Multi-class Sentiment Analysis using Deep Learning

Manva Trivedi
Department of Computer Science
Lakehead University
Student ID: 1095816
trivedim@lakeheadu.ca

***Abstract*—This paper aims to find the most effective Convolutional Neural Network solution for text-based movie review multi-class sentiment analysis. For classification, I have used raw train data of Rotten Tomatoes movie reviews available on GitHub. I have build a multi-class sentiment analysis model for classification. This model consists of three convolutional layers, two dense layers and uses softmax and RELU as activation functions after applying various operations such as max pooling and flattening. I have also performed hyper-parameter tuning and assigned a batch-size of 128, number of epochs as 10 with 5 classes. I also tried to use different optimizers out of which Nadam fits best to the model. For implementation I have used google colab in which we connected to GPU runtime, created a custom model using one dimensional convolution layers and train the model to see how well it performs by calculating R3 Score and accuracy. The model built for this assignment achieves an accuracy of 73.3 % for training and 61.86 % for testing.**

I. Introduction

Convolutional Neural Networks has delivered amazing results in recent years in the field of Speech Recognition and Natural Language processing(NLP) as it has been studied broadly. Using the rapid growth in data volume and the major developments in graphics processor unit capacities, research on Convolutional Neural Networks has appeared quickly and has achieved state-of-the art results on various tasks. Basically a Convolutional Neural Network(CNN) takes input, assigns importance to it and performs various operations such as convolution, max pooling, padding, batch normalization, flattening and the flattened output is feed forwarded to the network to update weights and biases via back propagation to train the model. The trained model will be able to differentiate between low-level features aver a series of epochs and later classify them using various classification techniques.

A CNN consists of convolution layer/s and activation function/s which includes feature learning and classification. Convolution layer extracts the features and performs pooling operations for dimensionality reduction. The output is then fed to a fully connected layer after flattening and further uses activation function for classification. In this assignment I have implemented a convolutional neural network for multi-class sentiment analysis, trained the model and tried to improve accuracy of the model.

II. Background/Literature Review

CNNs are a category of Neural Networks that have shown amazing results in various fields such as image classification and image processing. CNNs have been successfully identifying different objects such as traffic signs apart from decoding facial recognition, understanding climate, and analysing documents. There are four main operations performed in the CNN as listed below: [1]

- Convolution
- Non Linearity (ReLU)
- Pooling or Sub Sampling
- Classification (Fully Connected Layer)

Convolution is used to extract features from the input and it also preserves the relationship for example between the pixels if we consider image as an input. RELU is an activation function which replaces all the negative values in the feature map by zero and is found to perform better than other activation functions such as softmax, sigmoid, tanh.

The output of RELU is fed as input to the pooling layer. There are three types of pooling:

- Sum Pooling
- Avg Pooling
- Max Pooling

Spatial pooling which can also be called down-sampling reduces the dimensionality of feature map by retrieving useful information. I have used max pooling in the assignment which takes the largest value from the feature map in that window. We can also use average pooling which take average of the values from the feature map within the window. I have used max pooling as it performs much better than average pooling.

Lastly comes the fully connected layer which after performing flattening, uses activation function in the output layer for classification.

III. Experimental Analysis

This experiment includes various steps such as preprocessing, creating a one dimensional Convolution

Neural Network for classification, training the model, testing the model and making necessary changes to improve accuracy of the model. To get started with the implementation I imported python libraries such as pandas, sklearn, numpy, seaborn and tensorflow. After importing data from a csv file of the Rotten Tomatoes movie reviews dataset in a data frame, for preprocessing we split the data in a ratio of 70:30 for training and testing respectively.

A. Dataset

I have used raw train data of Rotten Tomatoes movie reviews available on GitHub and then used 70 percent of it for training and 30 percent for testing. Each row is assigned a sentiment label ranging from 1 to 5 where the numbers represent very negative, negative, neutral, positive and very positive respectively. The dataset is a collection of movie reviews and contains four attributes namely PhraseId, SentenceId, Phrase and Sentiment which are explained in Table 1:

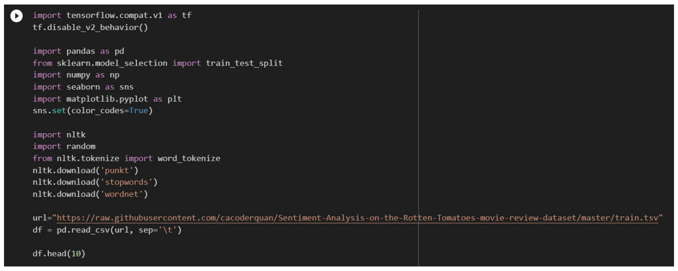
TABLE I
DESCRIPTION OF ATTRIBUTES

Attribute Name	Data type	Description
PhraseId	Integer	ID of a phrase of the review
SentenceId	Integer	ID of a sentence from a phase of the review
Phrase	String	A group of words standing together as a block, typically forming a component of a review
Sentiment	Integer	Each row is assigned a sentiment label ranging from 1 to 5 where the numbers represent very negative, negative, neutral, positive and very positive respectively

B. Libraries

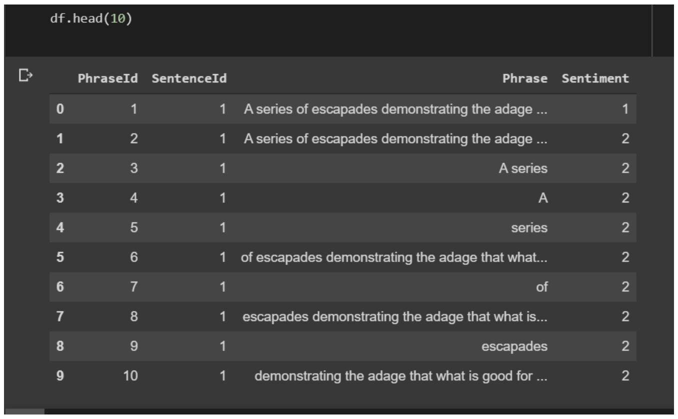
In order to do the classification using CNN, I have used the pandas library to read the dataset, train/test split package from sklearn for preparing the dataset to train and test the model with, numpy library to work with and manipulate the data, nltk to download packages (punkt,stopwords,wordnet) and keras library to build the network. Hereby is a list of libraries used:

- pandas
- sklearn
- numpy
- seaborn
- nltk
- keras



```
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()

import pandas as pd
from sklearn.model_selection import train_test_split
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(color_codes=True)

import nltk
import random
from nltk.tokenize import word_tokenize
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')

url="https://raw.githubusercontent.com/cacoderquan/Sentiment-Analysis-on-the-Rotten-Tomatoes-movie-review-dataset/master/train.tsv"
df = pd.read_csv(url, sep='\t')

df.head(10)
```

Fig. 1. Load dataset using URL



```
df.head(10)
```

```
   PhraseId  SentenceId                                              Phrase  Sentiment
0         1           1   A series of escapades demonstrating the adage ...          1
1         2           1   A series of escapades demonstrating the adage ...          2
2         3           1                                            A series          2
3         4           1                                                   A          2
4         5           1                                              series          2
5         6           1  of escapades demonstrating the adage that what...          2
6         7           1                                                  of          2
7         8           1   escapades demonstrating the adage that what is...          2
8         9           1                                           escapades          2
9        10           1     demonstrating the adage that what is good for ...          2
```

Fig. 2. First 10 instances of the dataset

C. Coding

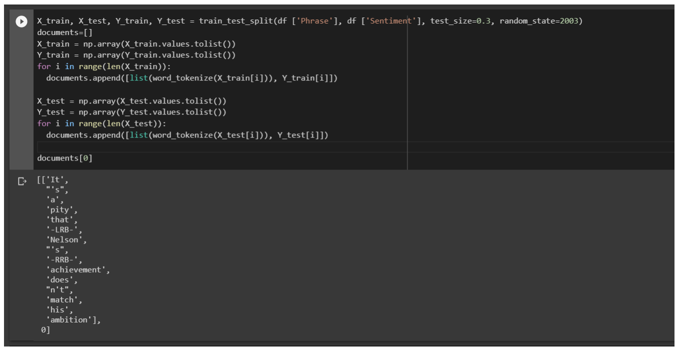
I have used google colab to for implementation as it provides free GPU support, all the major python libraries such as TensorFlow, Scikit-learn, Matplotlib,etc. are already installed, supports bash commands, etc. To load the data into pandas data frame I have used URL method. I have used this URL method because by using this I don't have to upload the data set every time to the Colab directory. The snippet of the code is show in figure 1.

As a part of visualization of the dataset we need to observe our data by studying values of the features for which I have printed top 10 instances of the dataset which can be depicted from by the figure 2. To split the dataset I have used train test split model of sklearn which splits the dataset for training and testing along with value of random state to be 2003.

D. Hyper Parameter Tuning

Hyper-parameters are the parameters whose value can be used to contol the learning process. Hyper-parameters are are the variables which our algorithm uses to adjust to the data. In my model following are the hyper-parameters:

- Learning Rate- 0.002
- beta1 = 0.9
- beta2 = 0.999



```
X_train, X_test, Y_train, Y_test = train_test_split(df ['Phrase'], df ['Sentiment'], test_size=0.3, random_state=2003)
documents=[]
X_train = np.array(X_train.values.tolist())
Y_train = np.array(Y_train.values.tolist())
for i in range(len(X_train)):
  documents.append([list(word_tokenize(X_train[i])), Y_train[i]])

X_test = np.array(X_test.values.tolist())
Y_test = np.array(Y_test.values.tolist())
for i in range(len(X_test)):
  documents.append([list(word_tokenize(X_test[i])), Y_test[i]])

documents[0]
```

```
[['It',
  "'s",
  'a',
  'pity',
  'that',
  '-LRB-',
  'Nelson',
  "'s",
  '-RRB-',
  'achievement',
  'does',
  "n't",
  'match',
  'his',
  'ambition'],
 0]
```

Fig. 3. Splitting the dataset in ratio 70:30

- Epochs- 10
- Batch Size- 64
- Optimizer- Nadam
- Kernel Size- 5

To achieve highest accuracy, I tried different optimizers such as Adadelta, Adam, Adamax and Nadam among which, Nadam gave the best results when epochs were 10 and batch size was 64. The model built for this assignment achieves an accuracy of 73.3 % for training and 61.86 % for testing.

E. Data Pre-Processing

I have used train data of Rotten Tomatoes movie reviews dataset and loaded into pandas dataframe. First I checked class imbalance in tokenized sentences and full sentences. Then I adjusted the parameters to see the impact on outcome. After that to split the dataset s, I used sklearn and used 70% of the data for training ans 30% for testing as shown in the figure 3.

For each review document I have saved review label, stored place holder list for new review, and for each word in the review if the word is a stopword and I want to remove stopwords skip the word and doń́t add it to the normalized review but if the word is a punctuation and I want to remove punctuations skip the word and doń́t add it to the normalized review else add normalized word to the tmp review and update the reviews list with clean review.

F. Proposed Model

The proposed CNN model takes an array as input. So, in order to convert the dataframe into pandas I have used the sklearn library which helps to convert the dataframe into numpy array. The numpy array is still not normalized. As normalization is important as it helps in faster convergence on learning rate and also more uniform influence for all weights. So, in order to perform the normalization, I have used the same sklearn library which helps me to perform the normalization.

I have performed various operations while creating the network such as adding different layers to perform specific tasks such as Convolution, max pooling, flattening and applying activation function. I have used different activation functions such as sigmoid, softmax, relu etc. but RELU activation function and softmax are applied to the output of the proposed CNN to achieve highest accuracy, the output of which is then passed as the input to the next layer to provide non-linearity.

To begin with, we declare a class which defines all the layers including an input layer, an output layer, max pooling layer, three convolution layers, flatten layer, two dense layers, linear layer and initialise and store the parameters. After that I have defined a method which inputs data to the input layer in defined batch sizes. The output of this layer is run through RELU activation function which passes to the max pooling layer. The output of this convolution layer goes through two more phases of max pooling and convolution and the final output is fed to flatten layer after running it through RELU activation function. Finally, we the output layer after running output of flatten layer through activation functions such as softmax and RELU in dense layers.

Next step after creating a network is training the model. Here I have used Nadam as an optimizer and created a method to calculate the accuracy which measures performance of the model. To train the model, we need to define epochs and the optimizer. Here, I have taken 10 epochs and Nadam as an optimizer to get the best possible accuracy according to the model. After performing Hyper-Parameter tuning, I have came up with the optimal hyper-parameters which are shown in Table 2.

TABLE II
OPTIMAL HYPER-PARAMETERS FOR THE PROPOSED MODEL

Hyper-Parameter	Value
Learning Rate	0.002
Optimizer	Nadam
Epoch	10
Batch Size	64
Convolution Layers	3
Dense Layers	2
Kernel Size	5
beta1	0.9
beta2	0.999

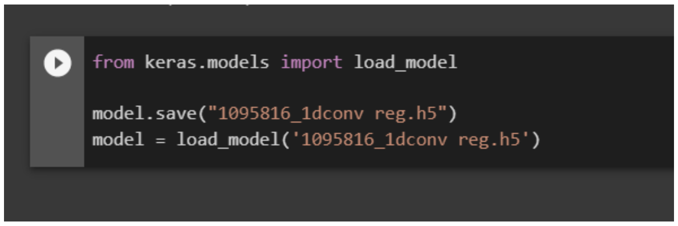
From the Table 3 it can observed that the Nadam optimizer works well and give the highest accuracy for proposed model.

G. Model Storing

After creating the CNN model, in order to save the model I have first connected my drive with the colab using the code displayed in the figure 4. Stored the model with .h5(header file) extension.

TABLE III
COMPARISON OF OPTIMIZER AND LEARNING RATE

Learning Rate	Optimizer	Acc(Train)	Acc(Test)
0.00001	Adadelta	61.45	51.67
0.001	Adam	57	54.18
0.00001	Adamax	64.08	51.9
0.002	Nadam	73.3	61.86



```
from keras.models import load_model

model.save("1095816_1dconv reg.h5")
model = load_model('1095816_1dconv reg.h5')
```

Fig. 4. Snippet for storing the code

ACKNOWLEDGMENT

I want to thank my professor Dr. Thangarajah Akilan who helped me understand the concept of CNN and assist me in my class labs. I also want to thank the our teaching assistants Andrew and Punardeep who took keen interest and solved my issues and concerns about the assignments on time.

IV. CONCLUSION

With one dimensional Convolutional Neural Network which I created and trained, I achieved an accuracy of 73.3 % for training and 61.86 % for testing. To improve the accuracy of the model I performed hyper-parameter tuning by changing number of epochs, batch size and used different optimizers. I have used Nadam as an optimizer which performs well for the model from all the other optimizers such as, Adadelta, Adam, Adamax and Nadam used before choosing the best one.

REFERENCES

[1] An Intuitive Explanation of Convolutional Neural Networks https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/