# Project Report

**for**

# Advanced Data Mining Project 2

## Very Deep Convolutional Networks for Text Classification

**Prepared by-**

**Rishabh Sharma [2016H112154P]**

**Jaishree Janu [2016H1121156P]**

**BITS, Pilani**

**28-April-2017**

## Table of Contents

## Revision History

| Name | Date | Reason For Changes | Version |
|---|---|---|---|
| Rishabh Sharma<br><br>Jaishree Janu | 28-April-2017 | Document creation | 1.0 |

## 1. Introduction

Classification is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs to, on the basis of a training set of data containing observations (or instances) whose category membership is known. An example would be assigning a given email into "spam" or "non-spam" classes or assigning a diagnosis to a given patient as described by observed characteristics of the patient (gender, blood pressure, presence or absence of certain symptoms, etc.).

Text Classification assigns one or more classes to a document or sentence according to their content. Classes are selected from a previously established taxonomy (a hierarchy of categories or classes).

**Problem Statement:**

As a part of this project we have chosen the problem of text classification using very deep convolutional neural network and the motivation of choosing this problem is derived from the paper presented by Alexis Conneau et al in [2]. The main idea is to make the convolutional network deep with multiple hidden unit layers and test with various different hyper-parameters to solve for multi-class text classification problem.

## 2. Convolutional Neural Network

In neural networks, neurons are learning units. Neurons learn how to convert input like text or images into output that is the class label. Convolutional Neural Network also known as ConvNet and CNN make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture. These then make the forward function more efficient to implement and vastly reduce the amount of parameters in the network

CNNs have several layers of convolutions and subsampling(pooling) with activation functions like RELU and TANH applied to the results of convolutions.

In CNN, each region is connected to a neuron in the output. During training, CNN learns the values of its filters. For text classification, features are from character-level to sentence-level namely n-grams, words, phrases and sentences. Complex features are learnt in the later layers of the CNN.

A sample CNN network looks like:-

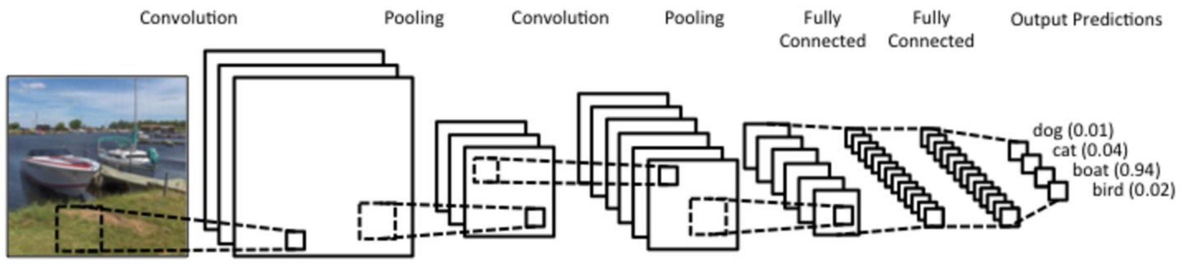


FIG. 1 Convolutional Neural Network

**Convolution:** Convolution is a Sliding window function applied to a matrix. Sliding window is called kernel or filter or feature detector. Filters of different sizes are used to identify different patterns and then these filters are used to multiply its values element-wise with the original matrix (original matrix is the input representing text), then sum them up.

**Sub-sampling(Pooling):** Pooling layer subsample the input. We can apply either max-pooling or average-pooling. We have applied max-pooling to the result of each filter. We don't need to do pooling over the whole matrix. We can pool over a window but for text, we have to apply pool layer over whole complete output yielding just single number for a filter.
Reasons:

1. Regardless of the size of input or size of filters or number of filters, we will get same size of output that is fixed size of output matrix. So, we can use variable sized filters and sentences.
2. Pooling keeps the most salient information. Each filter is for detecting a feature. If that feature appears somewhere in the sentence then value of feature map obtained is large. By applying the max pooling, we are keeping the information whether that feature appeared or not.

**Activation function:** It defines the output of the node based on the set of inputs. To learn more complex functions, we add non-linear activation functions like tanh and relu. If there are no activation functions, CNN would just be same as linear regression. If neurons have seen similar patterns of words or sentences before, then corresponding label gets activated.
When choosing activation function, three points are important:

- Continuity of the function
- Computational power to process large number of neurons in CNN
- Type of desired output categorical or continuous

There are types of activation functions namely RELU, TANH, Sigmoid function.

**RELU (REctifier Linear Unit):**

It is defined as f(x)=max(0,a)where a = Wx+b
Benefits of Relu are:

1. Reduced likelihood of vanishing gradient. When a>0, relu has a constant gradient. In sigmoid function and **tanh**, gradient becomes small when absolute value of x increases. Constant value of gradient results in fast learning.
2. If **RELU** is used, deep network can be trained without any pre-training.
3. It is faster because it do not require any normalization and exponential computation unlike **tanh** and sigmoid functions.

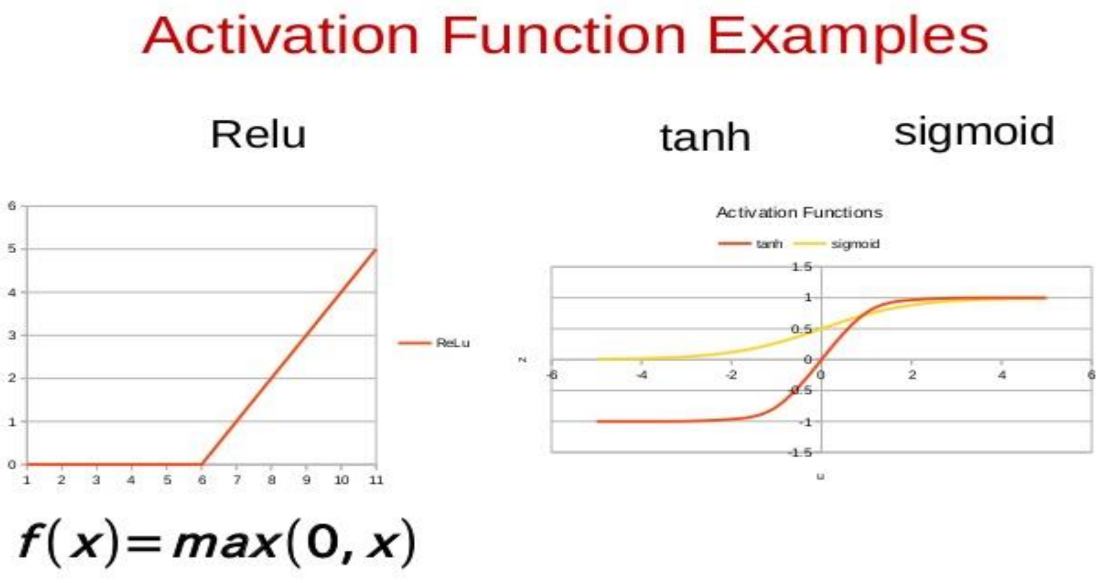


FIG. 1 Activation functions

**Sigmoid function**: It has range of [0,1]. Since its output is between 0 and 1, it can be interpreted as probability. It has nice derivatives which makes learning of weights of neural network easier.

$$S(t) = \frac{1}{1+e^{-t}}.$$

TANH function: It's range is [-1,1] and has asymptotic symmetry. It makes training easier as it saturates in the later layers of the network. It creates output values close to 0 on an average which is beneficial for forward propagation in subsequent layers.

**Fully-connected layer:** Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset.

## Representation of the Text (input to the network)

Sentences or documents are represented as a matrix. Each row of the matrix corresponds to a token or a word.

Each row is a vector representing a word. These vectors are word embedding or low-dimensional representations like **one-hot vectors(that index the word into the vocabulary).** For a 10 word sentence using 100 dimensional embedding, we will have a matrix of size 10*100 as input.
For text classification in neural networks, width of the filters is same as the width of the input matrix. Height of the filter may vary. Filters of words from 2-5 are normally used.

There is no need to represent whole vocabulary, convolutional filters learn good representations automatically.
Many of the features learned in the first layer are similar to n-grams as specified in [3].

## Hyperparameters

For Convolutional Layer, we have to specified 4 hyper parameters.
1. Number of filters
2. Spatial extent
3. The Stride
4. The amount of zero padding

For Pooling Layer, we have to specify 2 hyper parameters.
1. Spatial extent
2. The stride

# 3. Text Classification using Convolutional Neural Network

**Dataset**

AG is a collection of more than 1 million news articles. News articles have been gathered from more than 2000 news sources by ComeToMyHead in more than 1 year of activity. ComeToMyHead is an academic news search engine which has been running since July, 2004. The dataset is provided by the academic comunity for research purposes in data mining (clustering, classification, etc), information retrieval (ranking, search, etc), xml, data compression, data streaming, and any other non-commercial activity.

**Dataset Description**

The AG's news topic classification dataset is constructed by choosing 4 largest classes from the original corpus. Each class contains 30,000 training samples and 1,900 testing samples. The total number of training samples is 120,000 and testing 7,600.

The file classes.txt contains a list of classes corresponding to each label.

The classes are-

World → 1
Sports → 2
Business → 3
Sci/Tech → 4

The files train.csv and test.csv contain all the training samples as comma-separated values. There are 3 columns in them, corresponding to class index (1 to 4), title and description. The title and description are escaped using double quotes ("), and any internal double quote is escaped by 2 double quotes (""). New lines are escaped by a backslash followed with an "n" character, that is "\n".

**Preprocessing**

The data is in .csv file and so for preprocessing the data is fetched using csv package from python and the 1st and 3rd column of each record is picked from the .csv file and is appended into a numpy array.

Once the data is collected into numpy array then the data is collected in batches of 128 records and is passed on to the neural network for processing.

So a row of data which was initially-

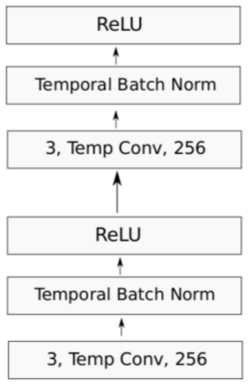
"3","Sorting Out The Winners if US Airways Doesn't Survive", "US Airways' second trip to bankruptcy court holds both promise and peril for the nation's other major airlines."

Has been converted to-

| Class | Text Statement |
|---|---|
| 3 | US Airways' second trip to bankruptcy court holds both promise and peril for the nation's other major airlines. |

**Structure of the Convolutional Neural Networks**

The neural network architecture described by Alexis Conneau et al in [2] is called as VDCNN (Very Deep Convolutional Neural Network) and this architecture uses multiple convolutional blocks stacked on top of each other and max pooling is done after 2 such blocks to reduce the size of the vectors. Each convolutional block comprises of-



Batch Normalization allows us to use much higher learning rates and be less careful about initialization. It also does regularization, in some cases eliminating the need for Dropout

**1. Input Layer-**

This layer takes in the attributes of each row and its corresponding class as input from the preprocessing layer and passes on to the embedding layer. The corresponding class of each record is also passed on to the output layer for comparing to the predicted output to the actual output.

**2. Embedding Layers-**

Each input row is represented as a vectors for words. We have used one-hot-vector representation for representing the input. A one-hot vector is a 1xN matrix (vector) used to distinguish each word in a vocabulary from every other word in the vocabulary. The vector consists of 0s in all cells with the exception of a single 1 in a cell used uniquely to identify the word.

**3. Convolution Layers-**

The convolution layers used in the base paper are modified to the following structure-

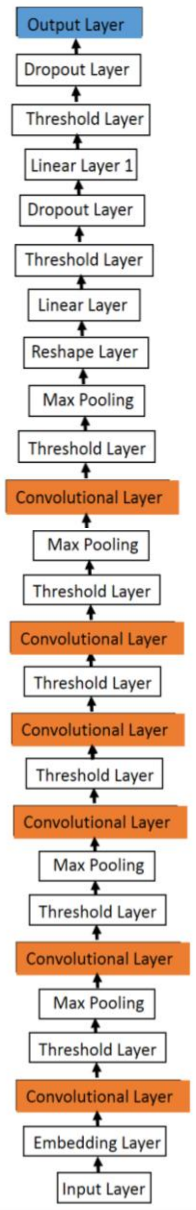
1st Convolution layer is a 3d layer and it has the structure 256 * 7 * 3
2nd Convolution layer is a 3d layer and it has the structure 256 * 7 * 3
3rd Convolution layer is a 2d layer and it has the structure 256 * 3
4th Convolution layer is a 2d layer and it has the structure 256 * 3
5th Convolution layer is a 2d layer and it has the structure 256 * 3
6th Convolution layer is a 3d layer and it has the structure 256 * 3 * 3

The formula used to derive the filter shape for various filters is:

Filter shape = [Conv_layer[1] , Max_Size_Of_Input , 1 , Conv_Layer[2]]

## 4. Modification to VDCNN Architecture-

Following is the new architecture of the system post modifications to the existing VDCNN architecture:



Various parameters specified are as follows-

Convolution Layers = 6
　　No of Neurons = 256
　　Size of Feature = 69
Max-Pooling Layer = 4
Threshold Layer = 8
Dropout Layer = 2
Linear Layer = 2
Fully Connected Layer = 2
　　No of Neurons = 1024
Activation Function = TANH
Pooling Strategy = MAX-POOLING
Input Embedding = One Hot Vector

Dropout keep probability = 0.6
Base rate = 1e-2
Momentum = 0.9
Decay steps = 15000
Decay rate = 0.95
Epochs = 15

## 4. Observed Results

The following are the list of results along with the parameters used with the neural network–

Accuracy = (Σ True Positive + Σ True Negative) / Σ (total population)

Precision = Σ True Positive / Σ Prediction Positive

Recall = Σ True Positive / Σ Condition Positive

F1- Score = 2 * Precision * Recall / (Precision + Recall)

1. Accuracy and Loss observed from the experiments conducted on the configurations mentioned above.

Confusion Matrix:

| Target | Selected | | | | |
|---|---|---|---|---|---|
| Class | 1 | 2 | 3 | 4 | Total |
| 1 | **1483** | 138 | 133 | 146 | 1900 |
| 2 | 138 | **1491** | 137 | 134 | 1900 |
| 3 | 147 | 143 | **1464** | 146 | 1900 |
| 4 | 139 | 144 | 155 | **1462** | 1900 |
| Total | 1907 | 1916 | 1889 | 1888 | 7600 |

| Class A | Selected A | Not Selected A | Total |
|---|---|---|---|
| Actual A | **1483** | 417 | 1900 |
| Not A | 424 | **5276** | 5700 |
| Total | 1907 | 5693 | 7600 |

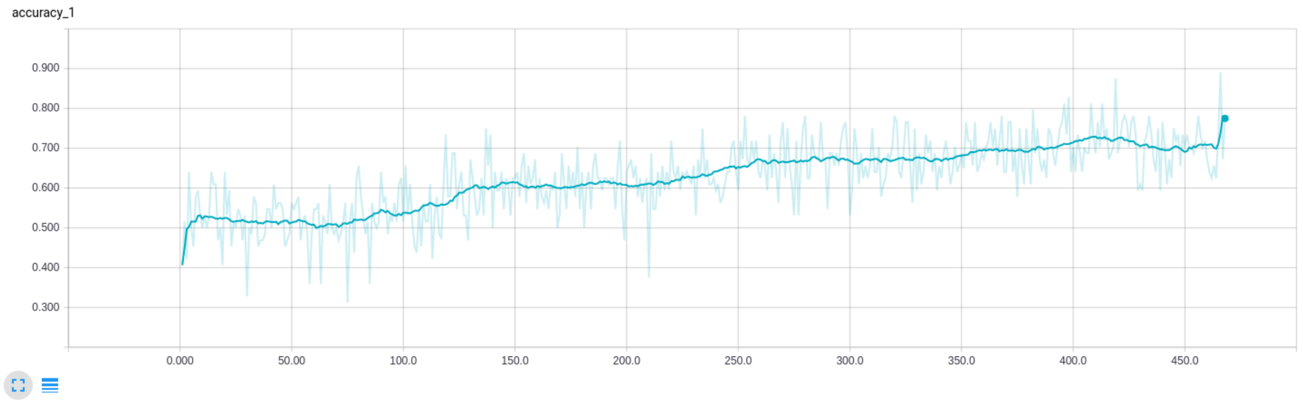
| Class B | Selected B | Not Selected B | Total |
|---|---|---|---|
| Actual B | **1491** | 409 | 1900 |
| Not B | 425 | **5275** | 5700 |
| Total | 1916 | 5684 | 7600 |

| Class C | Selected C | Not Selected C | Total |
|---|---|---|---|
| Actual C | **1464** | 436 | 1900 |
| Not C | 425 | **5275** | 5700 |
| Total | 1889 | 5711 | 7600 |

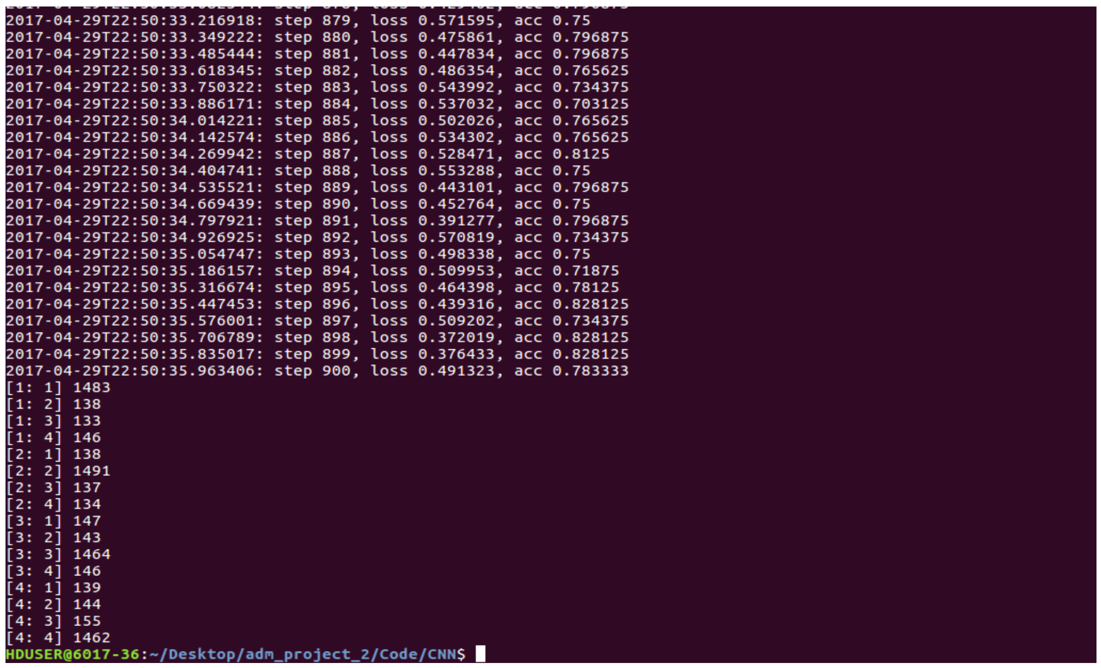
| Class D | Selected D | Not Selected D | Total |
|---|---|---|---|
| Actual D | **1462** | 438 | 1900 |
| Not D | 426 | **5274** | 5700 |
| Total | 1888 | 5712 | 7600 |

Precision: 0.77
Recall: 0.78
F1-Measure: 0.775

Testing Accuracy: 78%



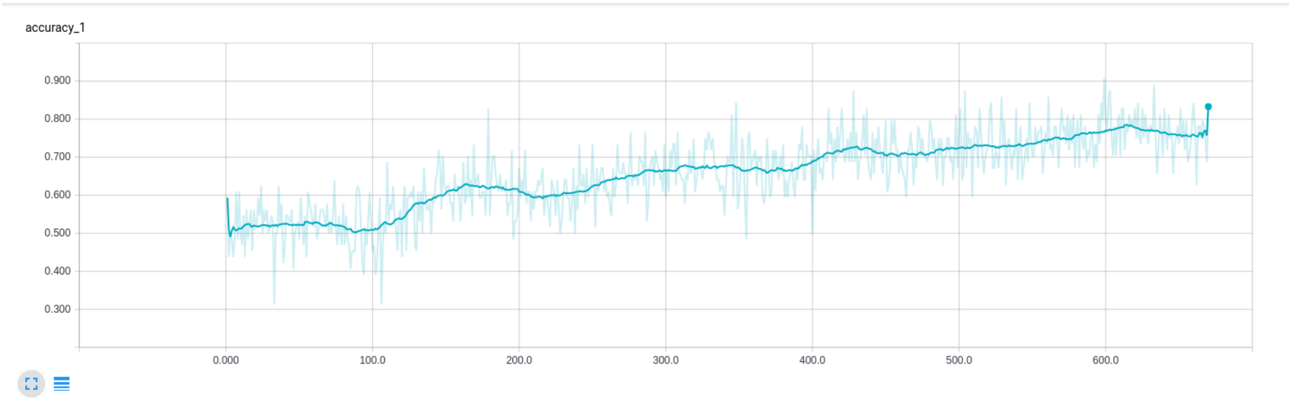
screen shot for the execution is shown below:



```
2017-04-29T22:50:33.216918: step 879, loss 0.571595, acc 0.75
2017-04-29T22:50:33.349222: step 880, loss 0.475861, acc 0.796875
2017-04-29T22:50:33.485444: step 881, loss 0.447834, acc 0.796875
2017-04-29T22:50:33.618345: step 882, loss 0.486354, acc 0.765625
2017-04-29T22:50:33.750322: step 883, loss 0.543992, acc 0.734375
2017-04-29T22:50:33.886171: step 884, loss 0.537032, acc 0.703125
2017-04-29T22:50:34.014221: step 885, loss 0.502026, acc 0.765625
2017-04-29T22:50:34.142574: step 886, loss 0.534302, acc 0.765625
2017-04-29T22:50:34.269942: step 887, loss 0.528471, acc 0.8125
2017-04-29T22:50:34.404741: step 888, loss 0.553288, acc 0.75
2017-04-29T22:50:34.535521: step 889, loss 0.443101, acc 0.796875
2017-04-29T22:50:34.669439: step 890, loss 0.452764, acc 0.75
2017-04-29T22:50:34.797921: step 891, loss 0.391277, acc 0.796875
2017-04-29T22:50:34.926925: step 892, loss 0.570819, acc 0.734375
2017-04-29T22:50:35.054747: step 893, loss 0.498338, acc 0.75
2017-04-29T22:50:35.186157: step 894, loss 0.509953, acc 0.71875
2017-04-29T22:50:35.316674: step 895, loss 0.464398, acc 0.78125
2017-04-29T22:50:35.447453: step 896, loss 0.439316, acc 0.828125
2017-04-29T22:50:35.576001: step 897, loss 0.509202, acc 0.734375
2017-04-29T22:50:35.706789: step 898, loss 0.372019, acc 0.828125
2017-04-29T22:50:35.835017: step 899, loss 0.376433, acc 0.828125
2017-04-29T22:50:35.963406: step 900, loss 0.491323, acc 0.783333
[1: 1] 1483
[1: 2] 138
[1: 3] 133
[1: 4] 146
[2: 1] 138
[2: 2] 1491
[2: 3] 137
[2: 4] 134
[3: 1] 147
[3: 2] 143
[3: 3] 1464
[3: 4] 146
[4: 1] 139
[4: 2] 144
[4: 3] 155
[4: 4] 1462
HDUSER@6017-36:~/Desktop/adm_project_2/Code/CNN$
```

2. Accuracy and Loss observed after changing the following parameters:

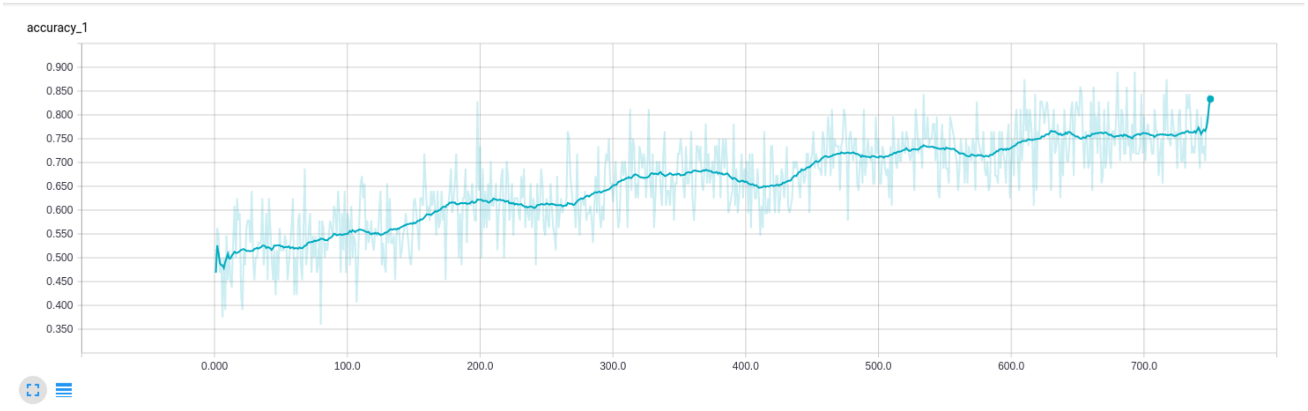
Activation Function = RELU



Precision: 0.82
Recall: 0.8
F1-Measure: 0.81
Testing Accuracy: 83%

3. Accuracy and Loss observed after changing the following parameters:-

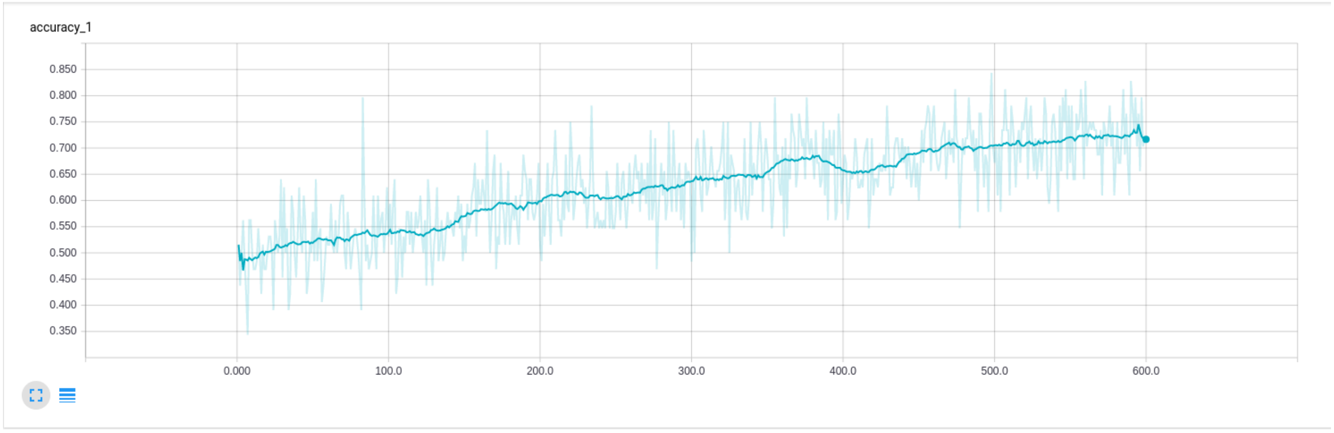
Activation Function = RELU
Dropout Keep Probability = 0.5



Precision: 0.84
Recall: 0.86
F1-Measure: 0.85
Testing Accuracy: 84%

4. Accuracy and Loss observed after changing the following parameters: -

Activation Function = RELU
Dropout Keep Probability = 0.5
Epochs : 10
Decay rate = 0.90



Precision: 0.725
Recall: 0.68
F1-Measure: 0.69
Testing Accuracy: 72%

## 5. Conclusions

Making CNN deep improves the accuracy if trained at correct range of parameters. Keeping large too large or too small value of dropout keep probability reduces the accuracy and RELU activation performed better than TANH activation function for the text classification problem. Increasing the epochs usually improves the accuracy but keeping too high value of epochs without adding dropout layers can lead to overfitting.

## 6. References

[1] http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp.

[2] Alexis Conneau, Holger Schwenk, Loïc Barrault, Yann Lecun, "Very Deep Convolutional Networks for Text Classification" arXiv:1606.01781 [cs.CL].

[3] Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014), 1746–1751.