Homework 7 Report

Problem Description:

The objective of the homework is to build a model for classifying the IMDB Movie reviews, this problem falls into category of sentimental classification which is a subset of natural language processing. Each review consists of varying length and it can be either positive or negative review. The dataset consists of 25000 training set and 25000 testing set. This input data is sequential and thus better neural network for solving this set of problems are Recurrent Neural Network. The requirement of the homework is to build model with architectures Long short-term memory (LSTM), Gated recurrent unit (GRU) and conv1D.

Deep learning Framework:

The choice of the framework was open, and I used Tensorflow Keras to implement and build all the three architectures.

Dataset Loading and Analysis:

Keras has an inbuilt imdb dataset available for using and code snippet was provided with homework on how to load the dataset in a configuration. The key parameters in this configuration where the *vocab_size* and *maxLen*. The value of *vocab_size* specifies the count of the most frequent words to be included in our case 8000 whereas *maxLen* is used to filter out the sequences above this limit from the dataset. The data is in the form of an array of integers which represent the index of word in the vocabulary dictionary. To analysis the data I have used *imdb.get_word_index()* method to see the word corresponding to index. After the data is loaded, we use padding to ensure each sequence is of the same length. The label of the dataset is binary, 0 representing a negative review and 1 representing a positive review. So, it can be further categorized as binary sentimental classification problem.

Architectures:

I have used LSTM, GRU and Conv1D to solve the binary sentimental classification, but common and important layer among all the layers is the Embedding layer. The following is a brief description on the word embedding and later section covers the overview of the architecture with experimental analysis with results.

Word Embedding:

Word embeddings are vector representation of a word. Words with similar meaning will have vector representations that are close together in the embedding space. Word embedding tries to capture some sort of relationship in meaning, morphology, context, or some other kind of relationship. By encoding word embeddings, we represent words numerically in a way that captures them in vectors that have tens or hundreds of dimensions instead of millions (like one-hot encoded vectors). Word embedding is an efficient, dense representation in which similar words have a similar encoding. The values of the embeddings are trainable parameters. Depending on the dataset size the dimensionality of the embedding is chosen, and it is 128 in this case.

The need of LSTM and GRU architecture is to overcome the exploding and vanishing gradient problem which occurs in vanilla RNN.

Training of Architectures:

The problem is a binary classification, so I have used binary cross entropy loss to calculate the loss function. RMSProp and Adam are most preferred learning algorithms thus I opted for Adam for training the models. The number of epochs and other hyperparameter were tuning by trail and error to obtain the desired accuracy

LSTM:

An LSTM network is a recurrent neural network that has LSTM cell blocks instead of regular neural network layers. These cells have different components called the input gate, the forget gate and the output gate as shown in Fig 1. The input x_t is concatenated to the previous output from the cell h_{t-1}. The first input is sent to the tanh layer. The second the input is passed to an input gate which is sigmoid layer acting as “kill off” of elements which are not required and the weights if the layer can be trained to turn off certain input values. The next step in the flow of data through this cell is the forget gate loop. LSTM cells have an internal state variable s_t that is added to the input data to create an effective layer of recurrence. Instead of a multiplication operation an addition operation is used avoiding the vanishing gradients. The recurrence loop is controlled by a forget gate that helps the network learn which state variables should be “remembered” or “forgotten”. Finally, we have an output layer tanh which is controlled by an output gate. This gate determines which values are allowed as an output from the cell.

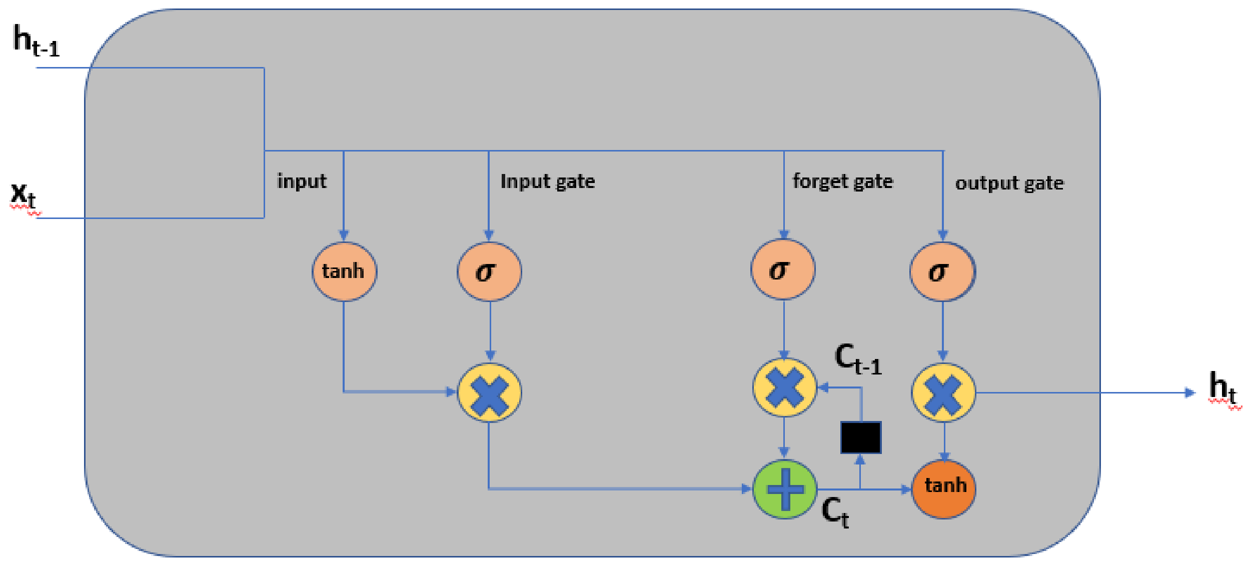

Figure 1: LSTM Cell

Building of LSTM network:

Steps I followed while building the model:

- Used a simple LSTM layer with varied dimension and epochs greater 30, but accuracy was halting around 71%.
- I tried multilayer LSTM with several configuration but was not able to achieve 85% accuracy.
- After some trail and error I started using Bidirectional LSTM and accuracy started increasing at lower range of epochs itself. So replaced simple LSTM with Bidirectional LSTM and varied parameters like no of cells in the layer, epochs etc.
- The single layer Bidirectional LSTM was followed by a Dense fully connected layer with relu activation functions and final layer with sigmoid activation as output needed to be binary.
- I used Dropout before the output layer to decrease the risk of overfitting.
- I even tried using multiple layers of Bidirectional LSTM, but did not find much improvement

Conclusions on LSTM:

Bidirectional LSTM was able to convert to desired accuracy in a less number of epochs compared to simple LSTM. But, the number of parameters to be trained are more in Bidirectional LSTM compared to simple LSTM.The addition of GlobalAverageMaxPooling and Dropout could stop overfitting to an extent.

GRU:

The GRU has a slightly different architecture where it combines both the forget and input gate into a single gate called the update gate. It merges the cell state and hidden state along with few changes which results in an architecture as shown in Fig 2. GRU's gets rid of the cell state and uses the hidden state to transfer information

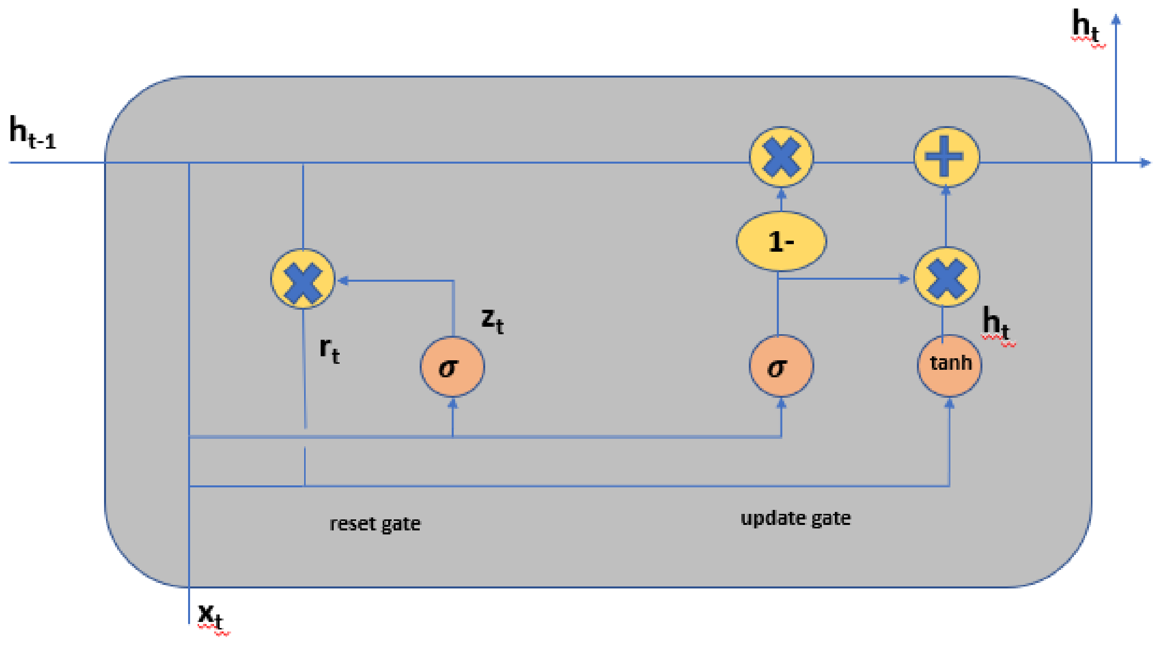

Figure 2: GRU Cell

Building of GRU network:

Steps I followed while building the model:

- With the intuition of the LSTM model, I used a Bidirectional GRU with 32 cells to start and without any Max pooling or dropout.
- I used same number of epochs as LSTM architecture i.e 10 and the accuracy of testing was around 89%

Conclusions on GRU:

The number of parameters to be trained for GRU are less compared to LSTM and the training took less time compared to LSTM. Since IMDB is not very complex dataset GRU was able to perform better than LSTM.

Conv1D:

Convolution operations are generally known for image classification and computer vision problems. But, Conv1D where convolution is carried only in one direction is suitable for Natural Language processing especially text classification. Conv1D moves only in one direction and the basic operation of convolution can help in identifying the relation between the words. Since the input is converted into embedding and a Conv1D can operate on the embedding layers and extract set of features that represent such relationship.

Building of Conv1D network:

Steps I followed while building the model:

- Started with a single convolutional 1D layer with 32 channels and kernel with 3 and increased the channels gradually to increase them to 128 channels and kernel 5.
- I added Global Average Pooling 1D as max pooling for convolution layer
- I had to try different dense layer configuration and was able to find out 64 being the suitable for my architecture
- To avoid risk of overfitting dropout layer is added before the output layer

Conclusions on Conv1D:

Conv1D may have a greater number of parameter but was able to train very quickly compared to other two networks. Since convolution is simple operation compared to LSTM and GRU time would be less. Training and Testing accuracy are comparable to GRU and LSTM.

Pros and Cons of Architecture:

LSTM	**GRU**	**Conv1D**
Pros		
Avoids Gradient Vanishing	Avoids Gradient Vanishing	Easier to train
Avoids Exploding Gradient	Avoids Exploding Gradient	It is suitable for short and fixed sequences
Works well with longer sequences	Simpler compared to LSTM	Model size is less
Can work with complex dataset		Less memory to train

	Less memory compared to LSTM	
Cons		
LSTM takes a longer to train	May not be as good as LSTM with complex dataset	Only for small sequences of data
More memory to train	More time to train compared to conv1D	Not suitable for all kind of Natural language processing
Dropout is much harder to implement	Training time is more compared conv1D	
Easier to overfit		

Results:

Please find the results in attached PPT.