Multi-Class Sentiment Analysis Using CNN

Prashanth Kumar Daram
Student Id: 1116540
Department Of Computer Science
Lakehead University

I. Abstarct

Sentiment Analysis specifies the opinion of the sentence as positive or negative by considering polarity of words. This paper presents implementation of Convolution neural networks for predicting better solution. In this paper, the convolution neural network predicting multi-class sentiment analysis for text-based movie reviews. This model uses some of the deep learning-based techniques for performing multi-class sentiment analysis. We are using Rotten Tomatoes movie review dataset for trainig the model. The Rotten Tomatoes movie review corpus is a huge collection of movie reviews. In this dataset, each sentence is parsed into its tree structure and each node is assigned a fine-grained sentiment label ranging from 0 to 4 where the numbers represent very negative, negative, neutral, positive and very positive respectively. The paper explains about preprocessing techniques to clean the data such as tokenization, punctuations and stop words removal, stemming and lemmatization. The paper also discusses about methods to load dataset, display dataset, train the model and test the model. The goal of this model is to minimize the loss value and generate best accuracy score. With the help of machine learning concepts model is predicting best values.

keywords used- Sentiment Analysis, Multi-class, Deep Learning, Convolution,kernal Size,batch size, epoch value, neural network, Maxpool, Input, Flatten layers

II. Introduction

With rapid growing of technology, a large amount of information in form of text is rapidly available which requires categorization in a systematic and efficient way. Although most of the categorization works focuses on the basis of subject matter, with the readily increase of social media and blogs which exhibits personal positive or negative opinions about a given information required a different kind categorization. This new type of categorization, called Sentiment Analysis, is the main part of this paper.

Now-a-days, People frequently using online services to share their feelings and commenting on other people's posts, online products or current issues. Analysing sentiments of this comments is mostly important for governments, manufacturers, and companies because it helps them to get a certain feedback and general review of users and feelings of the customer on their products and services. Customers also need these kinds of services, while purchasing products and services because, they are influenced by online reviews and recommendations.

Sentiment Analysis: Sentiment Analysis is called as Opinion Mining. It uses techniques of Natural Language processing, Text Analysis, Computational Linguistics for identifying the information systematically and studying effective states of information. Sentiment analysis mostly applied to reviews of customers and survey responses, online and social media posts. Sentiment analysis indicates weather the given text is positive or negative.

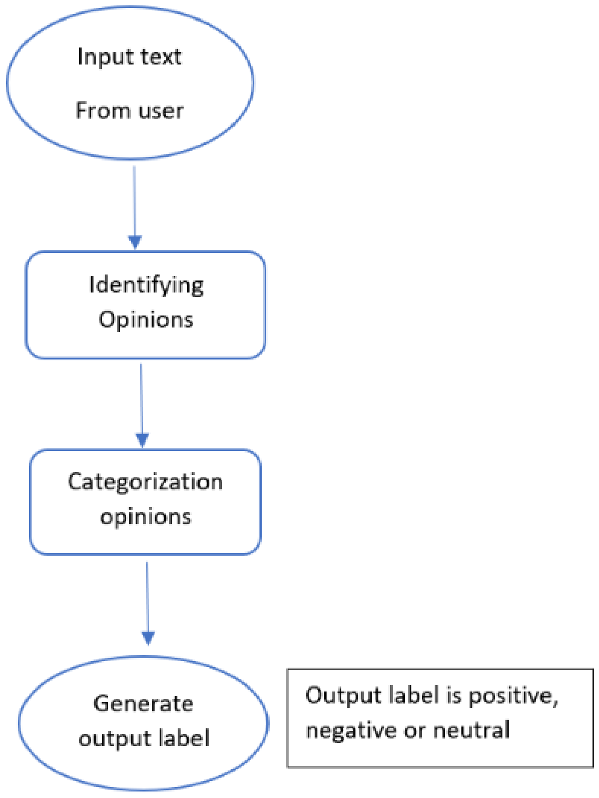

Fig. 1. Sentiment Analysis Flow Chart

Multi-class sentiment analysis: Multi-class sentiment analysis is extension to binary sentiment analysis which classifies only weather the given text is only positive or negative. The Multi-class sentiment Analysis provides scale of values that represents the strength of positivity or negativity of a text as rank. It gives more refined analysis which is most important in real life applications. For example, in comparison of several opinions and for marking ranks to various opinions.

Stemming: Stemming is a process of converting word into its root form by removing suffixes and prefixes such as -ed, -ize, -s,-de, mis. It is a normalization technique.

Lemmatization: Lemmatization is a process of converting words into their root form. It is one of the normalization technique. Lemmatization uses dictionary of words for converting word into its root form.

In this paper, we are applying deep learning-based approach for multi-class sentiment analysis on rotten Tomatoes movie review dataset. we shown different results by applying various deep learning techniques to given dataset.

Deep learning techniques such as convolution neural networks, recurrent neural networks are becoming popular in artificial intelligence research. Example of convolution neural network is one dimensional convolution neural network.

Artificial Neural Networks: We process Data Modeling with the help of artificial neural networks. We use Artificial Neural Networks for performing prediction. The results obtained by this network are very accurate. Representative data is mainly used in artificial neural networks. It is proved that artificial neural networks give greater efficiency and robustness.

Convolution Neural Networks (CNN: Convolution Neural Network is an Artificial Neural Network, it is most popularly used for analyzing images. It also can be used in data analysis and classification problems. CNN has ability to detect pattern and makes sense of patterns, pattern detection making CNN most useful.

What makes CNN different form other neural network models: Multilayer perceptron's or other neural networks are fully connected networks. In this network each neuron in one layer is connected to all other neuron in next layer. Whereas, in CNN the neuron is connected to nearest neighbor neuron. CNN makes complex network to be simple. CNN has input layer, output layer and hidden layers which are called as convolution layers. Convolution layer makes CNN most powerful network. The convolution layers contain some filters which helps them to abstract features or patterns. The main advantage of CNN is its architecture. The architecture of CNN is very simple, it contains input layer, convolution layers, pooling, RELU (rectified linear unit layer) acts as activation function ensuring nonlinearity and fully connected layer. The main function of RELU is conversion of non-linear model into linear model. Training and testing of model are simple with CNN. The CNN is used in many applications such as analysing the images, image classification, image and video recognition and in Natural Language Processing.

In this model I am using One Dimensional CNN (1-D CNN). 1-D CNN helps in providing best solution with help of non-linear regression model. Our model depends on some factors for generating good results. It depends on Batch Size, Learning Rate, Kernel, Number of Layers.

kernel: Kernels are very important in Neural Networks such as convolution neural networks and deep learning. We use kernel to perform statistical analysis. Kernels also called as filters. They perform filters on input. Kernel has two parameters which are height and width. We can customize our kernel size according to our convenience.

III. Literature review

Now-a-days, Social media content is increasing rapidly and the social media content making impact on people's behavior, this made many researchers to do research in studying these online media platforms. The most of their research focused on sentiment analysis and opinion mining. Here, Opinions of the people are automatically identified towards specific topics by analyzing their posts and publications. The task of multi-class sentiment analysis is identification of exact sentiment expressed by user rather than the entire sentiment polarity of user message or post. Multi-class sentiment analysis is different from binary classification. We are using some approach different from conventional method to generate better solution. The author of [2] performed multi-class sentiment analysis on twitter data set. Data set is the huge amount of data collected from twitter. The approach followed in this method is quantification. Through quantification, it identifies all the existing sentiments within an online post instead of attributing a single sentiment label to it. For this model, an approach is proposed that automatically assigns different scores to each sentiment in a tweet and it selects the highest score sentiments which gives sentiment of the text. To execute this model, we used a tool which consists of some deep learn models. We observe the feasibility of quantification by performing this approach on data set taken from twitter data for 11 different sentiment classes. The data set is labelled manually, and the automatic analysis results are checked against the human notation. The feasibility of this task shown through experiments and reaches an accuracy more than 65%.

The conventional Multi-class classification models does report some drawbacks, compared to deep learning models. The multi-class classification through the deep learning models to sentiment analysis provides an exact sentiment rather than traditional approaches such as positive or negative.

All the approaches that exist uses neural networks to solve sentiment analysis problem. In this paper I proposed the model uses convolution neural network (CNN) approach with associated operations like Max pooling and non-linear activation functions such as ReLU, SoftMax and Sigmoid.

IV. Proposed Model

Main Steps Involved in this Model:

- **Pre-Setup (Importing all the required packages)**
- **Dataset Processing**
- **Network Creation**
- **Training the model**
- **Testing the model**

Pytroch: Pytroch is an open source machine learning library developed by Facebook. It supports C++, Python and Cuda.

Keras: It is an open source library and it is related to neural network library. It can be treated as one step higher than tensor flow. It is mainly used for deep neural networks.

Libraries Uesd in this Model

- **Pandas**-Pandas is used for performing analyzing on values and to manipulate data.
- **Sklearn**-Sklearn supports Regression, classification and clustering algorithms. It also supports SVM (Support Vector Machine).
- **Numpy**-It is used for performing mathemathical calculations on multi dimensional arrays.
- **Matplotlib**-Matplotlib is used for ploting mathemathical values in forms of graphs
- **Torch.nn**- We train the neural network using Torch.nn library.
- **Flatten**-Flatten is mainly used to convert a different size grids into a straight one.
- **Maxpoo1d**-It reduces the dimensionality of an input by prividing maximum value in each patch of feature map.
- **Linear layer**-linear layer has a capability to learn correlation between input and output.
- **RELU**-RELU (rectified linear unit layer) acts as activation function makes sure nonlinearity and fully connected layer. The main function of RELU is converting non-linear model into linear model.
- **SoftMax**-SoftMax is an activation function. It maps an output to range from [0,1] and sum of values in output is equal to one. Therefore, output of SoftMax is probability distribution. In logistic regression model, SoftMax is used for Multi-classification.
- **Sigmoid**-Sigmoid is an activation function. It has an output range from 0 to 1 [0,1]. It helps to predict probability as an output. In logistic regression model, Sigmoid is used for binary classification.
- **SGD**-SGD is a sophisticated Gradient Decent. It is optimizer mainly used for optimizing.
- **NLTK**-It is called as Natural Language Tool Kit. NLTK is written in Python language. It is large number of libraries and programs for statistical processing of Natural Language.
- **NLTK**-It is called as Natural Language Tool Kit. NLTK is written in Python language. It is large number of libraries and programs for statistical processing of Natural Language.
- **Random**-It helps to set random seed and to shuffle our data.
- **Stop words**-Stop words are English common words does not add anything to our sentiment frequencies. Stop words library from nltk helps to remove stop words in our document.
- **WordNetLemmatizer**-Lemmatization is a process of converting word into a root form. Wordnet is large and publicly available lexical database for the English. It establishes relationships between words which are semantically structured.
- **Porterstemmer**-Stemming is a process of converting word into root form by removing suffixes and prefixes such as -ed, -ize, -s,-de, mis. It is a normalization technique. Porterstemmer is a nltk library which helps to perform stemming on given text.
- **CountVectorizer**-Sklearn CountVectorizer transforms text present in the corpus to a term of vector or token counts. It also helps to pre-process the text prior to generating vector representation and making it best feature representation module for text.
- **Tfidf Vectorizer**-It performs transformation of text into feature vectors that can be used as input for estimator.

The proposed model is based on implementation of one-dimensional convolution neural network using non linear regression for prediction. Here I am using Google Colab as IDE. Load the Rotten Tomatoes movie review dataset into Google Colab. Once dataset is loaded, we perform text normalization and data preprocessing and then we will create a custom model, from scratch, using 1D convolution layers. Next, we divide the dataset in training and testing. In this model we are dividing dataset into 70% for training and 30% for testing. we will create a custom method to train our new model in batches and let it run for a set number of epochs. At last, we will evaluate our model on the testing dataset to see how it performs.

A. *Pre-Setup*

Here I will explain clearly in steps. Google Colab is free to use, and it have many advantages. Colab runs the program in less time and generates the output. The program is stored in drive and we can access it when ever we need. The first step is open the google colab and connect to GPU runtime. Second step is to import all the required libraries to work with our dataset. We need a pandas library to read our dataset. we print the data using head function. The table 1 displays first 8 rows of the dataset.

```
# Import the pandas library to read our dataset
import pandas as pd

```

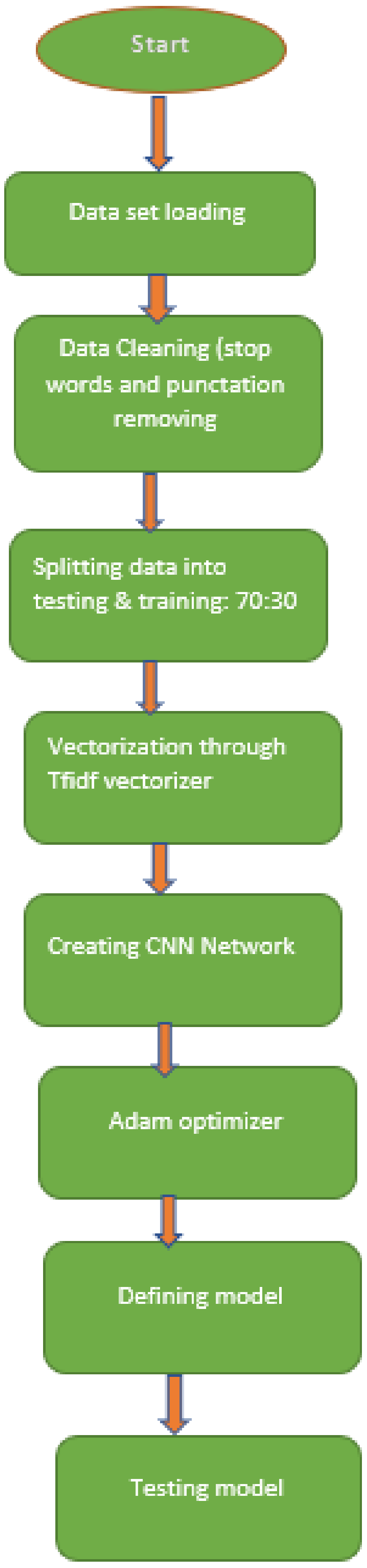
Start
Data set loading
Data Cleaning (stop words and punctation removing
Splitting data into testing & training: 70:30
Vectorization through Tfidf vectorizer
Creating CNN Network
Adam optimizer
Defining model
Testing model

Fig. 2. flow chart diagram of the model

```python
# Get the train/test split package from sklearn for
    preparing our dataset to
# train and test the model with
from sklearn.model_selection import train_test_split

# Import the numpy library to work with and
    manipulate the data
import numpy as np

# import data from github link
data = pd.read_csv('https://raw.githubusercontent.
    com/cacoderquan/Sentiment-Analysis-on-the-Rotten
    -Tomatoes-movie-review-dataset/master/train.tsv'
    , sep='\t')

# Check the head of the dataframe
data.head(8)
```

Listing 1. Dataset loading

TABLE I
FRIST 8 ROWS OF A DATASET

PhraseId	SentenceId	Phrase	Sentiment
1	1	A series of escapades ...	1
2	1	A series of escapades ...	2
3	1	A series	2
4	1	A	2
5	1	series	2
6	1	of escapades demonstrating ...	2
7	1	of	2
8	1	escapades demonstrating ...	2

After loading data set, we import sklearn train and test library from sklearn for training and testing our model. The numpy library is used for manipulating the data. Next, we import nltk library, which consists large number of libraries and programs for statistical processing of Natural Language. We read the dataset using panda's library. We use random library to shuffle our data. Next, we get number of unique sentences in the dataset. Later, we extract the full sentences from the dataset and place them into a data frame. The data frame consists of full sentences with phrase and sentiment.

B. Data Preprocessing

Text Normalization : Frist step is to import all the libraries required for stop words removal, punctuation removal, stemming and lemmatization. We import stop words library from nltk corpus to remove the stop words. Next, we import WordNetLemmatizer, PorterStemmer, LancasterStemmer. Lemmatization and Stemming are the normalization techniques to covert the word into its base/root form. Stemming converts the word into its base form by removing the suffixes and prefixes of the word. Lemmatization uses dictionary of words for converting word into its root form. Second step is setting true or false flags and we will see the impact of this flags on the model performance later. we clean and normalize the review text based on the flags set. Finally, we change the label of review and combine all words into a review sentence.

Spliting Data: we split the dataset into training and testing portions. We are splitting the dataset into 70% for training and 30% for testing. Next, data is placed into data frames to get numpy arrays out and to perform vectorization , We use sklearn's built in vectorizer such as CountVectorizer , TfidfVectorizer to convert our text reviews into vectors which is the size of our entire vocabulary. We can mention stop words removal statement and n-gram usage with in the vectorizer. Firstly, we fit the vectorizer on entire text so that it contains list of words, and then performs transformation of training and testing data, which is to be used for our model later. After that we are converting the datasets to numpy arrays using numpy() function to work with our model.

C. Network Creation

First, let us import all the required libraries to build our network. So, import sequential library from keras model. Next import Conv1d, MaxPool1d, Flatten, Linear, relu, Softmax, Sigmoid, DataLoader, Dense libraries from keras. Create the sequential model and add all other layer to the model.First add the Conv1d layer which has parameters like kernel size, filters and activation function. We are using 1D convolution which still expects a three-dimensional array to process our dataset in 1D fashion. Next add one more convolution layer to the model. Define the max pooling layer with pool size equal to 2. Next, add the Flatten layer to model. Finally add two dense layers which contains ReLu activation function in first layer and second layer has SoftMax activation function. We run the output through the relu function, and it gives values between the range 0 and 1 for our multi-class classifier and loss function. We run the output through the SoftMax layer for multi-class classification to get the integer values between range 0 and 1 for accuracy and other metrics.

```
from keras.models import Sequential
from keras.layers import Dense, Conv1D, Flatten,
    MaxPooling1D

#assigning the batch size
batch_size = 128
#defining the model
model = Sequential()
#adding 1 D convolution to model
model.add(Conv1D(filters = 128, kernel_size=1,
    activation='relu', input_shape=(x_train_np.shape
    [1],1)))
model.add(Conv1D(filters = 128, kernel_size=1,
    activation='relu'))
model.add(Conv1D(filters = 128, kernel_size=1,
    activation='relu'))
#adding maxpooling layer to model
model.add(MaxPooling1D(pool_size =2))
#adding flatten layer
model.add(Flatten())
#adding dense layer to model
model.add(Dense(100, activation='relu'))
model.add(Dense(5, activation='softmax'))
```

Listing 2. Model Creation

D. Training and Testing the model

To train the model. First, we need to import the optimizer. We import the Adam and SGD (Stochastic gradient descent) optimizers for our model. We import the L1Loss and CrossEntropyLoss package for measuring the performance. Next we use Compile function from keras to train our model. The compile function has three arguments. First argument is name of the optimizer, second one is loss and third is metric. In my model, I used Adam as optimizer, loss is categorical crossentropy and metrics are accuracy,recall, precision and f1 score.

```
model.compile(optimizer='adam',loss= '
    categorical_crossentropy', metrics=[metrics.
    categorical_accuracy,f1_fun,pre,re])
```

Listing 3. Model Training

I used model.fit() function to train our model on certain number of epochs. The fit function has five arguments. The arguments are input data, target data, validation data, epochs and batch size. I defined x-train as input data and y-train-np as target data. I initialized batch size equal to 128 and epochs equal to 10. I defined validation data as x test and y test. A history object contains all the records of training loss values and metrics values at successive epochs, as well as validation loss values and validation metrics values. Load the model and finally test the model.

```
history = model.fit(x_train, y_train_np, epochs =
    10, batch_size = 128)
```

Listing 4. Model Training

V. Results and Discussion

Definitions of Precision, recall and f-1 score:

True Positive (TP): True positive samples are really positive samples and classified as positive.

True Negative (TN): True negative samples are really negative samples and classified as negative.

False Positive (FP): False positive samples are negative samples but classified as positive.

False Negative (FN): False negative samples should be positive samples but classified as negative.

Precision

$$Precision = \frac{TP}{TP + FP}$$

Recall

$$Recall = \frac{TP}{TP + FN}$$

F-1 Score

$$F1 - Score = \frac{2 * (Recall * Precision)}{Recall + Precision}$$

In my model, I used sklearn built in metrics to figure out precision, recall and F-1 score and i uesd weighted as average instead of micro and macro to find out results.

```python
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score

# precision tp / (tp + fp)
precision = precision_score(round_label, r_pred,
    average='weighted')
print('Precision: %f' % precision)
# recall: tp / (tp + fn)
recall = recall_score(round_label, r_pred, average='
    weighted')
print('Recall: %f' % recall)
# f1: 2 tp / (2 tp + fp + fn)
f1 = f1_score(round_label, r_pred, average='weighted
    ')
print('F1 score: %f' % f1)
```

Listing 5. code to find out metrics

Model 1
In model 1, I used two layers of 1D convolution with kernel size equal to one. I used relu and softmax as my activation function in convolution layers. I used one max pooling layer with max pool size equal to 2. I used two dense layers, where one of them have relu as activation function and other have softmax as activation function. Here I used Adam optimizer and loss is categorical-crossentropy and metric is accuracy. I initialized filter size to 64. Firstly, I Trained the model with batch size equal to 64 and epochs equal to 10. At Second, I trained the model with batch size equal to 128 and epochs equal to 10. At last, I trained the model with batch size equal to 512 and epochs equal to 10. The results are stated in below table 2.

TABLE II
MODEL 1 RESULTS

Model 1				
Index	Accuracy	Precision	Recall	F1 Score
1	0.638	0.662	0.593	0.625
2	0.628	0.650	0.598	0.623
3	0.606	0.582	0.606	0.580

Model 2
In model 2, I implemented model with three layers of 1 D convolution with relu as activation function. I initialized filter size to 128 and kernel size to 1. Here I used one max pooling layer with pool size equal to 2 and two dense layers. I used softmax activation function in dense layer. I used SGD (Stochastic gradient descent) optimizer to train my model and loss is categorical crossentropy and metric is accuracy. Firstly, I trained the model with epochs equal to 20 and batch size equal to 64. Next I trained the model with batch size equal to 128 and epochs equal to 20. At last, Next I trained the model with batch size equal to 512 and epochs equal to 20. The results are stated in above table 3.

Model 3
In model 3, I implemented model with four layers of 1 D convolution with relu as activation function. I initialized filter size to 128 and kernel size to 1. Here I used one max pooling layer with pool size equal to 2 and two dense layers. I used softmax activation function in dense layer. I used adam optimizer to train my model and loss is categorical crossentropy and metric is accuracy. Firstly, I trained the model with epochs equal to 20 and batch size equal to 128. Next I trained the model with batch size equal to 512 and epochs equal to 20. At last, Next I trained the model with batch size equal to 1024 and epochs equal to 20.The results are stated in above table 4.

TABLE III
MODEL 2 RESULTS

Model 2				
Index	Accuracy	Precision	Recall	F1 Score
1	0.604	0.581	0.604	0.580
2	0.603	0.580	0.603	0.577
3	0.602	0.579	0.602	0.578

TABLE IV
MODEL 3 RESULTS

Model 3				
Index	Accuracy	Precision	Recall	F1 Score
1	0.610	0.588	0.620	0.578
2	0.601	0.575	0.600	0.571
3	0.615	0.592	0.608	0.586

Comparison of All 3 Models: The table 5 shows all the three models scores. In all three models, model 1 is better than other two models. It have accuracy and other metrics more than other two models. So,I selected model 1 as my final model.

TABLE V
COMPARISON OF MODELS

Final Results in percentages				
Models	Accuracy	Precision	Recall	F1 Score
Model 1	0.638	0.662	0.593	0.620
Model 2	60.41	58.10	60.41	58.20
Model 3	0.615	0.592	0.608	0.586

Over /under fitting issue:
In Over fitting, model performs well in train time but not performs good at testing time.To avoid this situation,we reduce number of Epchos. In Under fitting, model not performs well both in testing and training set. To avoid this issue, we need to change the optimizer.

The accuracy which is good in training time reduces during testing time. While I was using SGD optimizer I got a accuracy value of 0.50 which I didn't find accurate.Under fitting is the concept that occurs when algorithm shows low variance and high bias. In order to solve this problem, I used a new optimizer called Adam where I got an accuracy value of 0.64.

Graphs: The graphs of model 1 accuracy and loss are plotted below

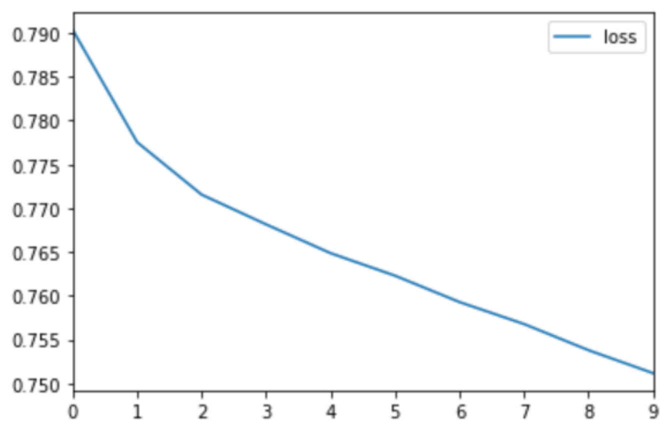

Fig. 3. Model 1 Loss v/s Epoch graph

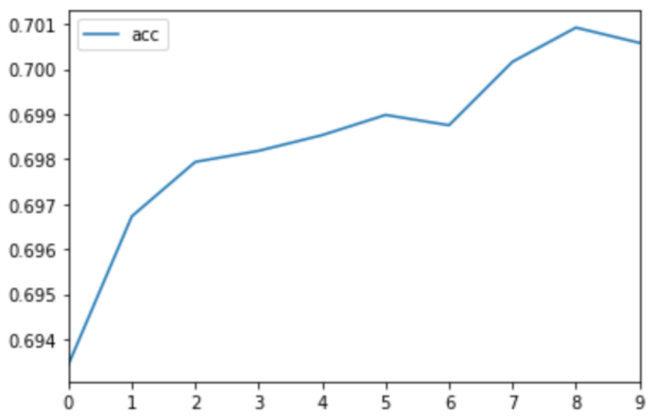

Fig. 4. Model 1 Accuracy v/s Epoch graph

Trainable parameters
The following are the training parameters that are used in CNN Model:
K = Kernel size used in CNN.
N = Number Of Keras.
C = Number of channels of the input image.

Kernal size= 1
Batch size= 512
Epoch size= 10

VI. Dataset used

The data set is extracted from github and below is the following link where u can download it
https://raw.githubusercontent.com/cacoderquan/Sentiment-Analysis-on-the-Rotten-Tomatoes-movie-review-dataset/master/train.tsv

VII. Appendix

We defining the number of convolution layers while training the model. First input is went through convolution layer.Next, the output of CNN Layer acts as input to max pooling layer.Next output of max pooling layer passes as input to dense layer. Finally, we get output from output layer.

VIII. Conclusion

My model generated best solution using convolution neural networks for given problem. My model minimized loss and generated best accuracy,precision,recall and f-1 score.

TABLE VI
Final results in percentages

Accuracy	Precision	Recall	F1 Score
63.80	66.25	59.83	62.57

Github Link:

References

[1] B. Pang, L. Lee and S. Vaithyanathan, “Sentiment Classification using Machine Learning Techniques,” in Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 79-86, 2002.

[2] k.Mondher Bouazizi, M. Tomoaki Ohtsuki,”Multi-Class Sentiment Analysis in Twitter: What if Classification is Not the Answer ”,IEEE,Vol. 6, 2018.

[3] A. Celikyilmaz, D. Hakkani-Tur, and J. Feng, ”Probabilistic model-based sentiment analysis of twitter messages,” in Spoken Language Technology Workshop (SLT), 2010 IEEE, pp. 79-84, IEEE, 2010.

[4] Y. Wu and F. Ren, ”Learning sentimental influence in twitter,” in Future Computer Sciences and Application (ICFCSA), 2011 International Conference on, pp. 119-122, IEEE, 2011.

[5] G. Vinodhini and R. M. Chandrasekaran, “Sentiment Analysis and Opinion Mining: A Survey,” International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 2, No. 6, 2012.

[6] Z. Niu, Z. Yin, and X. Kong, ”Sentiment classification for microblog by machine learning,” in Computational and Information Sciences (ICCIS), 2012 Fourth International Conference on, pp. 286-289, IEEE, 2012.

[7] M. Abdul-Mageed, M. Diabc, and S. Kübler, “SAMAR: Subjectivity and Sentiment Analysis for Arabic Social Media,” Computer Speech and Language, Vol. 28, No. 1, 2013.

[8] E. Cambria, B. Schuller, and Y. Xia, “New Avenues in Opinion Mining and Sentiment,” Intelligent Systems, IEEE, Vol. 28, Issue 2, pp. 15-21, 2013.