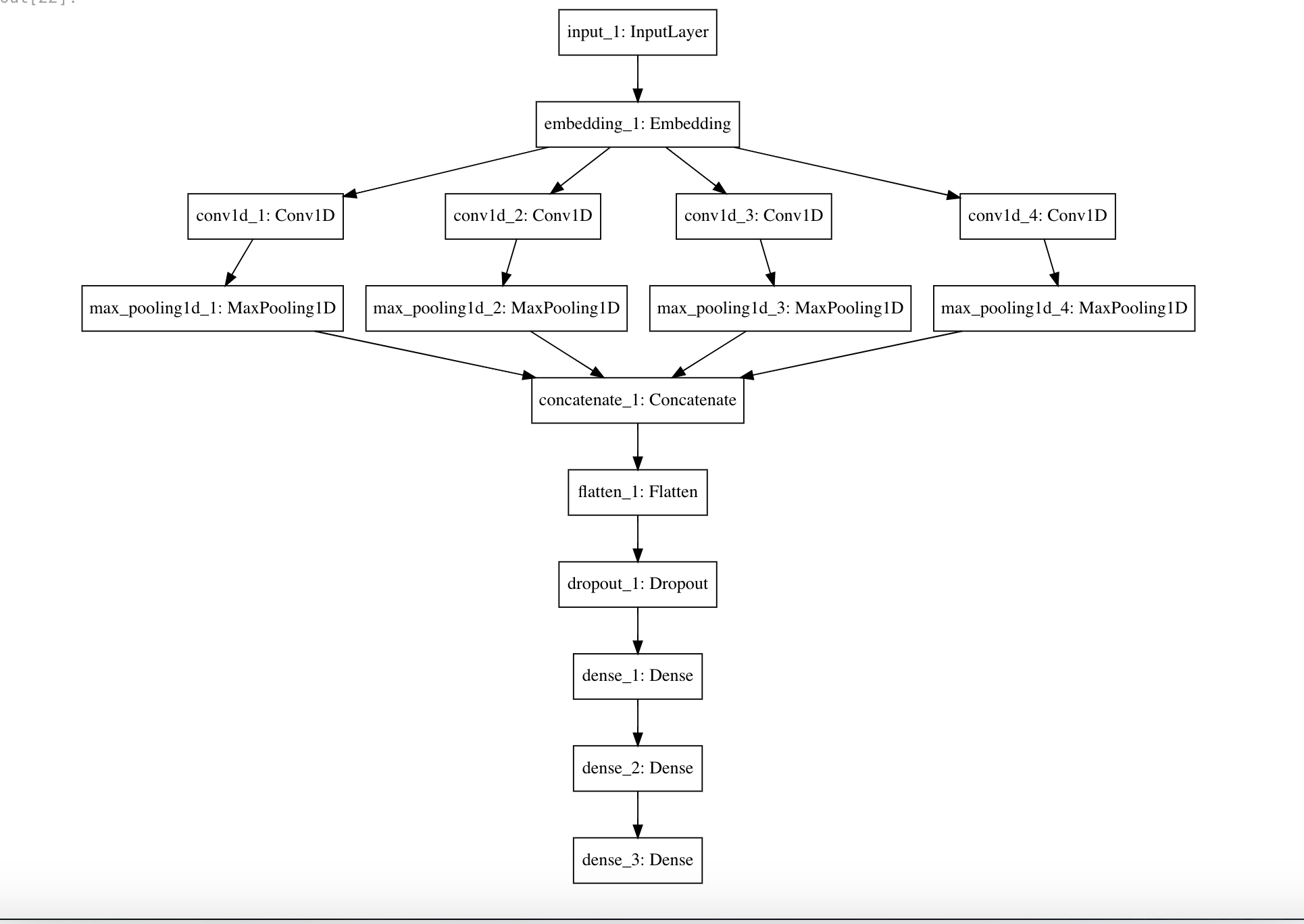
* **Word Embeddings and their usage**
  + We used the pretrained GloVe vector embeddings which are freely available at <https://nlp.stanford.edu/projects/glove/>
  + There are multiple flavours of pretrained GloVe word vectors available.For our implementation, we have used 300 dimensional word vectors trained on Common Crawl data (840B tokens, 2.2M vocab, cased).
  + Since these embeddings are trained on huge text corpus and also capture the context of the words, they were used to seed the Embedding layer in each of the Neural Networks in order to get better results.
* **Convolutional Neural Network (CNN)**
  + The idea was to vary the size of the kernels and concatenate their outputs, in order to allow the neural network model to detect patterns of multiples sizes of words.
  + We started with the kernel size of 1 and varied till 4 so as to capture the context of words and make better classifications.
* **Recurrent Neural Networks (RNN)**
  + We used the Bidirectional LSTM network bidirectional LSTM and concatenate both last output of LSTM outputs.
* **HAN**
  + The idea in the hierarchical attention neural network was to learn context of the reviews from words followed by a single sentence and then multiple sentences at once.
  + The attention layer here is used to understand the context of a word in different sentences. And then, give due weightage to words which matter more like “amazing” over “better”.
  + This way, first the context of words, then context from sentences was learnt to classify the review.

**5. Train/Test/Validation data size**

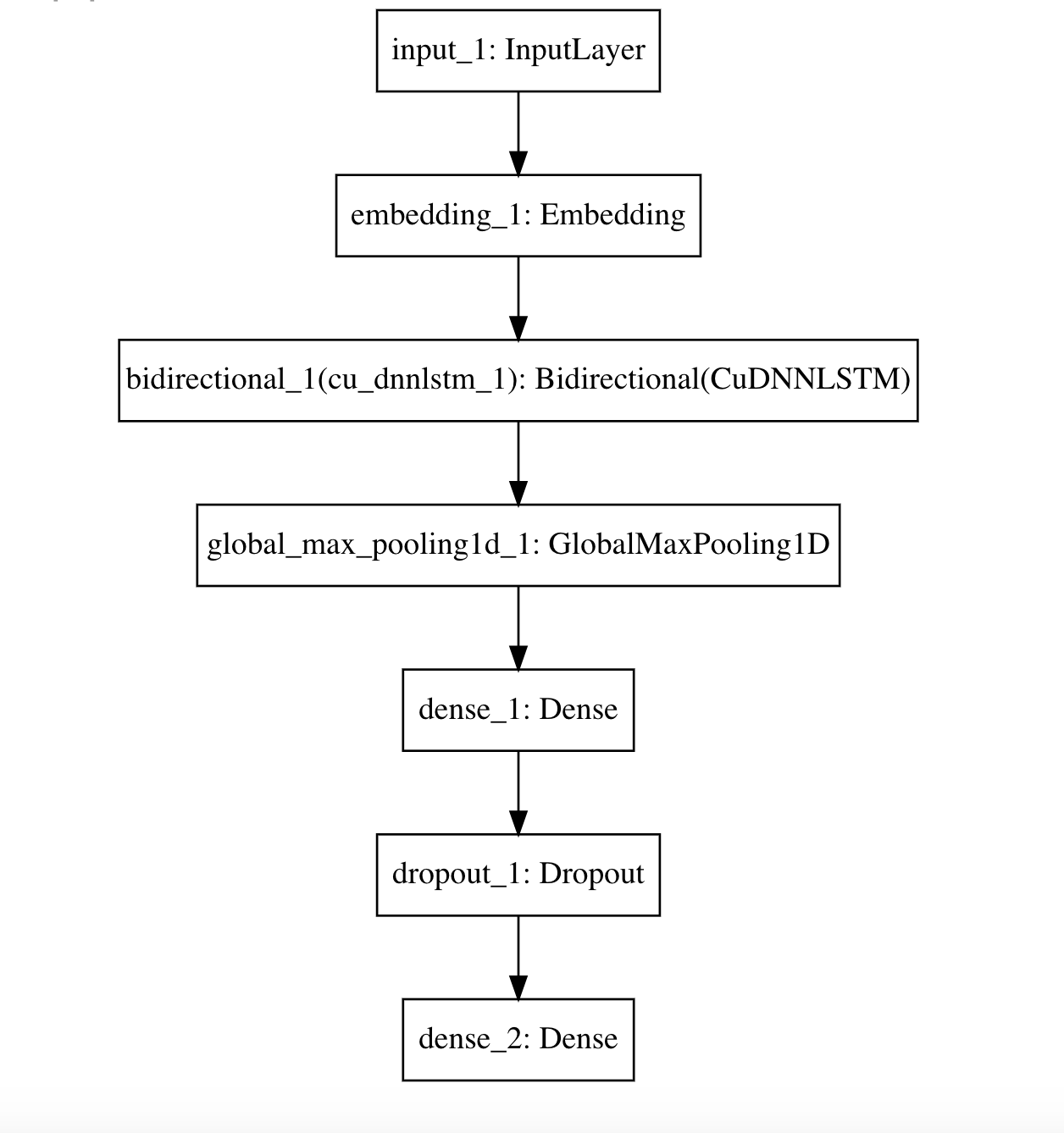
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dataset: IMDB reviews (total 50k reviews)** | **Size** | **Average number of words in each review** | **Max. number of words in review** | **Min. number of words in review** |
| Train | 17500 | 234 | 2470 | 10 |
| Validation | 7500 | 230 | 2357 | 9 |
| Test | 25,000 | 229 | 2278 | 4 |

**6. Architectures: visual graphs**

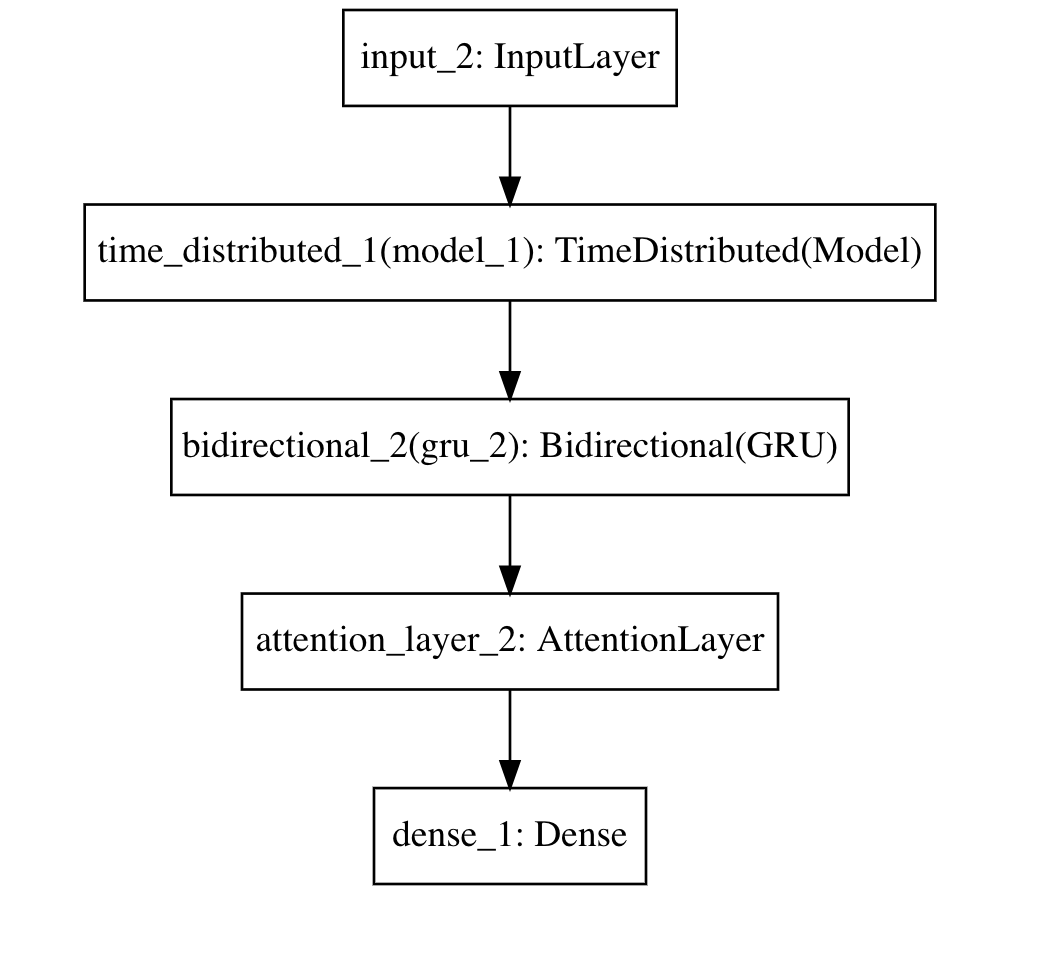
* **Architecture of CNN**

****

* **Architecture of RNN**

****

* **Architecture of HAN**

****

**7. Architecture hyperparameters:**

**CNN**

|  |  |
| --- | --- |
| **Layers** | **No. of units / Size** |
| Conv1D, MaxPool1D | filters=512, filter\_size =(1,1) |
| Conv1D, MaxPool1D | filters=512, filter\_size =(2,2) |
| Conv1D, MaxPool1D | filters=512, filter\_size =(3,3) |
| Conv1D, MaxPool1D | filters=512, filter\_size =(4,4) |
| Concatenate, Flatten |  |
| Dropout | prob=0.3 |
| Dense(relu) | No.of units=128 |
| Dense(relu) | No.of units=64 |
| Dense(sigmoid) | No.of units=1 |

**Optimizer:** adam

**Batch size:** 256

**RNN**

|  |  |
| --- | --- |
| **Layer** | **No. of units/Size** |
| Bidirectional(LSTM) | No.of units=512 |
| GlobalMaxPool1D |  |
| Dense(relu) | No. of units=64 |
| Dropout | prob=0.3 |
| Dense(sigmoid) | No. of units=1, |

**Optimizer:** adam

**Batch size:** 256

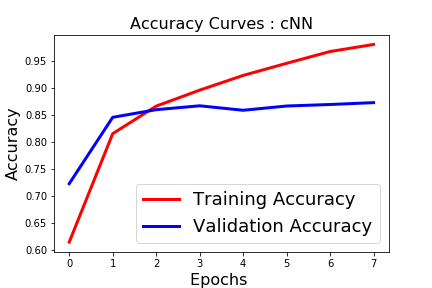
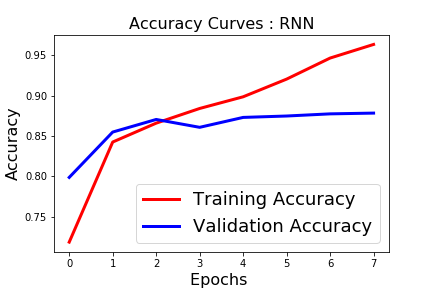
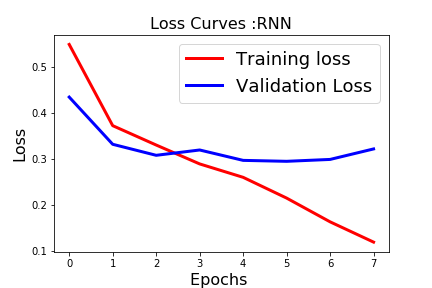
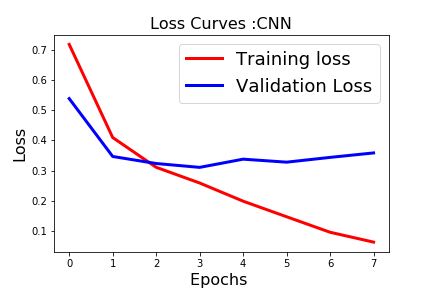
**HAN**

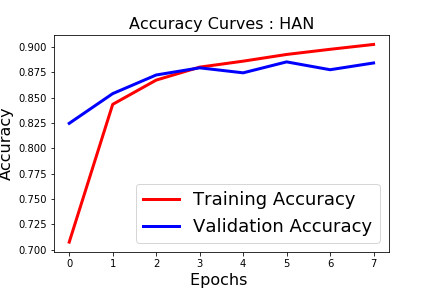
|  |  |
| --- | --- |
| **Layer** | **No. of units/Size** |
| Bidirectional(GRU)(words) | No. of units=20 |
| AttentionLayer(words) | No. of units=20 |
| Bidirectional(GRU)(sentences) | No. of units=20 |
| AttentionLayer(sentences) | No. of units=22 |
| Dense(softmax) | No. of units=2 |

**Optimizer:** adam

**Batch size:** 256

**8. Training and Validation Accuracy and Loss over Epochs: line graphs**

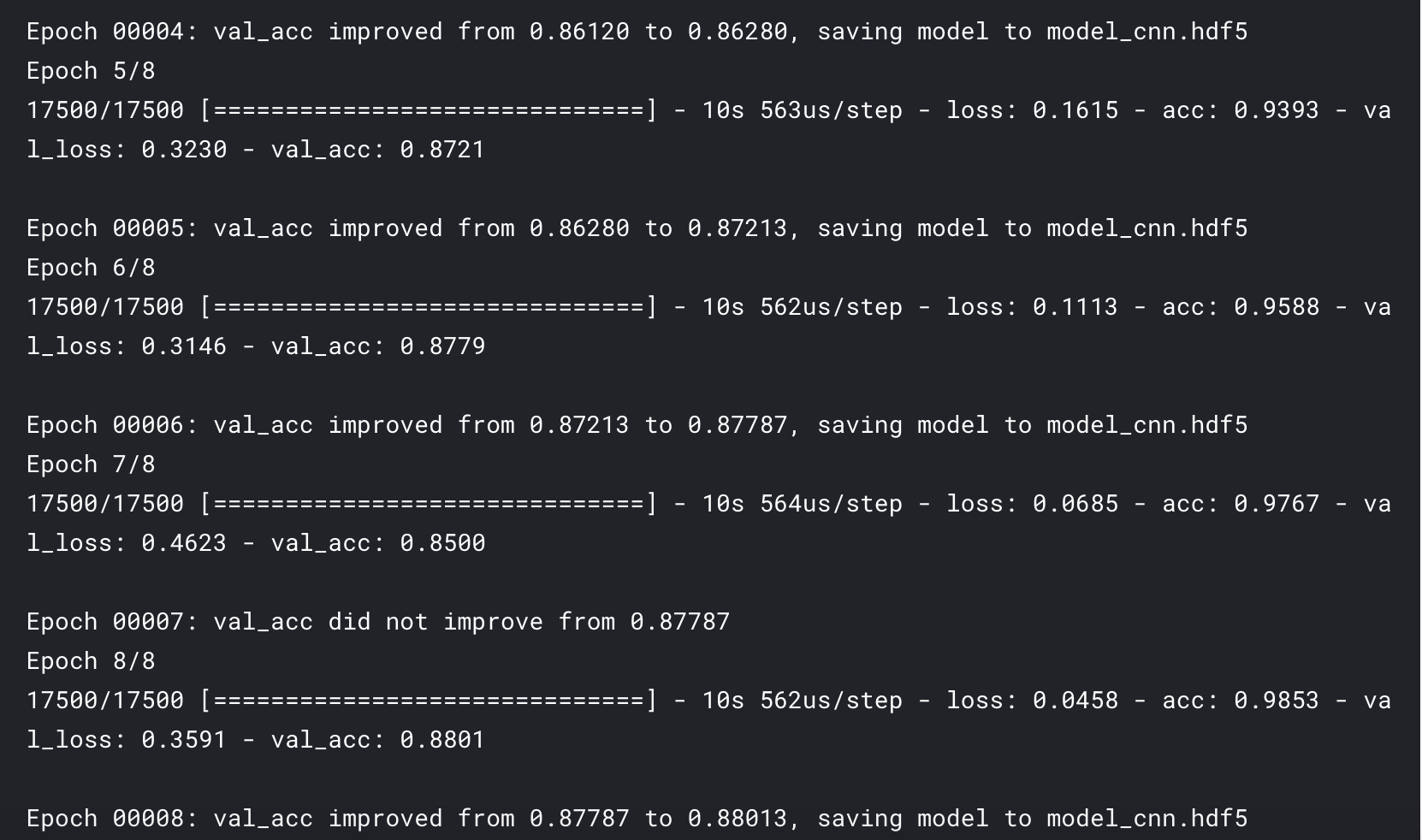
** **

****

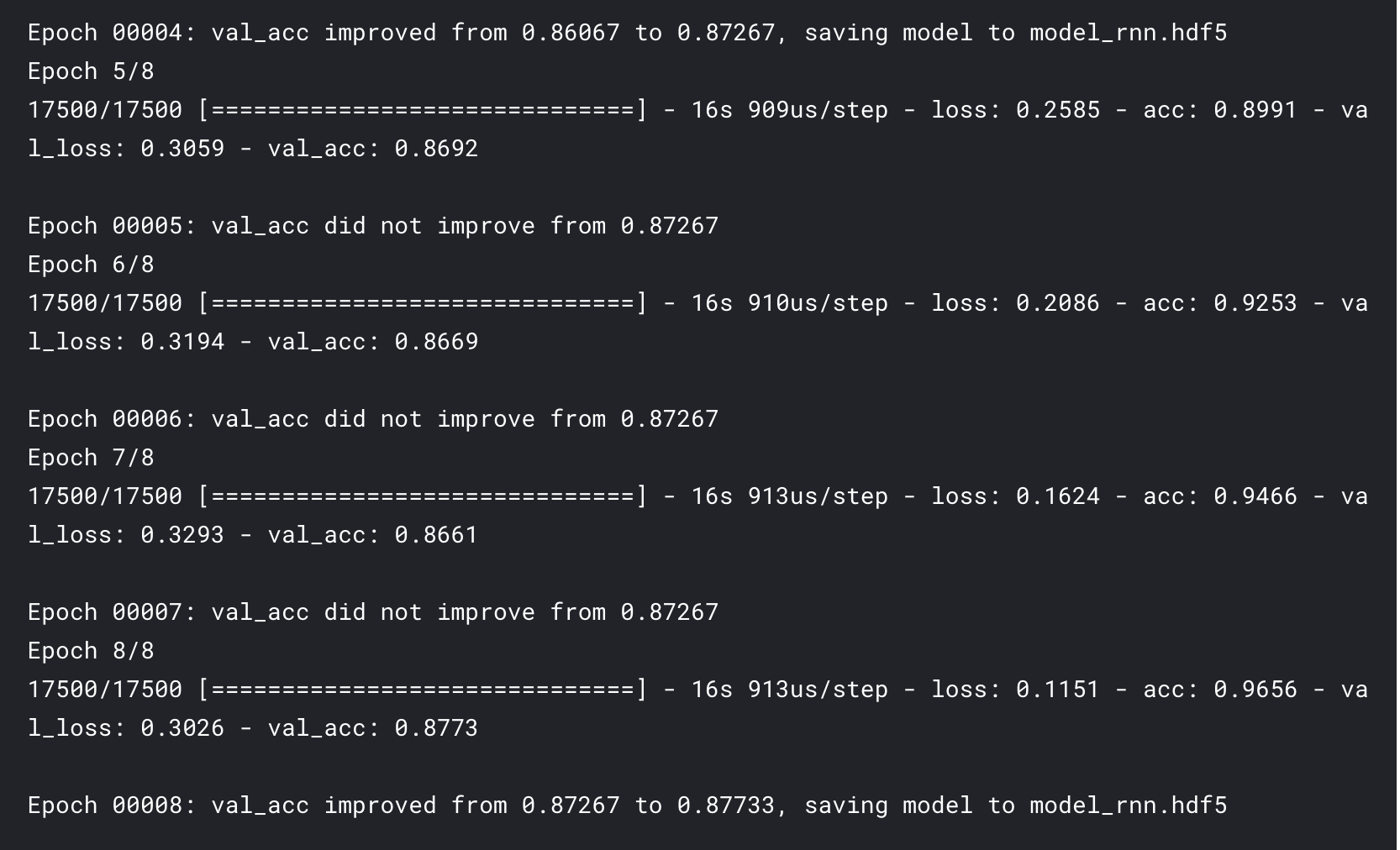
**9. Time/Epoch (min.) bar graphs**

The time per epoch bar graphs are attached as follows:

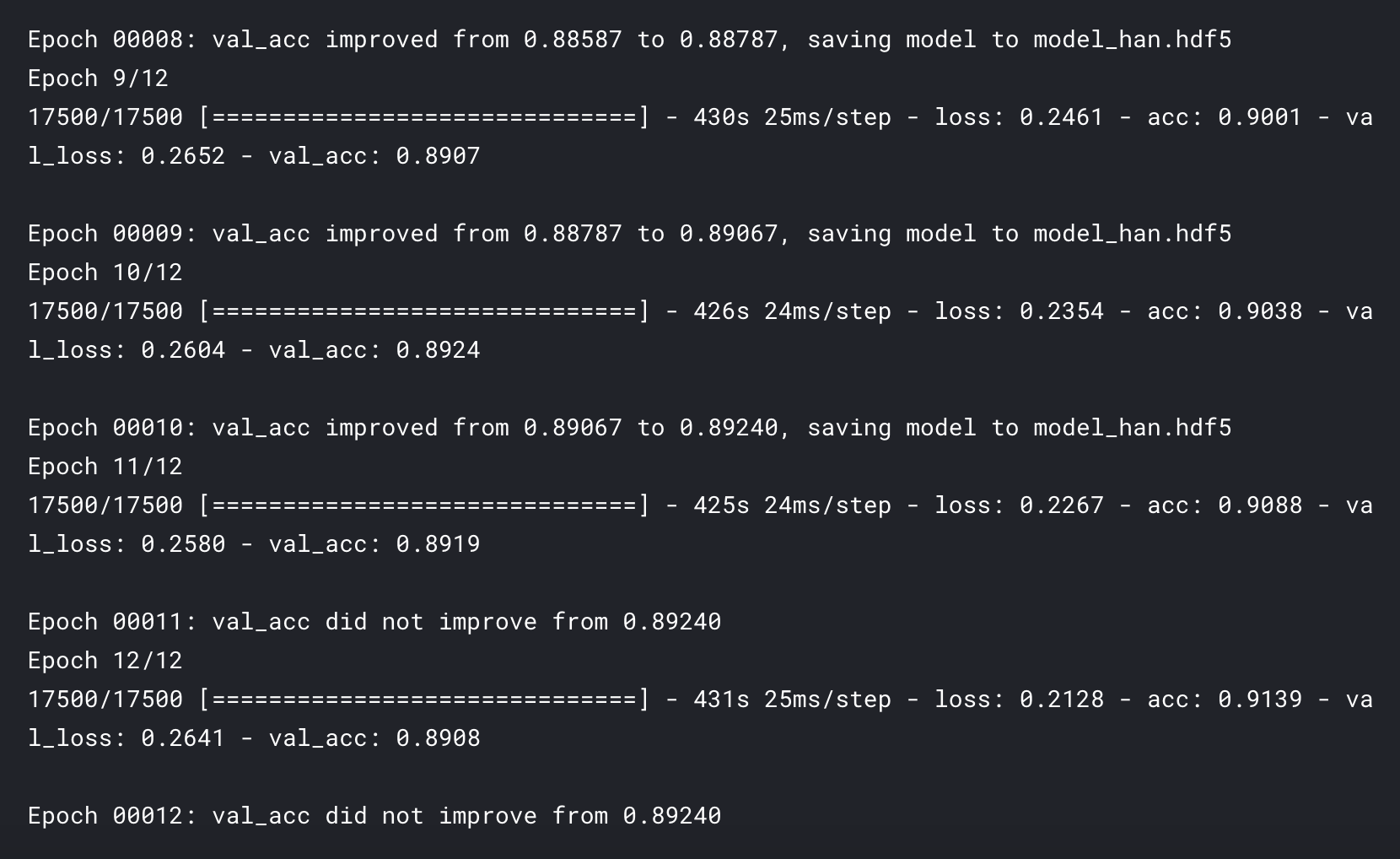
1. CNN:



1. RNN:



1. HAN:



Thus, helping us conclude that, HAN takes the longest time to train followed by RNN and finally the CNN.

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

1. We observed that using adam optimizer resulted in fast convergence when compared to SGD and RMSProp.
2. Shallow models underfit the training data. Very deep models increases the performance only until some point, after which it runs the risk of overfitting the training data.
3. Training time of CNNs is fast as compared to RNN and HAN.
4. Very small and very large batch sizes take longer for model training.
5. LSTMs and GRUs show indistinguishable performance.