Table of Contents

Table of figures

Section 1: Introduction

Be it the case of how Obama's administration gained insight from public opinion before 2012 presidential elections[1] or how Expedia Canada gained insight from the negative feedback[2] for their music given for a television ad, opinion mining or **Sentiment Analysis[3]** has always proved to be quite an effective tool. Over the recent years, sentimental analysis by social media sites, brands and companies has increased by many fold, as sentiment analysis gives valuable insight into the public opinion about relevant topics. Basically, Sentiment Analysis is finding the real emotion that exists behind written words. It is a useful tool to understand of the attitudes from opinions. It enables users to strategize and plan for the future in a better way. Both natural language processing (NLP)[4] and machine learning methods[5] have significance in sentiment analysis. NLP is obviously complex as human language is complex and with the evolution of language, the dictionary that machines use to comprehend sentiment is continuously expanding. It becomes an even more challenging task to teach machine to analyse spelling errors, grammatical errors and how context can change the tone. The way a linguist specialist teaches a machine to perform sentiment analysis is not very different from how young Jon Connor taught the T-800 the famous phrase "Aasta la vista, baby" in Terminator 2[6].Human can reasonably interpret the tone of a

[1]Bannister, Kristian. "Sentiment Analysis: How Does It Work? Why Should We Use It?" *Brandwatch*, Brandwatch, 26 Feb. 2018, www.brandwatch.com/blog/understanding-sentiment-analysis/

[2]"Sentiment Analysis: Nearly Everything You Need to Know." *MonkeyLearn*, 20 June 2018, monkeylearn.com/sentiment-analysis/

[3]Agarwal, Apoorv & Xie, Boyi & Vovsha, Ilia & Rambow, Owen & Passonneau, Rebecca. (2011). Sentiment Analysis of Twitter Data. Proceedings of the Workshop on Languages in Social Media

[4]Seif, George. "An Easy Introduction to Natural Language Processing." Towards Data Science, Towards Data Science, 2 Oct. 2018, towardsdatascience.com/an-easy-introduction-to-natural-language-processing-b1e2801291c1.

[5]"Sentiment Classification Using Machine Learning Techniques." International Journal of Science and Research (IJSR), vol. 5, no. 4, 2016, pp. 819–821., doi:10.21275/v5i4.nov162724.

[6]Bannister, Kristian. "Sentiment Analysis: How Does It Work? Why Should We Use It?" *Brandwatch*, Brandwatch, 26 Feb. 2018, www.brandwatch.com/blog/understanding-sentiment-analysis/

sentence or a paragraph based on intuition but training machines for similar interpretation is difficult as it might involve various emotions like positive, negative, sarcasm etc. With advancement of machine learning, **Neural Networks** have proved to make very accurate predictions in sentiment analysis is in comparison to traditional bag of words[7] model. Neural networks are all about giving the computer some data, and letting it figure out how to utilize the data to develop models to represent complex tasks with high accuracy – like analyzing twitter tweets for its sentiment. Neural networks are inspired from biological neurons. Neural Networks are generally used to model the problem and prediction becomes a task of optimizing weights to minimize a cost function. To make it more computationally inexpensive the "biological" neuron is modeled as a unit which assigns differing weights to each input and takes weighted average of it. For NLP each word is mapped to a vector defined by its features (which in turn relate to the word's surrounding context), and neural networks can be used to learn which features maximize a word vector's score.

Convolutional Neural Networks[8] (CNN) and Recurrent Neural Networks[9] (RNN) are two prominent types of neural networks which have been used to solve wide range of problems related sentiment analysis. These networks are fundamentally different, CNN takes whole input at first stage whereas RNN process one portion of the input at a time. For example, if a sentence is represented as words, convolutional network will take the sentence in the first stage and recurrent neural network will process each word at a time.

[7] BarryArtificial Intelligence and Cognitive Science , James. "Sentiment Analysis of Online Reviews Using Bag-of-Words and LSTM Approaches." *Artificial Intelligence and Cognitive Science 2017*, 2017, ceur-ws.org/Vol-2086/AICS2017_paper_21.pdf,print

[8] Pokharna, Harsh. "The Best Explanation of Convolutional Neural Networks on the Internet!" *Medium.com*, Medium, 28 July 2016, medium.com/technologymadeeasy/the-best-explanation-of-convolutional-neural-networks-on-the-internet-fbb8b1ad5df8.

[9] Banerjee, Suvro. "An Introduction to Recurrent Neural Networks – Explore Artificial Intelligence – Medium." *Medium.com*, Medium, 23 May 2018, medium.com/explore-artificial-intelligence/an-introduction-to-recurrent-neural-networks-72c97bf0912.

Long short term memory[10] (LSTM) is a type of RNN, whichcan learn relation between inputs which are spaced in excess of 1000 steps.

This essay compares neural network architectures based on LSTM and CNN for sentiment analysis, with a research question **"To what extent convoluted neural network (CNN) does better sentiment analysis than long short-term memory (LSTM)?"**

The worthiness of the study will not be only limited to IMDB and similar datasets rather it is being used by big firms and political parties to obtain information from the users and community to improvise on the basis of the sentiment of its users.

Section 1.1: Plan of Investigation

For developing the architecture of the neural networks, **Keras** (Python3) deep learning framework will be used. All the neural networks mentioned in this essay will be trained and evaluated using **TensorFlow**.

IMDB Dataset will be used: It is composed of 25,000 polar positive and negative movie reviews so has wide distribution of length of reviews.

Step 1: Optimizing both neural networks

Step2: Comparing them on the basis of the validation accuracy and validation loss

Step3: The neural networks will be compared on the basis of

1. **Prediction accuracy**
    - Dependency on length of review
    - dropout rate
    - learning rate

[10] Olah, Christopher. "Understanding LSTM Networks." *Understanding LSTM Networks -- Colah's Blog*, 27 Aug. 2015, colah.github.io/posts/2015-08-Understanding-LSTMs/.

2. CPU time

Section 2: Background

Section 2.1: Neural Network

To adequately describe neural networks along with CNN and LSTM this section walks through basic elements of neural networks and basics of LSTM and RNN.

Section 2.1.1: Functioning of individual neuron

A neuron can be modelled as a function f which takes weighted sumof the input I with k dimensions and produces output o.

$$f(I) = \sum_{i=0}^{k} I_i * w_k = o$$

Where w is the weight matrix having same dimensions as I.

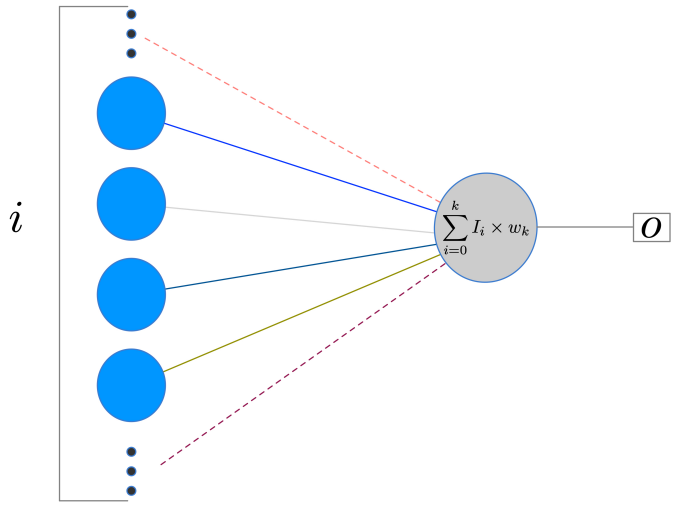

Figure 1 : Basic neuron accepting input

Section 2.1.2: Activation Functions

Activation functions calculates the weighted sum to decide which neuron should be activated. They are applied on output of each neuron. The purpose of activation functions is to establish non-linearity in a neural network, which is **essential for any nonlinear problem**[11].If the activation function is linear than no matter how many layers are present in the neural network, it will behave like a single perceptron only. Hence, non linear activation functions are used in the neural networks. In this investigation, sigmoid, rectified linear unit (ReLU), and tanh non linear activation functions are used in the neural networks.

Convention: y is output and x is input.

ReLUis defined as:

$$y = \max(0, x)$$

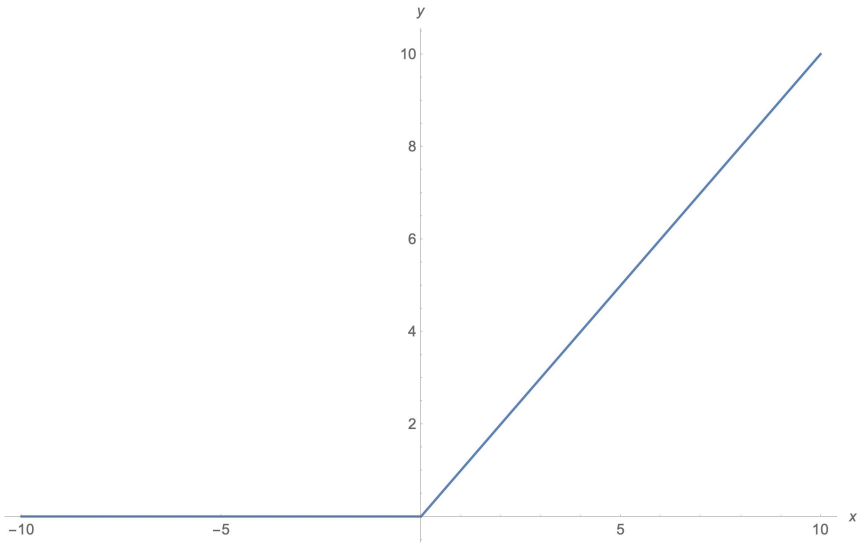

Figure 2 : ReLU

Gradient of ReLU is 1 when $x > 0$ and otherwise.

[11] Walia, Anish Singh. "Activation Functions and It's Types-Which Is Better?" *Towards Data Science*, Towards Data Science, 29 May 2017, towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f.

doug. "Why Must a Nonlinear Activation Function Be Used in a Backpropagation Neural Network?" *Stack Overflow*, 20 Mar. 2012, stackoverflow.com/questions/9782071/why-must-a-nonlinear-activation-function-be-used-in-a-backpropagation-neural-net.

Sigmoid(σ) is defined as:

$$y = \frac{1}{1 + e^{-x}}$$

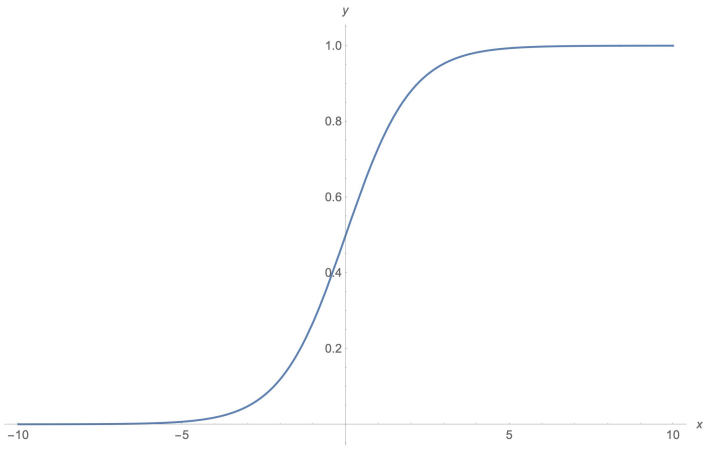

Figure 3 : Sigmoid

Since the value of the function vairs between 0 and 1.

Tanh function is used to map values to range of -1 to 1.

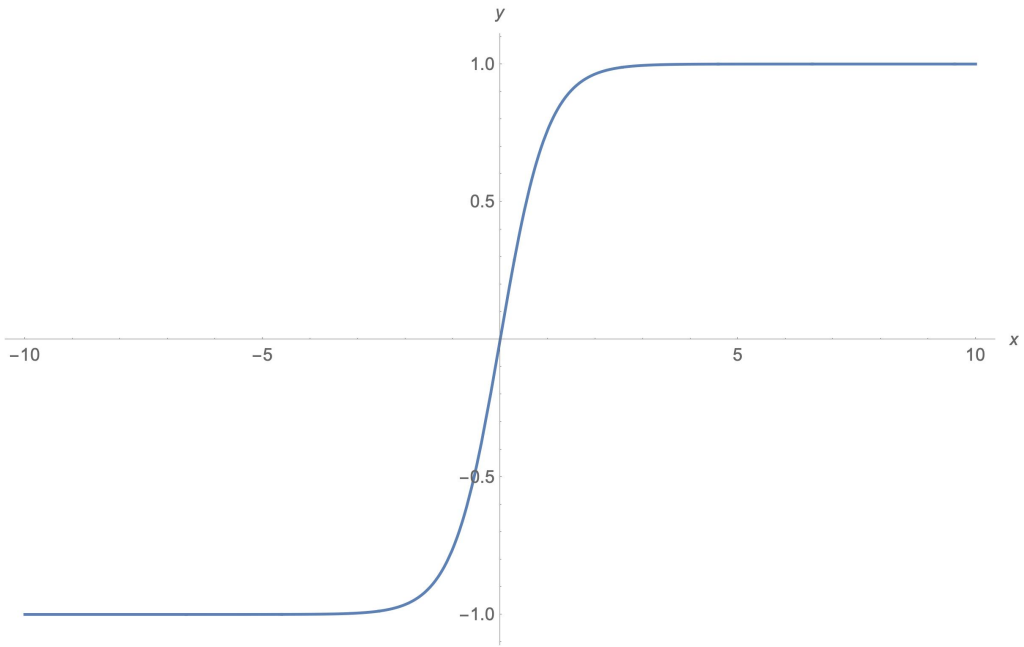

Figure 4 : tanh

Section 2.1.3: Back Propagation

Role of most neural networks is to model the input to the prediction. This is achieved by finding direction along which the neural network should step to reduce error. Back

propagation[12] algorithm defines how the weights for each neuron in the network should change to minimize the error.

The goal of the algorithm is to compute $\frac{\delta \mathrm{L}}{\delta w}$ where L is the loss and w is the weight which will be adjusted to minimize the loss L.

Section 2.1.4: Regularization

In most cases the number of parameters in neural network is a lot greater than the number of training examples. It leads to overfitting of the network to fit to the training data. In this investigation early stopping is used where the neural network is stopped after the validation loss stops to decrease. Dropout[13] is also used, which zeros out output of a layer randomly preventing co adaptation resulting in extraction of more features.

Section 2.1.5: Loss function and optimizers

Loss function[14] is defined for each neural network which compares the predictions of the model with the actual labels. The loss is then used in back propagation to train the neural network. In the neural networks used in the investigation, binary cross entropy[15] is used. This is given by: $-label * \log(pred) + (1 - label) * \log(1 - pred)$

Where $label$ is actual binary label and $pred$ represents the prediction of the model which is bounded by 0 and 1.This loss will push the model such that the prediction is pushed to 0 and 1 for label being 0 and 1 respectively.

---

[12] Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. "Learning Representations by Back-Propagating Errors."*Nature*, vol. 323, no. 6088, 1986, pp. 533-536. *CrossRef*, http://dx.doi.org/10.1038/323533a0, doi:10.1038/323533a0.

[13] Srivastava , Nitish. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." *Journal of Machine Learning Research*, 2014, jmlr.org/papers/volume15/srivastava14a.old/srivastava14a.pdf.

[14] Hao, Zhang. "Loss Functions in Neural Networks." *Isaac Changhau*, 6 June 2017, isaacchanghau.github.io/post/loss_functions/.

[15] Godoy, Daniel. "Understanding Binary Cross-Entropy / Log Loss: a Visual Explanation." *Towards Data Science*, Towards Data Science, 21 Nov. 2018, towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a.

Gradient descent is an optimization algorithm in machine learning used to find the value of parameters of function that minimizes its cost function (measurement of howwellthe neural network performs with respect to its training samples and its expected output) as much as possible. Since the training dataset consists of 20,000 samples, the gradient descent takes too long since in every iteration it goes through the complete training dataset to update the value of a parameter[16]. On the other hand, Stochastic gradient descent[17] uses only a subset of the training dataset to update value of a parameter.SGD only uses information in one-time step. In SGD, weights are updated by amount directly proportional to the gradient of the loss with respect to the weight. ADAM optimizer is basically an extension to SGD. It can be seen as combination of SGD with momentum and root mean square propagation. It[18] adjusts the learning rate for each parameter separately and is empirically faster for convergence. In this investigation, ADAM optimizer is used.

Section 2.2: Convolutional Neural Network (CNN)

Convolutional neural networks are feed forward networks. They consist of neurons with biases and learnable weights. In the network, each neuron is receiving several inputs which then take weighted sum over them. The calculated weighted sum is then passed through an activation function and produces an output. They use convolutions to extract features over the input data.

Convolution of function f with function g is given by:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau) d\tau$$

[16] Nevrekar, Akshay. "What Is the Difference between Gradient Descent and Stochastic Gradient Descent?" *Data Science Stack Exchange*, 4 Aug. 2018, datascience.stackexchange.com/questions/36450/what-is-the-difference-between-gradient-descent-and-stochastic-gradient-descent.

[17] "Optimization: Stochastic Gradient Descent." *Unsupervised Feature Learning and Deep Learning Tutorial*, deeplearning.stanford.edu/tutorial/supervised/OptimizationStochasticGradientDescent/.

[18] Nikhil, Nishant. "Everything You Need to Know about Adam Optimizer – Nishant Nikhil – Medium." Medium.com, Medium, 27 Jan. 2017, medium.com/@nishantnikhil/adam-optimizer-notes-ddac4fd7218.

If g and f are discrete then they can be represented as matrices, and the resultant formula is:

$$\sum_{m=-\infty}^{\infty} f[m]\, g[n-m]$$

where f represents the input matrix and g represents the weight matrix (kernel).

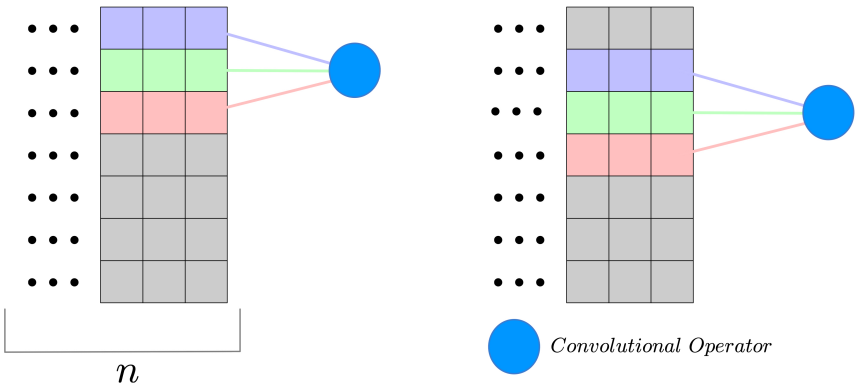

Figure 5: 1D discrete convolution with kernel size 3 and no padding

Figure 5 shows convolution where kernel size has 3 elements, resulting in only three weights being multiplied with respective inputs. The same weights will be used for the whole layer of output. Due to same kernel being used for the whole input number of parameters required is very small as compared to perceptron layer, and it can be seen as applying a filter over the entire input.

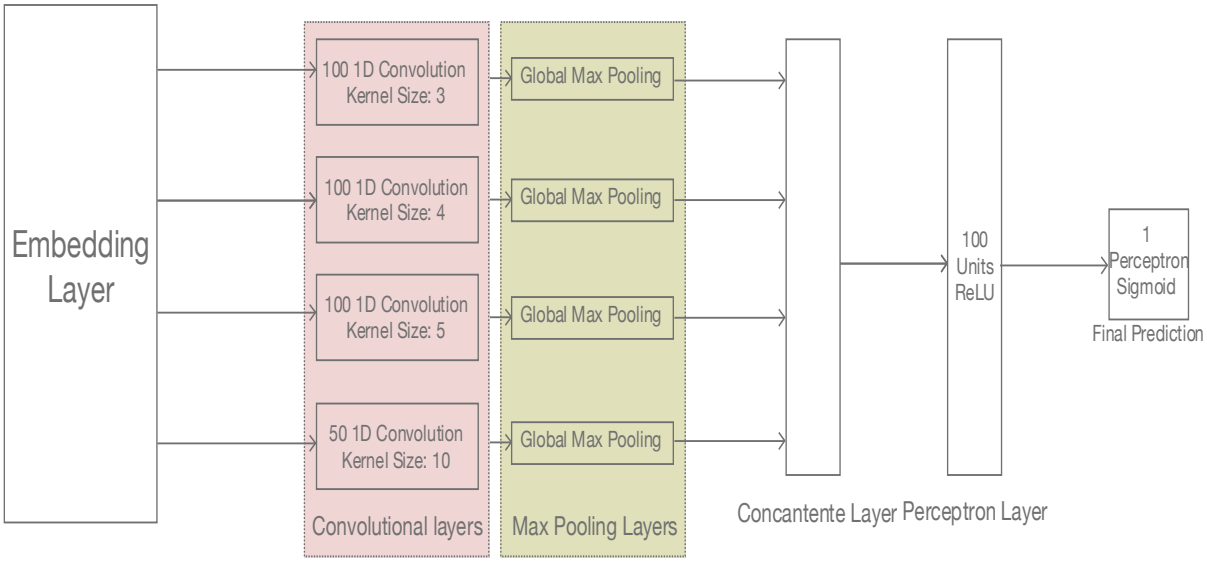

Figure 6: CNN architecture used in the essay

The CNN architecture used in this investigation, is inspired by Collobert et al. and Kim et al., where convolutions are used to extract feature from sentence.

Convolution with $kernel\ size\ =\ m$,will consider total of m words and extract features amongst them.

- The maximum of a convolutional kernel applied to a whole layer is taken which represents the highest value obtained by the application of the kernel on the entire input.
- The concatenated vector represents the maximum value found by convolving a number of different convolution kernels with the input.
- The concatenated vector is passed to perceptron layer having 100 perceptrons with ReLU activation function.
- The output generated by the perceptron layer is transferred to a perceptron with sigmoid activation function representing probability of the review being positive.

Global max pooling is used, which is an ordinary max pooling layer with pool size equal to the size of the input. The aim of the max pooling layer is to identify the most important

features by down sampling the input representation, reducing its dimensionality and allowing for the assumption to be made about features contained in sub regions binned.

It prevents the network from over-fitting and also reduces the computational cost.

Section 2.3: Long Short Term Memory (LSTM)

LSTM is a type of RNN. The LSTM module is defined as follows:

Two hidden inputs are propagated between the modules, referred here as C_t and h_t for cell t ($h_t$can also be used as output for each cell).

$$x_t: Input\ for\ cell\ t$$

$$b_i: Bias\ for\ operation\ i$$

$$W_i: Bias\ for\ operation\ i$$

Operations in LSTM can be divided in three sequential steps:

1. Forgetting previous cell state(F):

$$F_t = \sigma(W_f.[h_{t-1}, x_t] + b_f)$$

$$C_t = C_{t-1} \circ F_t$$

Sigmoid function ensures that the weight is between 0 and 1.

2. Adding information from current input (I):

$$I_t = \sigma(W_I.[h_{t-1}, x_t] + b_I) \circ \tanh(W_C.[h_{t-1}, x_t] + b_c)$$

$$C_t = C_t + I_t$$

Combination of sigmoid and tanh ensures that value added to C_t is bounded by -1 and1.

3. Producing output vector (O):

$$h_t = \tanh C_t \circ \sigma(W_O[h_{t-1}, x_t] + b_o)$$

Combination of sigmoid and tanh ensures that $h_t$is bounded by -1 and 1.

For this investigation, the LSTM architecture is composed of LSTM units with each unit having 100-dimensionalhidden state vector. Output from LSTM is fed into a sigmoid perceptron producing the probability of the review being positive.

Section 3: Investigation

Section 3.1: Methodology

Keras is used in the primary investigation.

```
from google.colab import drive
drive.mount('/content/drive')
import numpy as np
import json
from tqdm import tqdm
from random import shuffle
import io
import os
from keras.models import Sequential
import pickle
from glob import glob
import logging
from time import time
import keras
from keras.layers import Dense, Conv1D, Dropout, Input, LSTM,concatenate,Reshape
,Flatten,GlobalMaxPool1D,MaxPooling1D,Embedding
from keras.layers.merge import Concatenate
from keras import optimizers
from keras.utils import Sequence
from keras.callbacks import Callback
import tensorflow as tf
import keras.backend as K
from keras.models import load_model
from keras import regularizers
from keras.datasets import imdb
from keras.preprocessing import sequence
```

Figure 7: Importing Dataset

As can be seen in the figure, the imdb dataset is directly imported from the keras datasets. There was no requirement to download from the source and then use it.

```
MAXLEN = 512
VALSIZE = 2000
TRSIZE = 20000
TOP_WORDS = 15000
```

Figure 8: Data Description

Then set the maximum length of reviews which will be used and the validation size and training size of the dataset was decided. The vocabulary size is fixed which is used in the neural network.

```
(x_tr, y_tr), (x_test, y_test) = imdb.load_data(num_words=TOP_WORDS)
```

Figure 9: Segregation

Now, the data is loaded from keras's dataset using an api and segregated into training and testing dataset.

To feed this data into the CNN and RNN, they review must have the same word count. So the review length is limited, (MAXLEN=512). The longer reviews will be truncated and shorter reviews will be padded with null value (0). This is achieved by pad_sequences() function in Keras.

The input given into the neural networks is a sequence of words and the output received from the neural network is a binary value of either 0 or 1.

```
def load_pickle(fname):
    if not os.path.exists(BASE_PATH + '/' + fname):
        logging.warning("Pickle file not present" + fname)
        exit()
    with open(BASE_PATH + '/' + fname, mode='rb') as fid:
        return pickle.load(fid)


def save_pickle(fname, dat):
    with open(BASE_PATH + '/' + fname, mode='wb') as fid:
        pickle.dump(dat, fid)


def get_file_array(dir_name, matcher):
    file_arr = glob(pathname="{}/{}".format(BASE_PATH + '/' + dir_name, matcher))
    return file_arr


def load_json(fname):
    if not os.path.exists(BASE_PATH + '/' + fname):
        logging.warning("Json file not present" + fname)
        exit()
    with open(BASE_PATH + '/' + fname, mode='rb') as fid:
        return json.load(fid)


def save_json(fname, dat):
    with open(BASE_PATH + '/' + fname, mode='w') as fid:
        json.dump(dat, fid)

# Utility to shuffle data
def shuffle_data(v_1,v_2):
    assert(len(v_1)==len(v_2))
    shuffle_order = np.random.choice(len(v_1),size=len(v_2),replace=False)
    return v_1[shuffle_order], v_2[shuffle_order]

# Testing Model
def test_model(f_name):
    model = load_model('{}/{}.model'.format(BASE_PATH,f_name))
    pred_labels = model.predict(test_dat)
    acc = (pred_labels.flatten().round() == test_labels.flatten()).mean()
    print('Accuracy is : ', acc)
    if not os.path.isdir(BASE_PATH + '/pred'):
      os.makedirs(BASE_PATH + '/pred')
    save_pickle(dat={'pred':pred_labels.flatten(),'acc':acc},fname='pred/' + f_name.split('/')[-1] + '.pred')
```

Figure 10: Screenshot of program

In the above lines of code, preprocessing is being done and json files are created which will be used in the neural networks. The data is also shuffled to make the results of neural network valid.

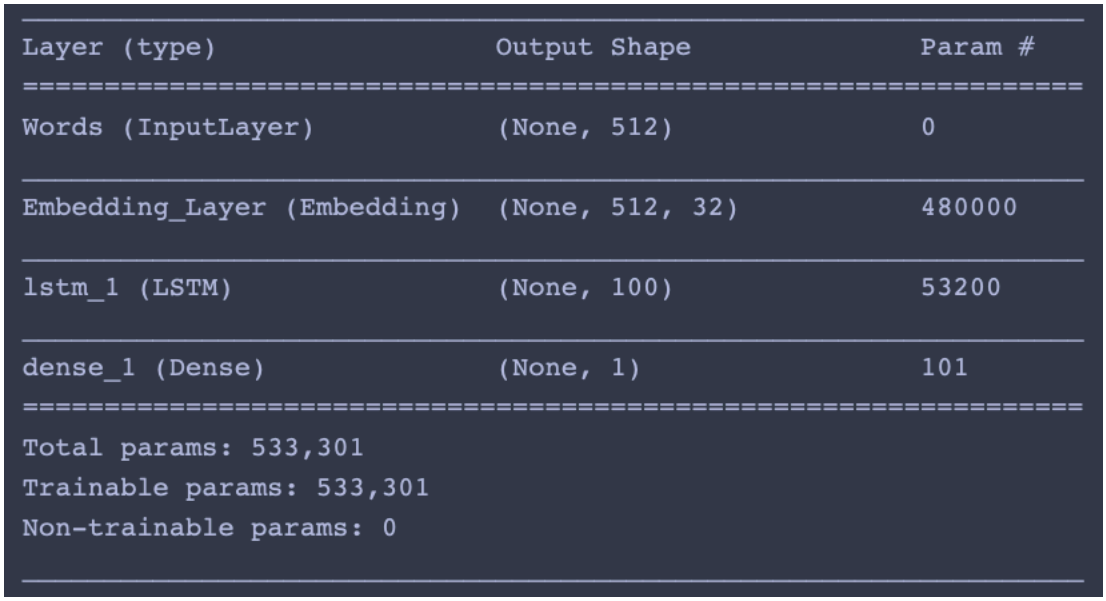

Figure 11: LSTM training

The LSTM model is simple with 1 imbedding, 1 LSTM and 1 dense layer. In total, 533,301in total need to be trained. This is only an example (parameters vary).

```
Layer (type)                 Output Shape              Param #     Connected to
==================================================================================================
Input_Sent (InputLayer)      (None, 512, 300)          0
__________________________________________________________________________________________________
C3 (Conv1D)                  (None, 512, 100)          90100       Input_Sent[0][0]
__________________________________________________________________________________________________
C4 (Conv1D)                  (None, 512, 100)          120100      Input_Sent[0][0]
__________________________________________________________________________________________________
C5 (Conv1D)                  (None, 512, 100)          150100      Input_Sent[0][0]
__________________________________________________________________________________________________
C10 (Conv1D)                 (None, 512, 50)           150050      Input_Sent[0][0]
__________________________________________________________________________________________________
global_max_pooling1d_1 (GlobalM (None, 100)            0           C3[0][0]
__________________________________________________________________________________________________
global_max_pooling1d_2 (GlobalM (None, 100)            0           C4[0][0]
__________________________________________________________________________________________________
global_max_pooling1d_3 (GlobalM (None, 100)            0           C5[0][0]
__________________________________________________________________________________________________
global_max_pooling1d_4 (GlobalM (None, 50)             0           C10[0][0]
__________________________________________________________________________________________________
Combining_Layers (Concatenate)  (None, 350)            0           global_max_pooling1d_1[0][0]
                                                                   global_max_pooling1d_2[0][0]
                                                                   global_max_pooling1d_3[0][0]
                                                                   global_max_pooling1d_4[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)              (None, 100)               35100       Combining_Layers[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)          (None, 100)               0           dense_1[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)              (None, 1)                 101         dropout_1[0][0]
==================================================================================================
Total params: 545,551
Trainable params: 545,551
Non-trainable params: 0
__________________________________________________________________________________________________
```

Figure 12: CNN training

The CNN model consist of 4 convolutional layers of varying kernel size and filters. These layers are followed by 4 1D max pooling layer and dense layer.

```
Epoch 1/2
32/32 [==============================] - 70s 2s/step - loss: 0.7222 - acc: 0.5040 - val_loss: 0.6959 - val_acc: 0.5060
Epoch 2/2
32/32 [==============================] - 71s 2s/step - loss: 0.7117 - acc: 0.5292 - val_loss: 0.6959 - val_acc: 0.5070
```

Figure 13: sample output

This is an example of the data generated by the neural network during its running.

Loss, acc, val_loss, val_acc will be used to producer graphs.

To make the graphs mathplotlib.pyplot is used.

Then the loss and accuracy is gathered corresponding to training and validation data.

Graph for varying learning rate and dropout rate for CNN and LSTM networks are generated for evaluation.

To generate the graph of review length vs accuracy is used, as given in fig.14

```
def make_plot(fname_arr,heading,out_fname):
    # Collect multiple in array
    x_axis = np.arange(0,540,50)
    plt_arr = np.concatenate(list(map(get_length_wise_acc,fname_arr)),axis=0)
    for c in range(plt_arr.shape[1]):
        for r in range(plt_arr.shape[0]):
            plt.scatter(x_axis[c],plt_arr[r,c],c='#a6c2f4')
    plt.xlabel('Review Length')
    plt.ylabel('Test Accuracy')
    plt.ylim(83,95)
    plt.title(heading)
    plt.savefig(out_fname,dpi=600)
```

Figure 14: accuracy on the basis of the length of the reviews

Section 3.3: Dataset

IMDB Movie Review dataset is used. In this investigation, the 15,000 most frequent words are taken into account.

In language some words are used much more often than others. In large volumes of text the frequency of use of a word decreases with its rareness in the concerned text in hyperbolic manner (Moreno-Sánchez et al). This was empirically observed by Zipf, and a slightly more generalized formulation is (Moreno-Sánchez et al):

$$n \propto \frac{1}{r^{\alpha}}$$

$$r: Rank\ (lower\ rank\ is\ higher\ occurance)$$

$$n: Frequency\ of\ use$$

$$\alpha: Exponential\ parameter\ (\sim 1)$$

Figure 15: Relation between Frequency of word and Number of words

In the dataset the frequency of word and number of words follow similar trend (Figure 15).

As seen in the histogram this would be vast majority of frequently occurring words in the dataset. All other words will be replaced by special token indicating unknown word. This will help in reducing number of parameters to learn.

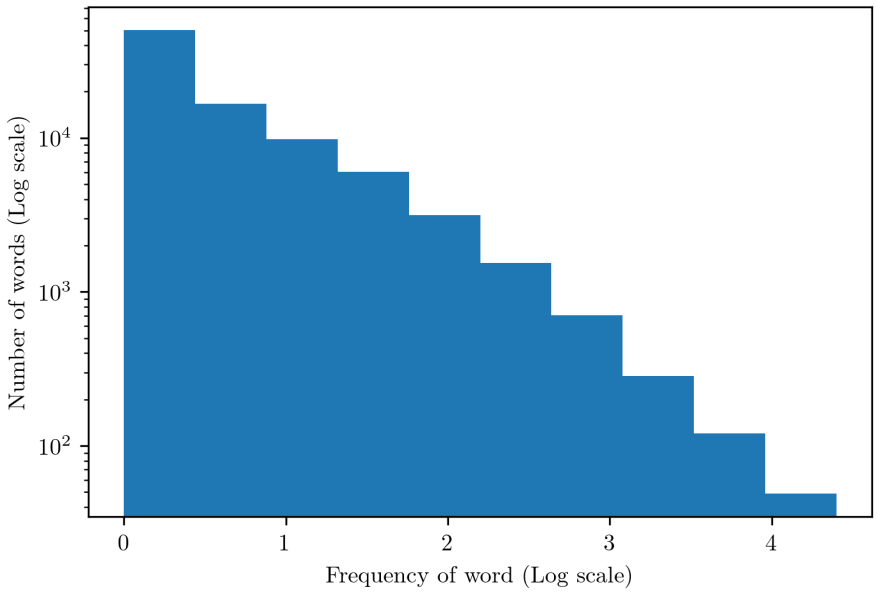

Figure 16: Frequency of words vs. number of words in train dataset

Section 3.3.1: Training

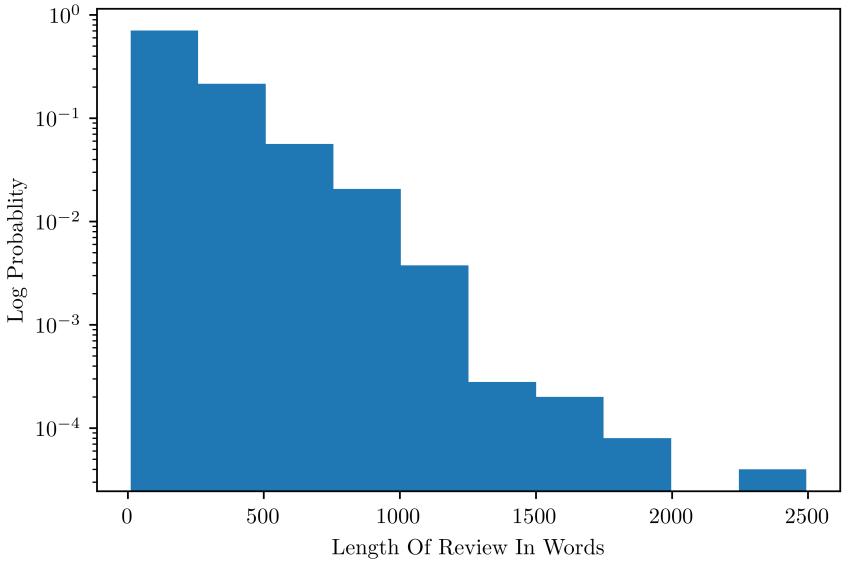

Figure 17 : Probability distribution of length of reviews in terms of words

The training dataset has vast majority of review with word count less than 512 words. Only reviews less than or equal to 512 words were used to reduce time taken for the network to train. As a result, ~12.6% of the movie reviews have been excluded. This helped in reducing training time of the neural networks.

IMDB data is provided separately for training and testing. The training data is shuffled and 20,000 samples are used for training and 2,000 samples are used for validation. The test set is comprised of 23,179 examples.

Section 4: Data Analysis

To decide which model is better than other before testing on the test data, the model where the validation loss is lowest is chosen.

Learning rate for the ADAM optimizer needs to be optimized. To find optimal learning rate for the ADAM optimizer, grid search was done for LSTM and CNN architecture from learning rate of 10^{-5} to 10^{-2} with step size equal to factor of 10.

The accuracy of the neural networks on the test data is evaluated and binned according to the number of words in the review. For binning the number of words in the review is rounded off to the nearest 50th word.

Number Of Words (Length)	Samples
0	32
50	1762
100	4097
150	6641
200	3850

250	2191
300	1621
350	1130
400	887
450	615
500	353

Table 1: Showing Length of words in sample

Table 1 shows Length of reviews rounded off to the nearest 50 and the corresponding number of samples in the IMDB test dataset.

Here the first bin to have very few samples, and accuracy fluctuations would be insignificant for that point.

Section 4.1: Neural network optimization

For the optimization of the neural networks, the dropout rate and the learning rate will be varied. In the course of training the neural networks, randomly neutrons are ignored. So, the impact of these 'dropped out' neurons on the forward pass is partially ignored and weight updates are not applied on the neuron on the backward pass.

As a result, the neural network dependence on specific weight is reduced and as a result it prevents the neutrons from developing co-dependency. Hence, the dropout significantly reduces overfitting. The learning rate is a hyper parameter which governs how much the weight of the network is adjusted in regard to the loss gradient. Very large learning rates can lead to skipping the minima region and small learning rate can lead to larger training time along with possibility of getting stuck in local minima. Hence, learning rate of the network needs to be tweaked for the best performance of the network.

The learning rate for both the neural networks will be varied from 10^{-1} to 10^{-5} with a decrement of 10^{-1}. The dropout rate for both the neural network will be varied from 0.1 to 0.4 with increment of 0.1. The optimizations of the neural networks will help in making the investigation worthier and its results will be more valid.

Section 4.1.1: CNN Optimization

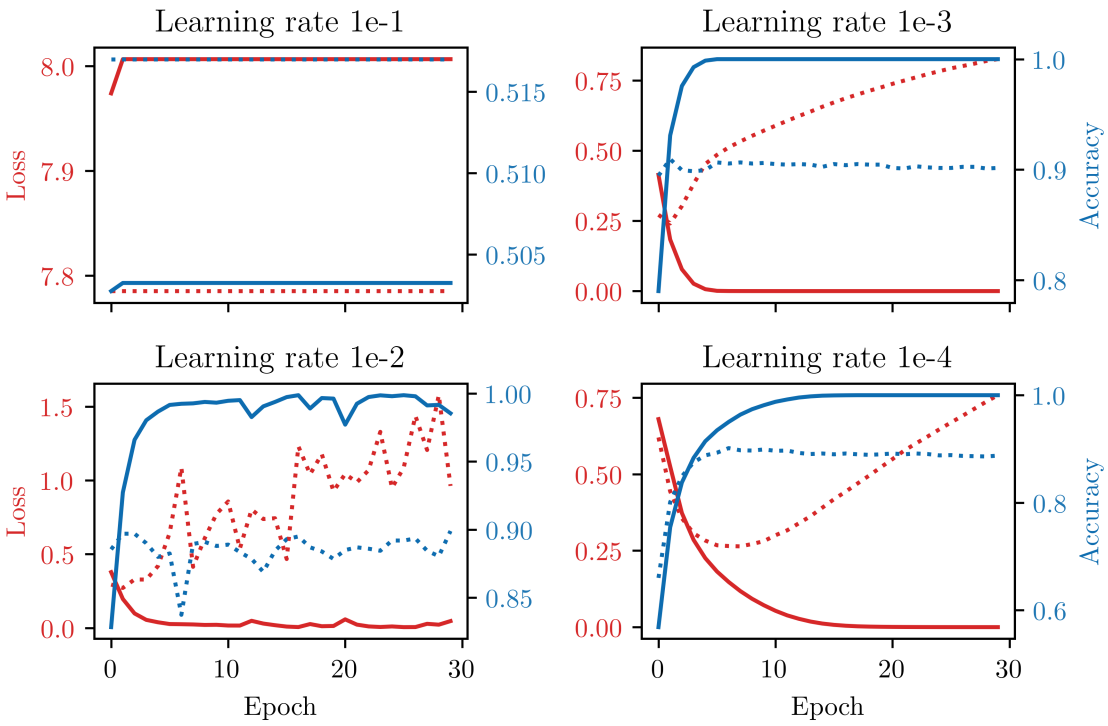

Figure 18 : CNN with varying learning rate

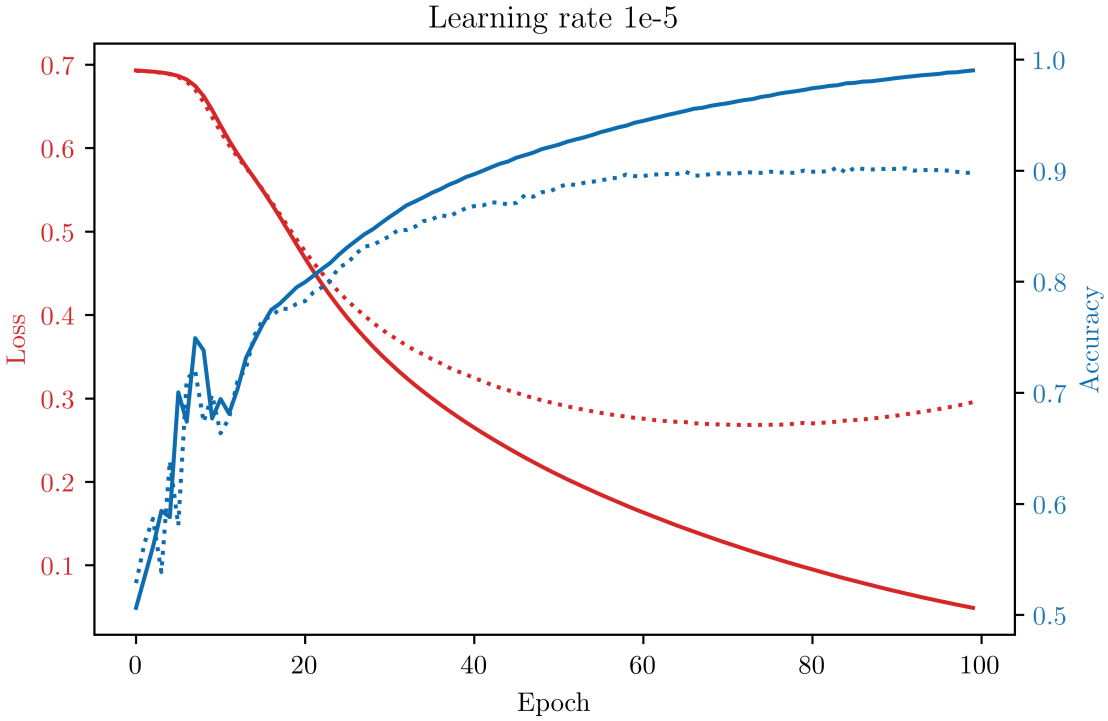

Figure 19 : CNN with e^{-5} learning rate

On running the CNN network with varying learning rate from 10^{-1} to 10^{-5} and produced data consisting of validation loss, validation accuracy, training loss and training accuracy of each epoch of the run. In figure 9 and 10, the training data is represented by the continuous line and the vandalization data is represented by the dotted line.

- Learning rate of 10^{-1} is too high for the CNN as it is not able to have significant improvement in accuracy as compared to models trained at lower learning rates. Validation accuracy at lowest validation loss is 89.7% for 10^{-2}, 90.95% for 10^{-3}, 90.15% for 10^{-4}, and 89.8% for 10^{-5} learning rate.
- Based on the result from the grid search for optimal learning rate 10^{-3} is used, which resulted in validation accuracy of 90.95%.
- Dropout was introduced between two sets of layers: from the concatenated layer to the perceptron layer, from the perceptron layer to the final perceptron.

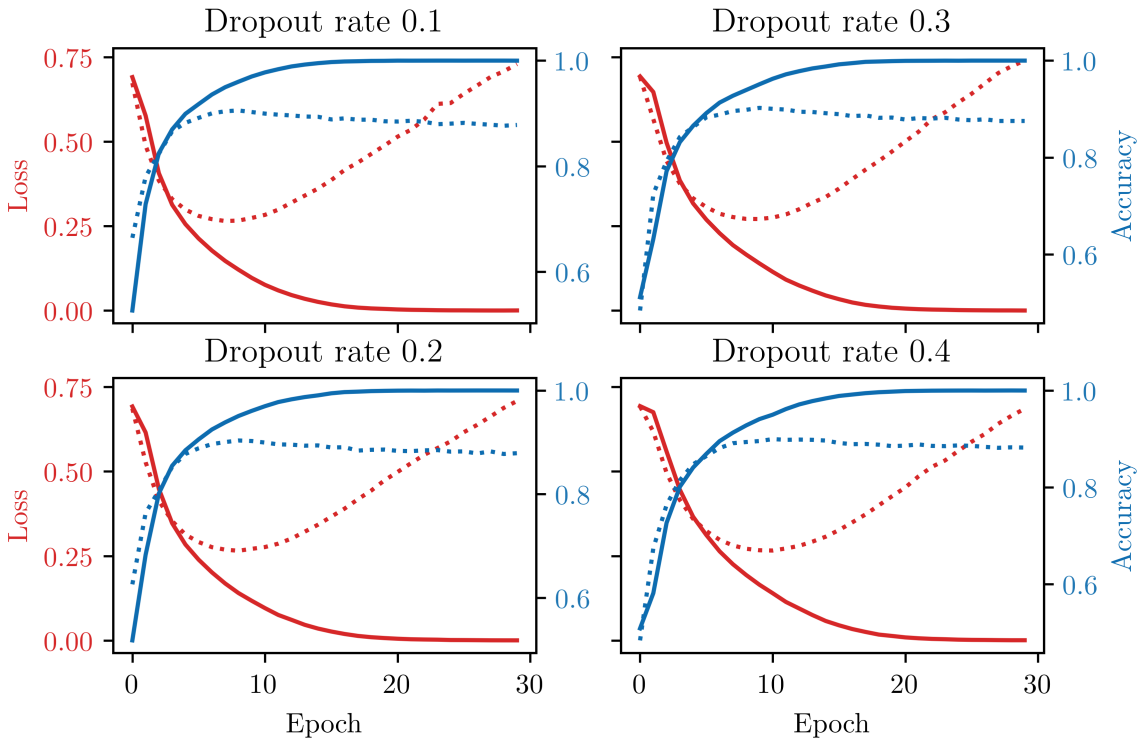

Figure 20: CNN with varying dropout rate (Folder: CNN_4_Dropouts).

The CNN network ran with varying dropout rate from 0 to 0.4 with increment of 0.1 and produced data consisting of validation loss, validation accuracy, training loss and training accuracy of each epoch of the run. In figure 11 and 12, the training data is represented by the continuous line and the vandalization data is represented by the dotted line.

Validation accuracies at minimal validation losses for the models was 89.95% for Dropout rate of 0.4, 90.25% for Dropout rate of 0.3, 90.35% for Dropout rate of 0.2and 90.55% for Dropout rate of 0.1. The model with 0.1 dropout rate has highest performance amongst all model. Dropout rate of 0.1 was chosen for the CNN.

To use context from wider range of words, 50 convolution filters with kernel size of 40 were added but did not result in increment of accuracy of the neural network on the test set.

The best CNN model was the initial model with learning rate of 10^{-3}, four models were trained. The average accuracy of the models overall was 88.97%.

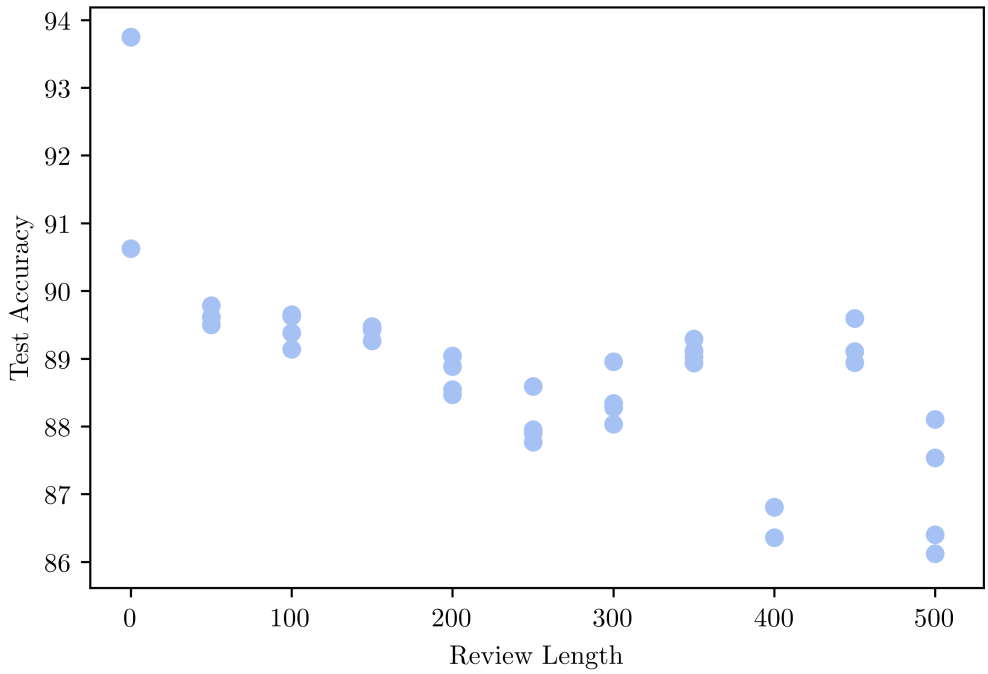

Figure 21: CNN test accuracy vs review length

In the above graph, the review length on the x axis is not specifically for that value. For example, for the points on the graph corresponding to 100 Review Length, reviews with ± 25 words are considered. The figure shows an overall general decrease in the test accuracy as the review length increases.

Section 4.1.2: LSTM Optimization

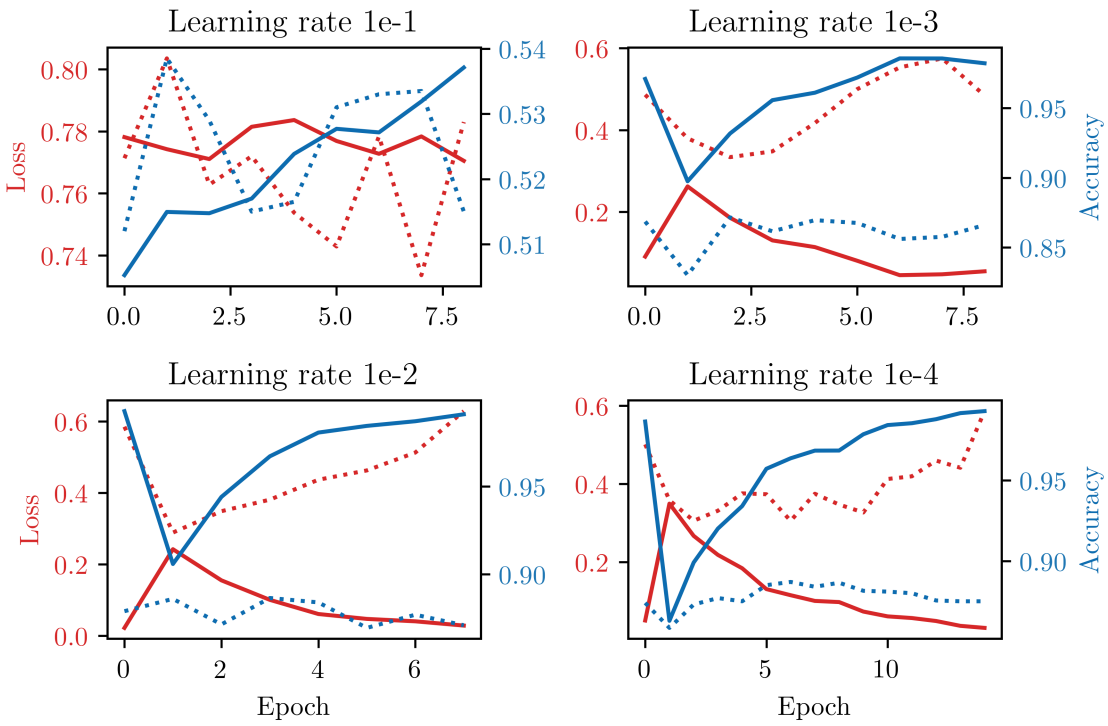

Figure 22: LSTM with varying learning rate

On running the LSTM network with varying learning rate from 10^{-1} to 10^{-4} and produced data consisting of validation loss, validation accuracy, training loss and training accuracy of each epoch of the run. In figure 13, the training data is represented by the continuous line and the vandalization data is represented by the dotted line.

Validation accuracy at lowest validation loss is 53.1% for 10^{-1}, 88.6% for 10^{-2}, 87.15% for 10^{-3}, and 88.7% for 10^{-4} learning rate.

Based on the result from the grid search for optimal learning rate 10^{-4} is used, which resulted in validation accuracy of 88.7%.

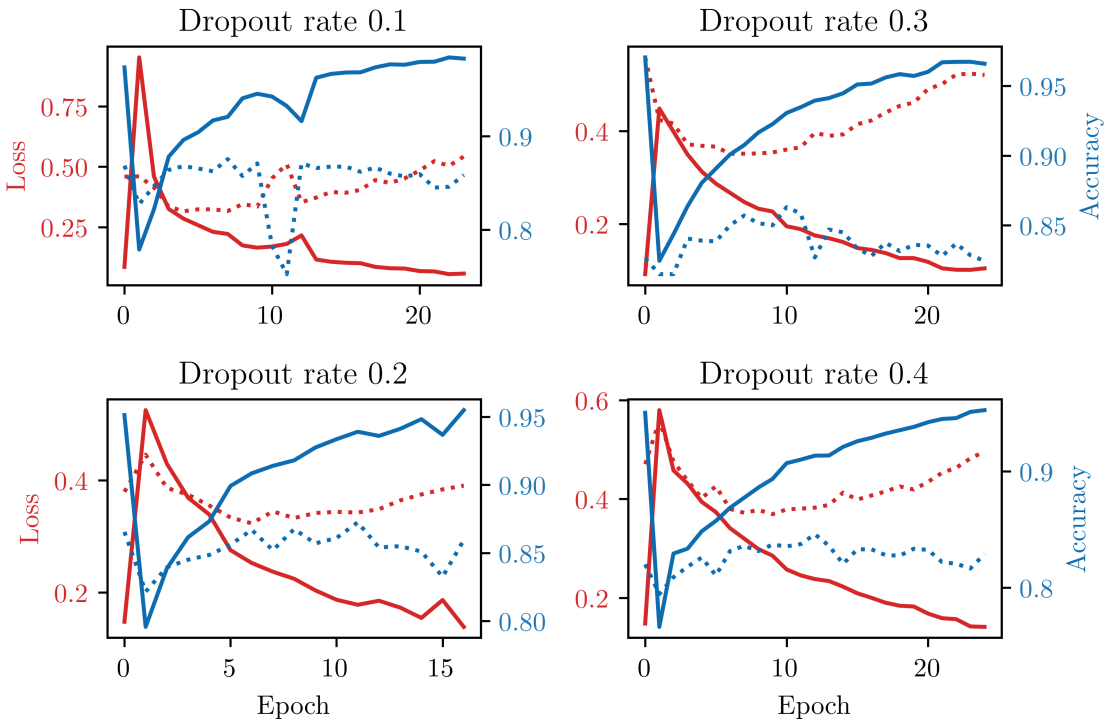

Figure 23: LSTM with varying dropout rate

The CNN network ran with varying learning rate from 10^{-1} to 10^{-5} and produced data consisting of validation loss, validation accuracy, training loss and training accuracy of each epoch of the run. In figure 14, the training data is represented by the continuous line and the vandalization data is represented by the dotted line.

Validation accuracies at minimal validation losses for the models was 83.7% for Dropout rate of 0.4, 85% for Dropout rate of 0.3, 86.7% for Dropout rate of 0.2 and 86.8% for Dropout rate of 0.1. The model with 0 dropout rate has highest performance amongst all model. Dropout rate of 0 was chosen for the LSTM.

The best LSTM model was the model with learning rate of 10^{-4}, four models were trained. The average accuracy of the models overall was 88.73%.

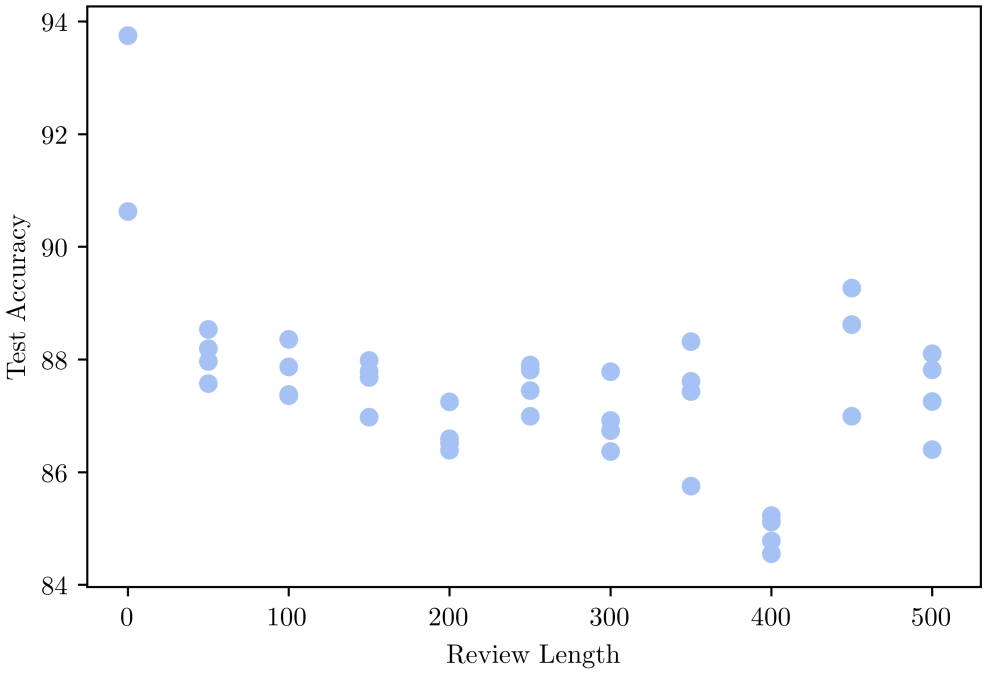

Figure 24: LSTM final accuracy variation with review length

In the above graph, the review length on the x axis is not specifically for that value. For example, for the points on the graph corresponding to 100 Review Length, reviews with ± 25 words are considered. In the figure, the test accuracy decreases sharply for review length of 200 and 400 respectively. Overall,the test accuracy of the data points varies around 88%.

For both the networks, there is dip around 400 review length data point. Overall, the test accuracy of CNN is higher compared to LSTM for all the review length data points.

Section 4.2: Runtime Performance

To get the run time performance of the neural networks, each network was run on 20,000 processed samples on NVIDIA Tesla K80 processors and Intel (R) Xenon (R) CPU @ 2.2GHz using GoogleCollab. Identical commands were used to run and time for execution was measured. It took 5min 52s, 5min 37s, 5min 27sCPU time for the whole test set to run on

the LSTM. It took3.71s ,2.98s, 2.89sCPU time for the whole test set to run on CNN. Average CPU time of LSTM is 5min 39sand 3.18s for CNN respectively. This performance is measured using the “model.predict()” functionality provided by keras, and requires a rectangular array. In practice for each example the running time would vary as the models conceptually can take inputs of different lengths.

Section 5: Evaluation

From the above analysis of investigation it is clear that CNN is better than LSTM in terms of and accuracy of test and runtime taken by CPU.

Section 5.1: Prediction Accuracy

From the graphs of review length vs test accuracy (fig.22 and fig.25), the training **accuracy of CNN is higher for any review length compared to LSTM**. Hence, LSTM’s performance is lower on average than CNN’s.

However, both the networks’ test accuracy drops significantly at review length of 400 words. Moreover, there is a much greater variation in the accuracy of LSTM model compared to the CNN model. In both the graphs, it is evident that **the test accuracy decreases with increase in the review length**. The decline in the test accuracy means that the neural network isn’t able to correctly judge the sentiment of the review as the word count of review increases. The reason is both these architectures have not been optimized to handle this task. For LSTM’s, Bi Directional LSTM’s could be used, for CNN attention-based CNN networks could be used. Further, the embedding layer of the network could have been optimized by having regularization and dropping out words with random probability

Another reason for lower accuracy is that no preprocessing was done on the tokens used in the dataset. This has led to large number of similar tokens. Therefore tokens which were not there in 15000 tokens used, could not contribute to the accuracy.

Another reason for higher performance of the CNN would be that CNN filters consider context before and after whereas LSTM only considers input before the current time step.

Section 5.2: CPU time

CNN	LSTM
3.71 s	5min 52s
2.98 s	5min 37s
2.89 s	5 min 27 s

Table 2 : CPU runtime performance

From Table 2, the mean CPU time taken for evaluation of LSTM was 5.64 min whereas the mean time taken for evaluation of CNN was 3.18s. CNN is approximately 106 times faster than LSTM. However, the CPU time of LSTM depends on the structure which can be modified further to obtain lower CPU time.

Section 5.3: Architecture

CNN architecture is more parallelizable as compared to LSTM as in LSTM at each time step the computation from previous state is required. The computational cost associated with each cell is also much more, requiring sigmoid and tanh nonlinear operations, whereas, for CNN multiplication and addition are the only operations used.

Section 6: Conclusion

Summing up the above analysis and evaluation, the RQ: **"To what extent convoluted neural network (CNN) does better sentiment analysis than long short-term memory (LSTM)?"**, could be answered as CNN seems to be better in sentiment analysis both in terms of run time and accuracy. Usage of CNN results in higher accuracy on the task and it is still accurate for extraction for longer reviews. Moreover, the runtime of CNN is much faster as compared to LSTM by 106 times. So seems to be really helpful in analyzing reviews. The recent growth in popularity of online reviews is due to its application in sentiment analysis. Concrete conclusions are made based on it. Real life application areas ranging from taking opinion on any recent social activity on social media websites to taking reviews on products on company websites, are resulting in useful opinion mining by the use of CNN.

Limitation

No preprocessing resulted in words which were not present in the training data, could not contribute to the analysis. And also only the top 15000 most frequent words were considered. Usage of pre trained word embeddings would remove this limitation.

The aspect of transfer learning[19], in which neural network trained on large dataset used as starting point for training neural network on a similar task is not explored in the essay.

Future Scope

The analysis needs to be further extended to take recent advancements such as Bi-directional LSTM, attention-based CNN along with networks which use the structure of natural language.

19

Bibliography:

Andrew L. Maas, et al. Learning Word Vectors for Sentiment Analysis.

Collobert, Ronan, et al. "Natural Language Processing (almost) from Scratch.", 2011, https://arxiv.org/abs/1103.0398.

Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins. "Learning to Forget: Continual Prediction with LSTM." Neural Computation, vol. 12, no. 10, 2000, pp.2451-2471. MEDLINE, http://www.mitpressjournals.org/doi/abs/10.1162/089976600300015015, doi:10.1162/089976600300015015.

Kim, Yoon. "Convolutional Neural Networks for Sentence Classification.", 2014, https://arxiv.org/abs/1408.5882.

Kingma, Diederik P., and Jimmy Ba. "Adam: A Method for Stochastic Optimization.", 2014, https://arxiv.org/abs/1412.6980.

Moreno-Sánchez, Isabel, Francesc Font-Clos, and Álvaro Corral. "Large-Scale Analysis of Zipf's Law in English Texts." PloS One, vol. 11, no. 1, 2016, pp. e0147073. MEDLINE, https://www.ncbi.nlm.nih.gov/pubmed/26800025, doi:10.1371/journal.pone.0147073.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. "Learning Representations by Back-Propagating Errors." Nature, vol. 323, 1986, pp. 533, https://doi.org/10.1038/323533a0.

Sepp Hochreiter, and Jürgen Schmidhuber. "Long Short-Term Memory." Neural Computation, vol. 9, no. 8, 1997, pp. 1735-1780. MEDLINE, http://www.mitpressjournals.org/doi/abs/10.1162/neco.1997.9.8.1735, doi:10.1162/neco.1997.9.8.1735.

Zoph, Barret, and Quoc V. Le. "Neural Architecture Search with Reinforcement Learning.", 2016, https://arxiv.org/abs/1611.01578.

Livni, Roi, Shai Shalev-Shwartz, and Ohad Shamir. "On the Computational Efficiency of Training Neural Networks.", 2014, https://arxiv.org/abs/1410.1141.

Appendix:

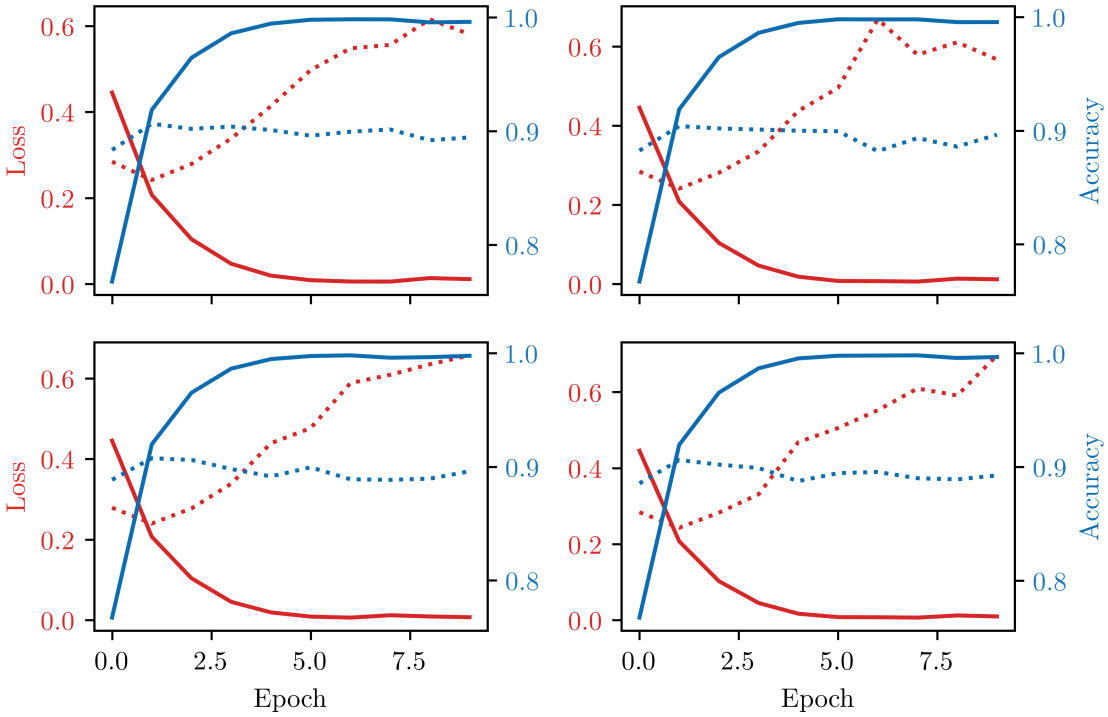

Figure 25: CNN final learning rate curves

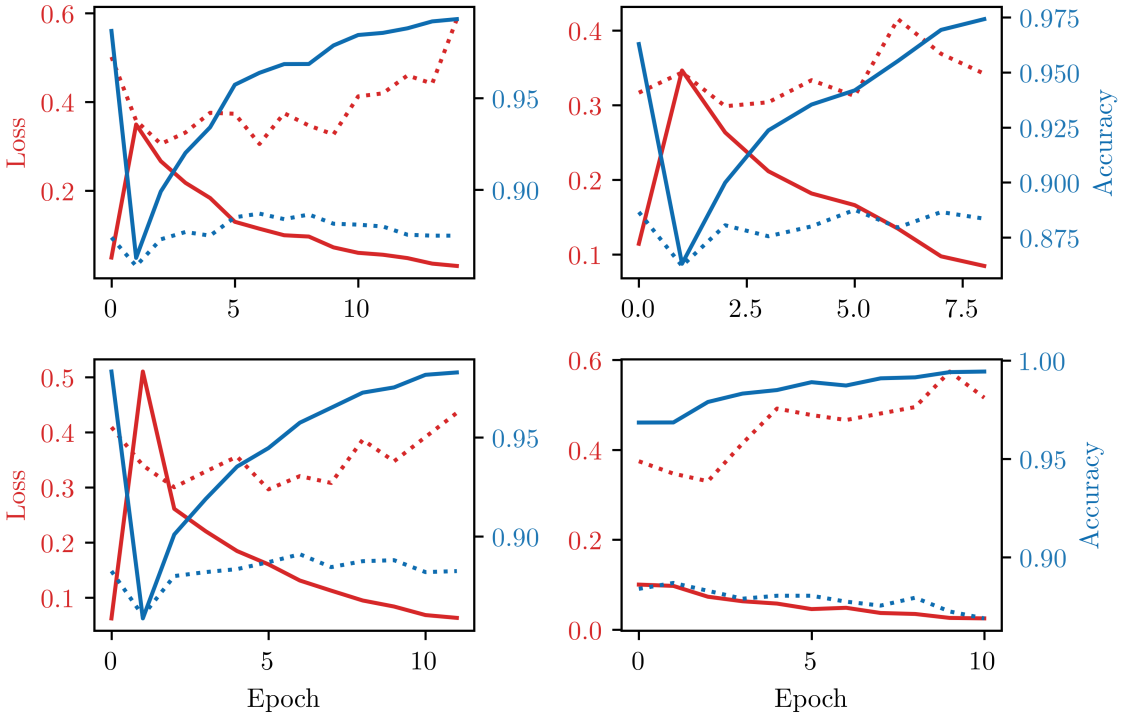

Figure 26: Final LSTM model trained

Appendix

CNN Network

```
def cnn_network():
    # Definng input sentence
    input_sent = Input(shape=(512, 300), name='Input_Sent')
    # Defining convolutions
    c3 = Conv1D(filters=100, kernel_size=3, padding='same', name='C3')(input_sent)
    c4 = Conv1D(filters=100, kernel_size=4, padding='same', name='C4')(input_sent)
    c5 = Conv1D(filters=100, kernel_size=5, padding='same', name='C5')(input_sent)
    c10 = Conv1D(filters=50, kernel_size=10, padding='same', name='C10')(input_sent)
    # Doing 1D max pooling
    m3 = GlobalMaxPool1D()(c3)
    m4 = GlobalMaxPool1D()(c4)
    m5 = GlobalMaxPool1D()(c5)
    m10 = GlobalMaxPool1D()(c10)
    # Doing concatenation
    combined_vec = concatenate([m3, m4, m5, m10], name='Combining_Layers')
    # Dense layer
    fcl = Dense(units=100,activation="relu")(combined_vec)
    fin_input = Dropout(rate=0.2)(fcl)
    final_layer = Dense(units=1, activation="sigmoid")(fin_input)

    model = keras.Model(inputs=input_sent, output=final_layer)
    logging.info('Summary of model: {}'.format(model.summary()))
    return model
```

LSTM Network

```
def lstm_network():
    # Defining input sentence
    input_sent = Input(shape=(512,), name='Words')
    input_embedding = Embedding(input_dim=TOP_WORDS,output_dim=32, input_length=512,
                                name='Embedding_Layer') (input_sent)
    # Defining LSTM
    lstm_layer = LSTM(100,recurrent_dropout=0.3)(input_embedding)
    # Dense layer
    final_layer = Dense(units=1, activation="sigmoid")(lstm_layer)
    model = keras.Model(inputs=input_sent, output=final_layer)
    logging.info('Summary of model: {}'.format(model.summary()))
    return model
```

Neural network run:

```
Train on 21011 samples, validate on 2000 samples
Epoch 1/30
   64/21011 [..............................] - ETA: 18:59 - loss: 0.6935 - acc: 0.4219
```

Output sample file:

Training loss	Training accuracy	Validation accuracy	Validation loss

```
0.6797647575378418,0.5684,0.66,0.6199826307296753
0.5250095624446869,0.75425,0.801,0.44364314222335816
0.372968801856041,0.83755,0.849,0.35587224197387696
0.2852023903846741,0.88335,0.875,0.30660747027397156
0.2254561345100403,0.91435,0.888,0.28296604681015014
0.18174867789149285,0.93465,0.893,0.2675727758407593
0.14749376824498175,0.9503,0.9015,0.2649960176944733
0.11768885007798671,0.96385,0.8975,0.26400902712345126
0.09258156110346318,0.97395,0.897,0.27164827036857603
0.07164117115288973,0.98135,0.899,0.28095517176389695
0.05340203659981489,0.98775,0.8965,0.30027781677246096
0.03846722876802087,0.99205,0.897,0.3164272980690002
0.027187326814234258,0.99565,0.8925,0.33949179977178573
0.018199989038147034,0.9978,0.8905,0.36263448697328565
0.01173608318027109,0.9991,0.8915,0.39039487844705584
0.007437394000496715 5,0.9996,0.89,0.41813845187425613
0.0045796343316324055,0.99985,0.8915,0.44287264919281005
0.0027339592026080937,1.0,0.89,0.46926725339889525
0.001728116137732286,1.0,0.889,0.49530682009458543
0.0011272702184040 1,1.0,0.89,0.5225145847797393
0.0007290382798062638,1.0,0.8915,0.5492316278219223
0.0004836236908158753,1.0,0.89,0.5770943535864353
0.0003224696469318587 6,1.0,0.8915,0.6002771466970444
0.0002148236785127664 8,1.0,0.8905,0.6229168847650289
0.0001455038890155265,1.0,0.888,0.6463992527723312
9.859006350161508e-05,1.0,0.8885,0.6688455163836479
6.708075259957695e-05,1.0,0.887,0.6910213522017002
4.560934121036553e-05,1.0,0.8865,0.7137918017506599
3.138374077152548e-05,1.0,0.886,0.7364815162122249
2.1650469048472586e-05,1.0,0.8875,0.7580001363307237
```