TEXT CLASSIFICATION USING DEEP NEURAL NETWORKS

A PROJECT REPORT

Submitted by

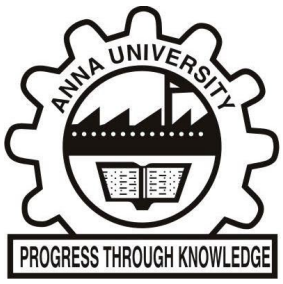
M. HARISH	**2014503517**
H. TEJA SURYA	**2014503550**
R. THARANIDHARAN	**2014503551**

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING



MADRAS INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY::CHENNAI 600044

APRIL 2018

TEXT CLASSIFICATION USING DEEP NEURAL NETWORKS

A PROJECT REPORT

Submitted by

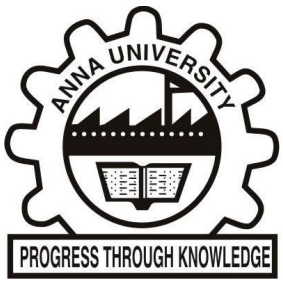
M. HARISH	**2014503517**
H. TEJA SURYA	**2014503550**
R. THARANIDHARAN	**2014503551**

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING



MADRAS INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY::CHENNAI 600044

APRIL 2018

ANNA UNIVERSITY : CHENNAI 600044

BONAFIDE CERTIFICATE

Certified that this project report titled "**TEXT CLASSIFICATION USING DEEP NEURAL NETWORKS**" is a bonafide work done by **HARISH M (2014503517), TEJA SURYA H (2014503550) and THARANIDHARAN R (2014503551)** under my supervision in partial fulfilment for the award of the degree of Bachelor of Engineering in Computer Science and Engineering. Certified further that to the best of my knowledge the work reported here in does not form part or full of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion to this or any other candidate.

SIGNATURE

Dr. P. ANANDHAKUMAR
HEAD OF THE DEPARTMENT
Professor
Department of Computer Technology
Madras Institute of Technology
Anna University
Chennai - 600 044.

SIGNATURE

Dr. S. THAMARAI SELVI
PROJECT SUPERVISOR
Professor
Department of Computer Technology
Madras Institute of Technology
Anna University
Chennai - 600 044.

ACKNOWLEDGEMENT

We are highly indebted to our respectable Dean, **Dr. A. RAJADURAI** and to our reputable Head of the Department **Dr. P. ANANDHAKUMAR**, Department of Computer Technology, MIT, Anna University for providing us with sufficient facilities that contributed to success in this endeavour.

We would like to express our sincere thanks and deep sense of gratitude to our Supervisor, **Dr. S. THAMARAI SELVI** for her valuable guidance, suggestions and constant encouragement which paved way for the successful completion of this phase of project work.

We sincerely thank all the Panel members **Dr. P. JAYASHREE, Dr. P. VARALAKSHMI** and **Dr. R. KATHIROLI** for the valuable suggestions.

We extend our sincere thanks to **Mr. P. KARTHIKEYAN**, Center of Advanced Computing Research and Education, for providing valuable guidance.

We would be failing in our duty, if we forget to thank all the teaching and non-teaching staff of ours department, for their constant support throughout the course of our project work.

HARISH M (2014503517)

TEJA SURYA H (2014503550)

THARANIDHARAN R (2014503551)

ABSTRACT

Text Classifiers are universal systems that help us in categorizing document collections into one of the predefined set of classes based on the common features among them. Text Classification is one of the fundamental tasks in the field of information retrieval because when searching for information having certain characteristics, grouping relevant data is useful in retrieving them. Existing systems that uses probability based or machine learning based methods to estimate the similarity among documents provide importance only to the occurrence of a particular set of terms rather than context based representation. After the advent of convolutional and recurrent neural networks, the content based intelligent classification is being given high priority. But the biased feature learning by recurrent networks and window restricted feature learning by convolutional networks restrict the performance of these networks. We propose three variants of deep learning based text classification system for combining the convolutional and recurrent networks in a parallel manner that overcomes their individual limitations. We also propose a novel activation function Inverse Exponential Linear Unit (IELU) that allows the network to learn more features in comparison with traditional activation functions like Rectified Linear unit and Exponential Linear unit. The throughput of the system is analysed by considering a variety of datasets and performance metrics into consideration. It has been inferred that all the three parallel architectures have higher performance when compared with the existing classification systems. Also, our IELU performs better than the widely used activation functions in text classification application.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

KNN	K-Nearest Neighbor
RNN	Recurrent Neural Network
LSTM	Long Short Term Memory
GRU	Gated Recurrent Unit
CNN	Convolutional Neural Network
TANH	Hyperbolic Tangent
RELU	Rectified Linear Unit
ELU	Exponential Linear Unit
LRELU	Leaky Rectified Linear Unit
IELU	Inverse Exponential Linear Unit
RCPSNN	Recurrent Convolutional Parallel Sequential Neural Network
RCPCNN	Recurrent Convolutional Parallel Cross Neural Network
RCPSCNN	Recurrent Convolutional Parallel Sequential Cross Neural Network
RCSNN	Recurrent Convolutional Sequential Neural Network

CHAPTER 1
INTRODUCTION

1.1 OVERVIEW

Chapter 1 introduces the basic ideologies and concepts used todevelop the proposed Text Classification system. The system proposed to develop aims tobe a deep learning based classifier that is useful for segregating chunks of documents into various categories based on the content of the documents. Traditional text classifiers make use of the important keywords and phrases for the purpose of categorization which can be misleading at times. The proposed system aims to exploit the temporal and spatial relationship between the various wordsand generates a higher level abstract representation to improve the accuracy in text classification.

1.2 TEXT CLASSIFIER

Text classification is one of the basic and indispensableprocesses in Natural Language Processing. The objective of Text classification is to allocatedefined classes to pre-processed text documents. The classes are allocated from a previously represented hierarchy.A text classifier integrates factual archive classification with rule-based sifting, which permits to get an accurate representation of the text document and its relation to that class. It has a large set of applications, includingdocument indexing, web searching, information filtering, tag suggestion and spam detection.Hence it has captivated an incredible amount of focus from a lot of research scientists.

1.2.1 BENEFITS

Huge amount of unused data and information are produced regularly through financial, scholastic and social exercises, many with noteworthy potential financial and societal esteem. In order to utilize the resources better,

these data must be compartmentalized for quick retrieval and use it for relevant processing. For exemplification, consider a news article repository that contains millions of articles. For a typical lookup program, the time taken to locate a particular document is directly proportional to the size of the repository if no categorization is applied. But after segregation, the time taken to search a particular document reduces exponentially based on the levels of classification applied over the corpus. This practical ramification is highly preferred to reduce the query retrieval time in search engines, crawlers and many other information retrieval systems.

1.3 TYPES OF TEXT CLASSIFIER

Numerous algorithms have been proposed throughout the ages to process and classify text documents. The algorithms that involve Neural Networks outperform the algorithms that don't in terms of accuracy, precision and recall.Basically, there are two kinds of text based classifiers, the Word-based classifier and the Content-based classifier. Most of the algorithms that have been used before Neural Networks involved Word-based classification. But almost all the machine learning based models aim to exploit the content to extrapolate the significance. The various algorithms for text classification are listed as follows.

1. Naive Bayes Classifier
2. Decision tree based classifier
3. Support vector machine classifier
4. KNN based classifier
5. Multinomial Logistic regression
6. Supervised Latent Dirichlet Allocation
7. Hidden Markov Model
8. Neural Network based classifier

1.3.1 NAIVE BAYES CLASSIFIER

When we consider the field of machine learning, Naive Bayes classifiers are an elicit group of basic probabilistic based classifiers based on applying Bayes' hypothesis with naive independent suspicions between the highlights. Naive Bayes remains a prevalent (pattern) strategy for content categorization, the issue of judging records as having a place in one category or the other (such as spam or authentic, sports or legislative issues, etc.) with word frequencies as the highlights. Naive Bayes classifiers are exceedingly adaptable, demanding various featuresdirect in the number of factors in a learning issue.

Naive Bayes is a tool that is hugely employed for designingprobabilistic classifiers: models that relegate class names assigned as vectors of highlight values, where the class names are drawn from a few limited sets. It is not a single calculation for preparing such classifiers, but a family of calculations based on a common guideline: all Naive Bayes classifiers accept that the value of a specific feature is free of the value of any other highlight, given the class variable. For illustration, a fruit may be considered to be an apple in the event that it is ruddy, circular, and approximately 10 cm in distance across. A credulous Bayes classifier considers each of these highlights to contribute freely to the likelihood that this natural product is an apple, in any case of any conceivable relationships between the color, roundness, and distance across highlights.

For a few sorts of likelihood models, Naive Bayes classifiers can be prepared exceptionally effectively in an administered learning setting. In numerous practical applications, parameter estimation for Naive Bayes models employments the strategy of most extreme probability; in other words, one can work with the Naive Bayes demonstrate without tolerating Bayesian likelihood.

1.3.2 DECISION TREE BASED CLASSIFIER

Decision tree learning uses a decision basedbranching structure to go from understanding around a thing that is dividedinto branches to conclusions almost the item's target value. It is one of the prescient demonstrating approaches utilized in insights, information mining and machine learning. The model that involves trees in which the final value chooses an independent value set called the classification trees; in these tree structures, leaves refer to class names and branches representconfluence of prime features that lead to those classvariables.

Decision trees where the target variable can take ceaseless values (ordinarily genuine numbers) are called regression trees. During decision examination in order to outwardly and unequivocally speak to decision and decision making, we use a decision tree. In information mining, a decision tree depicts information.

Decision tree learning is the development of a decision tree from class-labelled prepared from the documents. A decision tree is a flow-chart-like structure, where each inside (non-leaf) hub indicates a test on a property, each branch speaks to the outcome of atest, and each terminal node holds a class variable.

The concept of greedy based programming is used at each hub to locally make optimal choices and hence by which the decision-tree learning calculations are done. To decrease the greedy impact of locally-optimality a few strategies such as the dual information distance (DID) tree were proposed. For information counting categorical factors with distinctive numbers of levels, information gain in choice trees is one-sided in favor of those properties with more level.

1.3.3 SUPPORT VECTOR MACHINE CLASSIFIER

In the field of machine learning, when we consider a particular set of model termed as support vector machines which are administered learning models with related learning calculations that can analyze the information utilized for classification and regression examination. For a predefined set of training examples, each denoted as having one or two of the given categories, the SVM model performs some predefined set of calculations and assigns the values to one or the other category, henceforth making it as a non-probabilistic classifier.

Unused illustrations are at that point mapped into that same space and anticipated to have a place to a category based on which side of the branch they drop. SVMs are supportive in content and hypertext categorization as their application can altogether decrease the requirement for labelled training occurrences in both the standard inductive and transductive settings.

1.3.4 KNN BASED CLASSIFIER

In pattern matching, for the purpose of classification and regression, we use the non-parametric strategy k-nearest neighbor calculation (k-NN). In both cases, we have the input as a set of the k closest preparing outlines in the representation space. The yield set of highlight depends on the reason of utilization such as whether k-NN is utilized for classification or relapse: In k-NN classification, the output is a one-hot embedded class representation. An object is classified by a larger part vote of its neighbors, with the question being doled out to the lesson most common among its k closest neighbors (k is a positive number, ordinarily little). In case k = 1, at that point the protest is basically assigned to the class of that single closest neighbor.

In k-NN regression, the expected is the approximate property value for the expected object. This value is the normal of the values of its k closest

neighbors. Since only a locally approximate work is done by the algorithm till classification computation purpose, k-NN can be termed as a kind of lazy learning.

In either the case of multivariate classification and regression, the prime methodology of algorithm is to allot random feature value to the surrounding particles in the space so that the value of each member is influenced by the values of its surrounding members. The neighbors are taken from a set of objects for which the lesson or the object based property value is viable and visible. This can be thought of as the training set for the calculation, in spite of the fact that no unequivocal preparing step is required. An idiosyncrasy of the k-NN calculation is that it is highly sensitive and volatile to the surrounding information structure.

1.3.5 MULTINOMIAL LOGISTIC REGRESSION

When the consideration is shifted to the field of statistics and machine learning, multinomial logistic regression is a classification oriented strategy that generalizes calculated regression to multiclass problems. That is, are presentation that is utilized to anticipate the probabilities of the distinctive possible results of a categorically disseminated subordinate variable, given a set of autonomous factors. Multinomial logistic regression is utilized when the subordinate variable at address is ostensible (identically categorical, meaning that it falls into any one of a set of categories that cannot be requested in any significant way) and for which there are more than two categories. The multinomial calculated show accept that information are case particular; that is, each autonomous variable has a single esteem for each case. The multinomial logistic model too assumes that the subordinate variable cannot be perfectly anticipated from the independent factors for any case.

1.3.6 SUPERVISED LATENT DIRICHLET ALLOCATION (SLDA)

In natural language processing, Latent Dirichlet allotment (LDA) is a generative measurable demonstrate that permits sets of perceptions to be clarified by imperceptible groups that clarify why a few parts of the information are compared. For case, on the off chance that observations are words collected into documents, it sets that each record is a blend of a little number of points and that each word's creation is attributable to one of the document's topics. In LDA, each report may be seen as a blend of different points where each record is considered to have subjects that are assigned to it through LDA.

The LDA model is profoundly secluded and can subsequently be effectively amplified. The main field of interest is modelling relations between points. This is accomplished by utilizing another distribution on the simplex instead of the Dirichlet. Nonparametric extensions of LDA incorporate the various Latent Dirichlet process mixture model, which permits the number of themes to be unbounded and learnt from information.

1.3.7 HIDDEN MARKOV MODEL

Hidden Markov Model (HMM) consists of a bunch of secluded set of states or positions within a standard Markov Model and this kind of framework is hence termed as hidden in nature. It is one of the simplest and easily configurable dynamic Bayesian network. When a normal Markov model is subjected to interpretation, then it is patent that the state and its probability based transitions are viable to an observer. But in case of the hidden kind, the state is unnoticeable, but the output is contingent on the state. Each state has likelihood dissemination over the conceivable yield tokens. In this manner, the grouping of tokens produced by an HMM gives a few data about the arrangement of states; this is also known as design theory, a subject of linguistic use induction.

1.3.8 NEURAL NETWORK BASED CLASSIFIER

Previous sentence level categorizers involved word based analysis rather than content based interpretation. After the advent of neural network based classifier, content based categorization has been an integral part of supervised natural language processing. The Long short term memory based recurrent neural network interpret input data as a time series, hence it is helpful in capturing the temporal relation between the data in the document. But the main problem with this is the bias assigned by the network to latter input in comparison with the previous input states.

The neural networks use embedded representations of words so that text can be numerically processed by employing convoluted mathematical operations. Also the convolution based neural network is helpful in getting a higher level representation by extrapolating n-gram features and thus the spatial relation between the input is preserved. But the problem involved in this is the extent of the input size. The activation functions employed by the network are mainly used for the purpose of normalizing the input parameters to a viable scale. In spite of having the drawbacks of long training times to process data, neural networks are responsible for providing the state of the art performance in natural language processing.

1.4 DEEP NEURAL NETWORK

A Deep neural network (DNN) is typically an advancement of the basic Artificial Neural Network with several hidden layers between the input and output layers (Fig 1.1). DNN designs create compositional models where the object is communicated as a layered structure of primitives. The additional layers empower piece of highlights from bringing down layers, possibly demonstrating complex information with less units than a correspondingly performing shallow system.

Profound designs incorporate numerous variations of a couple of fundamental methodologies. It isn't generally conceivable to look at the execution of numerous structures, unless they have been assessed on similar informational collections.

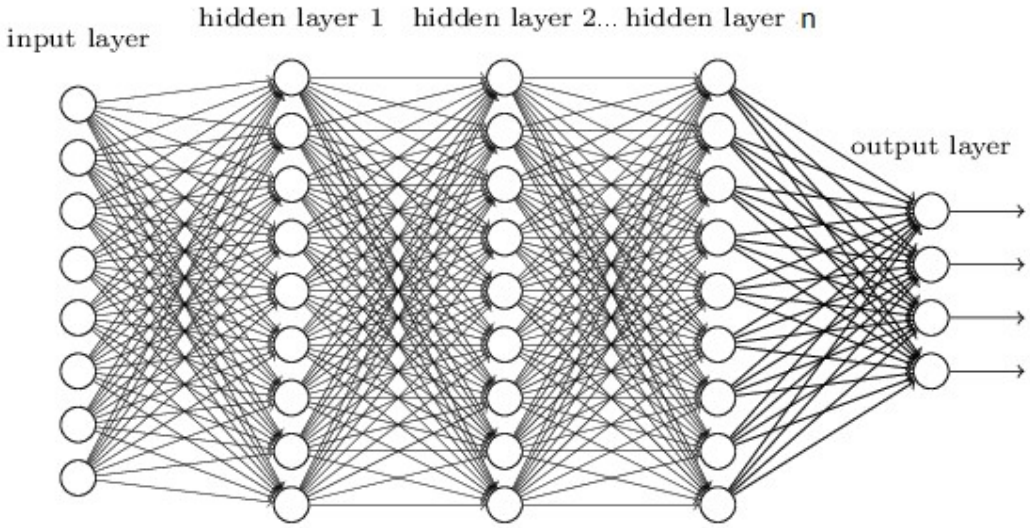


Fig 1.1 ARCHITECTURE OF DEEP NEURAL NETWORK [53]

1.5 APPLICATIONS OF DEEP NEURAL NETWORK

DNNs are ordinarily feed forward systems in which information conversions from the information layer to the yield layer without circling back. Profound neural systems (DNNs), in which information would confluence be able to toward any path, are utilized for suggestions, for example, dialect demonstrating. Long short-term memory is particularly effective for this use. DNNs are also used in several fields like detecting weather arrangements, speech processing, natural language processing, recommendation system and Bioinformatics. Convolution deep neural networks (CNNs) are used in computer vision such as object recognition.

1.6 CHALLENGES OF DEEP NEURAL NETWORK

Likewise with ANNs, many issues can emerge with prepared DNNs. Two normal issues are over-fitting and calculation time. DNNs are inclined to over-fitting due to the additional layers of deliberation, which enable them to show uncommon conditions in the preparation information. On the other hand dropout regularization arbitrarily overlooks units from the concealed layers amid preparing. This avoids uncommon conditions.

DNNs must consider many preparing parameters, for example, the size, the learning rate and introductory weights. Clearing through the parameter space for ideal parameters may not be attainable because of the cost in time and computational assets. Different traps, for example, grouping accelerate calculation. The expansive handling throughput of GPUs has created noteworthy speedups in preparing, in light of the fact that the lattice and vector calculations required are appropriate for GPUs.

1.7 ORGANISATION OF THE REPORT

Chapter 1 gives a brief introduction of the concepts and techniques used. Chapter 2 describes the various works previously done in the segregation of text documents. Chapter 3 gives the detailed design and description of the proposed Deep learning based text classifier. Chapter 4 describes about the implementation details. Chapter 5 presents the conclusion and the future works.

1.8 CHAPTER SUMMARY

In this chapter, the overview of the various concepts and techniques currently existing in the field of text classification were discussed. In the next chapter, literature survey of the existing works on document classification and categorization will be discussed in detail.

CHAPTER-2
LITERATURE SURVEY

2.1 OVERVIEW

This chapter presents a literature survey on the existing works on Text processing and Document grouping. The survey explores the works on various implementations of Text Classification for several implications.

2.2 LITERATURE REVIEW

2.2.1 BAYES CLASSIFIERS

Haiyi Zhang, Di Li[9] have designed a Naive Bayes based text classifier for document categorization. This paper uses pre-classified emails for the purpose of training the spam email detector. The detector is able to decide whether an email is a spam email or an ordinary email, based on the features learnt from the training set. Classification is done based on the human chosen features in a document. The class of a text is based on the prior conditional existence of these features. The features are processed using the Probability Inference task, Joint probability, Conditional Probability over inference to classify the document. The dataset used are Pre-classified E-Mails while evaluation metrics used are Precision, Recall, F-1 Score. But identifying these features becomes difficult as the data quantity increases. Accuracy is poor compared with the KNN models.

Shahnaj Parvin, Md. Delowar, Md. Nadim, Sayed Golam, Tangina Sultana [10] has enhanced the performance of Naive Bayes Classifier by augmenting extra weights with less number of training examples. In that paper a compelling and proficient strategy for classifying content records in arranging to provide attainable data recovery utilizing Naive Bayes calculation has been proposed. A Weight Network is presented in preparing content record which is a combination of Term Recurrence and Converse Course Recurrence and

afterward this term is fuelled by critical number and included with the back esteem amid the forecast time of Naive Bayes calculation to set up a superior and productive execution of the classification assignment. The combinational term Weight Framework gives an additional weight and equalizations the weight where fundamental. Performance is better in comparison with a traditional Bayes classifier. The dataset used are Manual Dataset created based on News articles while the evaluation metrics are Precision, Recall, F-1 Score. But the Feature extraction becomes difficult as data quantity increases. Accuracy is less than that of the Deep learning based models.

2.2.2 MACHINE LEARNING CLASSIFIERS

Srinivasan Ramaswamy [11] in his paper, meticulous text classification proposed a decision tree based approach that incorporated Support vector machine for clustering purposes. The paper talks about around combining Support Vector Machine and decision trees for multi lesson content classification. Support Vector Machines are prepared for each lesson at each level of the tree and the SVM which is more effective in foreseeing a course at that level is chosen as the choice in that hub. Hence a tree is built with diverse SVM in each hub. And the tree built is utilized for classifying the multiclass content. Comes about had appeared that this strategy works comparably way better than the other classifiers like basic SVM and Naive Bayes. This approach combines Support Vector Machine with Decision Tree. A tree is built with diverse SVM in each hub. The tree developed is utilized for classifying the multiclass content and in each hub the choice is made by SVM. The dataset used are 20Newsgroup Dataset. While the evaluation metrics are Precision, Recall, F-1 Score. Although, the tree is restricted to one side and hence the misclassified instances i.e. False positives are not considered for reclassification into the correct class. Accuracy is poor in comparison with the KNN based models.

Jing Zhong Wang and Xia Li [12] on their paper opinionated a K-nearest neighbor algorithm for performing text classification. This paper analyzes the points of interest and drawbacks of KNN algorithm and presents a progressed KNN algorithm (WPSOKN) for content classification. It is based on molecule swarm optimization which has the capacity of arbitrary and coordinated global look inside preparing the report set. Besides, it reduces the impact of individual particles from the overall. Each text data is matched with one of the k-closest clusters. If a text data doesn't match with any k neighbours, the data is removed. Dataset used are Sohu news corpus and the evaluation metrics used are Precision, Recall, F-1 Score. Deciding the K value is difficult. Classification becomes complicated as the number of data increases. The property of the cluster is dependent on the members of the cluster.

Yun Lin, Jie Wang[13] have research on combining Support vector machines and KNN algorithm to perform document categorization. This paper has been proposed based on the contingency over the basic support vector machine algorithm. The SVM-KNN algorithm for text classification has been put forward which amalgamates both the support vector machine algorithm and KNN algorithm. The SVM-KNN algorithm can improve the performance of classifier by the feedback and improvement of classifying prediction probability. SVMs maximize margin between disparate data while the KNNs cluster data based on similarity. Combining SVMs and KNNs, clusters of KNNs separated by SVMs are generated. The performance is better than individual SVM or KNN. Dataset used is Sohu Laboratory Corpus while evaluation metrics used are Precision, Recall, F-1 Score. Accuracy is poor in comparison with the Neural Network based models.

2.2.3 DEEP LEARNING CLASSIFIERS

Mohammadreza Ebrahimi, Ching Y.Suen, Olga Ormandjieva[3] in his paper have elaborated on howdeep convolutional neural networks can be

rendered for detecting intolerable conversations in social media. In this paper, application of a profound learning strategy to the programmed recognizable proof of ruthless chat discussions in huge volumes of chat logs is portrayed. A classifier based on Convolutional Neural Arrangement (CNN) is utilized to address this issue space. The proposed CNN engineering beats other classification strategies that are common in this space including Support Vector Machine (SVM) and standard Neural Arrangement (NN) in terms of classification performance, which is measured by F1-score.

Siwei Lai, Liheng Xu, Kang Liu, Jun Zhao[14] have implemented are current based convolutional neural networks for Text Classification. This paper proposes a repetitive convolutional neural arrangement for content classification without human-designed highlights. In our demonstrate, we apply a repetitive structure to capture relevant data as distant as conceivable when learning word representations, which may present significantly less commotion compared to conventional window-based neural systems. The model captures left, right context using bidirectional RNN. Maxpool layer combines output from the RNN layer. The unbiased model of CNN is applied using the maxpool layer. Parameters are initialized using uniform distribution. Dataset used are 20NewsGroup, Fudan Set, ACL Anthology, Stanford Sentiment. Evaluation Metrics used are Precision, Recall, F-1 Score. Splitting sentences into character level and word level is difficult. Unknown words in corpus are unhandled.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, Eduard Hovy[42] had designed a hierarchy based attention network for document classification. This demonstrate has two particular characteristics which incorporate various levelled structure that mirrors structure of records and two levels of consideration instruments connected at the word and sentence-level, empowering it to go to differentially to more and less vital substance when building the report representation. Dataset used are Yelp review, Amazon reviews, IMDB reviews, Yahoo answers and the evaluation metrics used are

Precision, Recall, F-1 Score. Though word attention is combined with sentence attention, unknown vocabulary is not handled by the model.

Yujun Zhou, Bo Xu, Jiaming Xu, Lei Yang, Chang liang Li, Bo Xu[15] in their paper designed a Compositional Recurrent Neural Networks for document compartmentalization in Chinese language. For the purpose of diminishing the affect of word division and improve the in general performance of Chinese brief content classification framework, this paper propose a crossover show of character-level and word-level highlights based on recurrent neural organize (RNN) with long short-term memory (LSTM). By coordination of character-level highlight into word-level highlight, the lost semantic data by the blunder of word division will be developed, in the mean time the off-base semantic pertinence will be decreased. The last highlight representation is that it stifled the mistake of word division in the case of keeping up most of the semantic highlights of the sentence. The whole model is finally trained end-to-end with supervised Chinese short text classification task. Softmax – Conditional probability is used to estimate the class. Dataset used are Chinese News titles, Douban movie reviews while the evaluation metrics used are Precision, Recall, F-1 Score. Lot of Training data is required. RNNs take a lot of training time. RNNs allow latter words dominate than earlier words.

Julio Reyes, Diego Ramırez, Julio Paciello[16] in their paper demonstrated program code classification based on programming languages using Deep neural networks. This paper proposes the use of a Deep Learning technique, the Long Short-Term Memory (LSTM) recurrent neural network, for the automatic classification of source code archives by programming language. Experiments show that this simple recurrent neural network architecture gives promising results in accuracy compared to the Naive Bayes classifier, currently used by Linguist, one of the most popular programming language classifiers. Programming codes are classified using LSTM classifiers. Cross Entropy is used along with Adam optimizers. Leave one out Cross validation is used

during training and testing. Dataset used are Rosetta Dataset, Github Source Codes. Evaluation Metrics used are Precision, Recall, F-1 Score.Classification is based on the syntax of the code. Regular Expressions and other heuristics are not used.

Kamran Kowsari, Donald E. Brown, Mojtaba Heidarysafa, Kiana Jafari Meimandi, Matthew S. Gerber, and Laura E. Barnes[44] have designed atwo level hierarchical deep learning based text classifier. This paper mainly tries to compartmentalize the level of text documents into levels of classification to establish hierarchical structure in text classification. Sequential LSTM layers are used to classify document on various levels. Dataset used are Web of Science WOS – 11967, 46985, 5736 and the evaluation metrics are Precision, Recall, F-1 Score.

Yuan Luo[2] for classifying relations in clinical notes designed a recurrent neural network in his paper. In this paper, the LSTM model of recurrent neural networks is used for classifying relations in clinical notes. Two types of models are defined based on the input word vectors as Sentence based and segment based model. The models are compared with existing word embedding based models and their performance are analysed.

Meng Joo Er, Yong Zhang, Ning Wang, Mahardhika Pratama[18] in their journal constructed an attention pooling-based convolutional network for sentence modelling. In this paper, a modern pooling plot named Attention Pooling is proposed to hold the most noteworthy data at the pooling arrangement. A halfway sentence representation produced by the bidirectional long short-term memory is utilized as a reference for nearby representations created by the convolutional layer to get attention weights. The sentence representation is shaped by combining neighbourhood representations utilizing obtained attention weights. The model can be prepared end-to-end with constrained hyper-parameters. Comprehensive data is extricated by the modern pooling plot and the combination of the convolutional layer and the

bidirectional long-short term memory. The model can implicitly separate the sentences from different classes.

Mark J. Berger[43] in his paper designed semantic word vectors for multi-class document classification. With the advent of powerful semantic embeddings, this paper expands on the query of how word embeddings and word orderings can be used to improve multi-label learning. Specifically, this paper also explore how both a convolutional neural network (CNN) and a recurrent network with a gated recurrent unit (GRU) can independently be used with pre-trained word2vec embeddings to solve a large scale multi-label text classification problem. Word2Vec and GloVe is used to consider word order. Glorot uniform is used for weight initialization along with RMS Prop descent and Soft clip gradient. Dataset used are BioASQ Challenge while the evaluation metrics used are Precision, Recall, F-1 Score. Word Embeddings doesn't consider the context of Unknown words. GRU network is not suitable to handle long term dependencies.

Tao Chen, Ruifeng Xu, Yulan He, Xuan Wang[1] have designed a deep learning based sentence classifier by combining recurrent and convolutional neural network. Previous sentence level categorizers involved word based analysis rather than content based interpretation. After the advent of deep neural network based classifier, content based categorization has been an integral part of supervised natural language processing. The Long short term memory based recurrent neural network interpret input data as a time series, hence it is helpful in capturing the temporal relation between the data in the document. But the main problem with this is the bias assigned by the network to latter input in comparison with the previous input states. On the other hand, the convolution based neural network is helpful in getting a higher level representation by extrapolating n-gram features and thus the spatial relation between the input is preserved. This paper thus aims to interpolate the spatial and the temporal relation in a sequential manner. Thus the output of the temporal network is fed

into the spatial network. Datasets used are a combination of document collections belonging to various languages like English, Spanish, French, Russian, Dutch and Turkish. The evaluation metrics are precision, recall, accuracy. The disadvantage of combining in this sequential manner leads to the loss of temporal features when it is fed into the spatial network.

2.3 INFERENCE

From the existing literature survey it has been inferred that techniques involving Decision trees outperformed Naive Bayes based algorithms. Though there have been significant improvements in text classification after introducing Support vector machine and combining it with decision tree, it suffered from drawbacks such as long training time and limited precision. This was overcome after the advent of machine learning algorithms like K-nearest neighbour algorithm. But the precision and recall of the output still remained limited for processing large datasets. This is because of the bag of words approach implemented by the traditional classifiers. The Deep learning algorithms were empowered to handle large repository and the accuracy of the model has also been improved. Deep neural network's trade off between precision and training time were also analysed. The idiosyncrasy and purpose of spatial and temporal relation among words were also reviewed. The recurrent neural network based classifiers inspite of being able to handle long term sequences, has the drawback of giving importance to terms occurring later in the sequence rather than earlier which leads to a bias. Though convolutional neural networks don't provide such bias, they can handle only n-grams of words at a time. This window based approach results in the loss of information relating to long term dependencies. Hence the motivation behind our proposed work is to overcome the limitations of both the convolutional and recurrent neural network and thereby improve the performance of the text classification system. In next chapter, the proposed deep learning architecture and its components will be discussed in detail.

CHAPTER 3
PROPOSED TEXT CLASSIFICATION SYSTEM

3.1 OVERVIEW

Chapter 3 describes the architecture of the proposed text classifier. The modules of the system along with the workflow are enunciated. The various components of the architecture along with the techniques used are also discussed.

3.2 DESCRIPTION OF THE PROPOSED WORK

The proposed text classification system is designed as a deep learning based intelligent classifier that is useful for processing and segregating huge chunks of text documents. When a large section of text corpus is given to the classifier model, the document is split into training, validation and testing dataset samples. The training phase is used to ensure that the model analyzes the content of the input, to gather features and interpolate the relation among the feature extracted with respect to the class to which each sample belongs to.

In the validation phase, the validation data is fed into to the trained model to analyze the performance of the model on the data to ensure that the training features don’t narrow down the feature range of the values in the network model and the generality of the features remains intact. The testing phase is the actual challenge posed to the model where the practical performance of the model is measured and analyzed. First, the input data is pre-processed and the output is one-hot encoded. After that the feed-forward phase of the neural network is instantiated followed by back-propagation phase for the purpose of learning the features of the network. Finally the actual output is evaluated and compared with the expected output to analyze the performance of the network.

3.3 CONVOLUTIONAL NEURAL NETWORK

In Deep learning, the convolutional neural network is a type of deep neural network that uses spatial analytics to capture features. Within a CNN there are several kinds of layers each with a distinct function. When the CNN is perceived in a mathematical dimension, there exists a cross-correlation among the inputs and the output generated by the network. The various layers are depicted in Fig.3.1. The various kinds of layers in Convolutional Neural Network are,

1. Convolution Layer
2. Pooling Layer
3. Fully Connected Layer
4. Dropout regularisation Layer

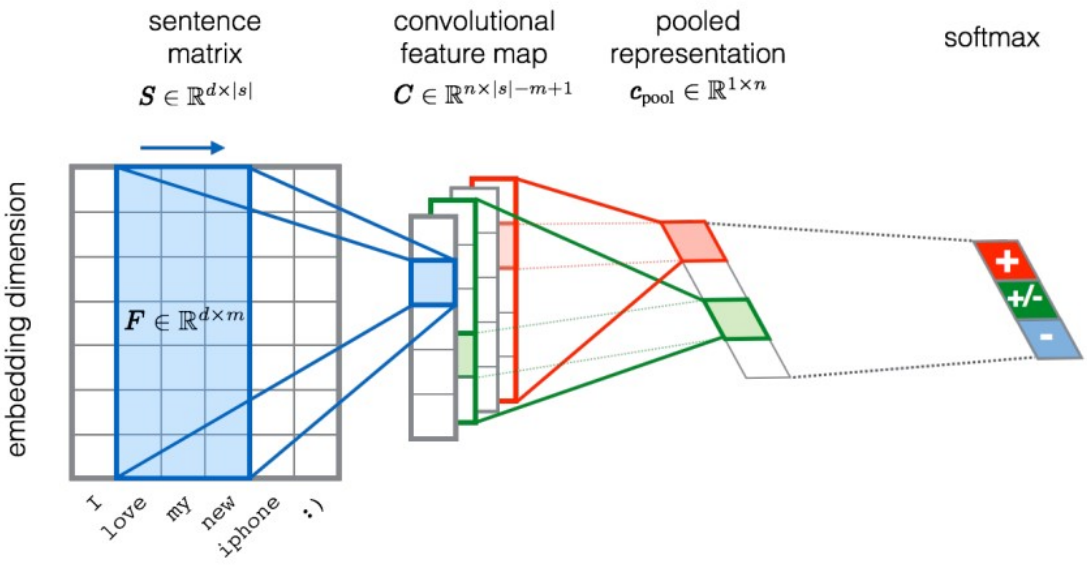


Fig 3.1 LAYERS OF CONVOLUTIONAL NEURAL NETWORK [54]

3.3.1 CONVOLUTION LAYER

In simple terms the convolution layer, will apply the cross correlation based procedure over all the data on the input tensor, and also transform the input data to match the number of filters based on parameters. It is the prime

base block of the convolutional neural network. The layer is composed of filters that cater as a tool to learn features over the input fed into the networks. During the feed forward phase, the filters convolve and apply mathematical extrapolation over the two dimensional input and produce a reduced version of the input. The filters are nothing but weights that are initially randomized and are then learnt progressively by the network during the training phase. The filter helps in permuting only the desirable parameters to pass further into the neural network.

Since the filters are applied in patches, the spatial correlation surrounding the context window is captured and hence can be attributed to the local connectivity around the region of the input dimension. One of the significant techniques in this layer is Parameter sharing. This is a scheme in the convolutional layer that assumes that if a particular patch filter is helpful in analysing and extracting relevant weights then that same patch can be applied across different cross section in order to help gain those features.

Apart from filters, Strides is a tuning parameter that is helpful in analysing the extent to which the input region is analysed. The window size is technically adapted by the strides and is helpful to look into the receptive fields between the columns of the two dimensional input. In order to not leave the end margins of the input, care must be taken to pad the input with zeros such that the stride properly covers all the regions.

3.3.2 POOLING LAYER

The Max pooling layer (Fig 3.2), is used to reduce the spatial dimensions, but not depth, on a convolution neural network, model. This is a method of non-linear down sampling. But by having less spatial information you gain computation performance. Less spatial information also means less parameter so less chance to over-fit and hence we retrieve some translation invariance.

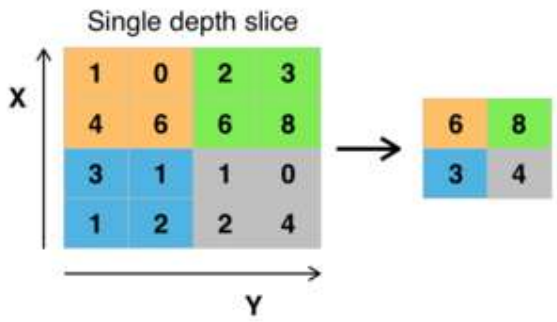


Fig 3.2 MAXPOOLING [55]

3.3.3 FULLY CONNECTED LAYER

Consider the completely associated layer as a black box with the accompanying properties: On the forward progressing system we have 3 inputs (Input flag, Weights, Bias) and 1 yield. On the back propagating the system has 1 input (dout) which has an indistinguishable size from yield and furthermore has 3 (dx,dw,db) yields, that has an indistinguishable size from the sources of info.

3.3.4 DROPOUT LAYER

This layer is an intermittent non-permanent layer that is activated at random intervals during the complete thirty one iterations of the network. This layer has the prime responsibility of eliminating over-fitting in the network. Over-fitting is the term used in the neural network domain to prevent complex co-adaptation on training and test data. It is an effective methodology for performing model averaging with deep neural network. We use the terminology dropout to indicate the dropping units in the neural network. What happens here is that when this layer is activated, the layer haphazardly removes some random nodes from the network and analyse the performance of the network. If the over-fitting declines then the network is made intact as such.

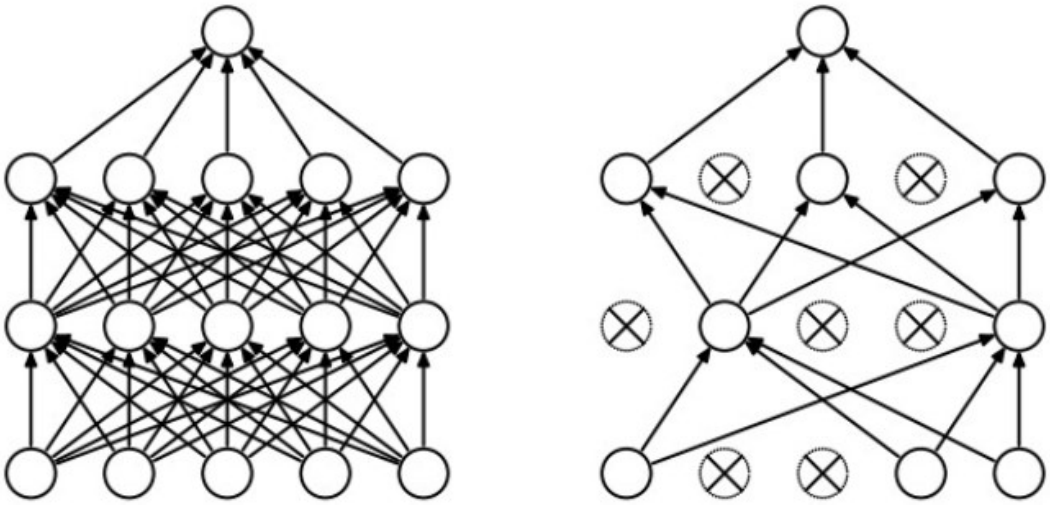


Fig 3.3 EFFECT OF APPLYING DROPOUT LAYER [10]

If the performance worsens then the action is rewound and some other slapdash nodes are eliminated and the process is repeated. A typical structural difference in the network has been depicted in the Fig 3.3 regarding the implementation of dropout layer. This in turn also decreases the processing time of the network.

Over-fitting is negated as a part of the neural system area to avoid complex co-adjustment on preparing and test information. It is a successful procedure for performing model averaging with profound neural system.

We utilize the phrasing dropout to demonstrate the dropping units in the neural system. What occurs here is that when this layer is enacted, the layer indiscriminately expels some arbitrary hubs from the system and investigate the execution of the system. In the event that the over-fitting decays then the system is made in place all things considered. On the off chance that the execution exacerbates then the activity is rewound and some other slapdash hubs are wiped out and the procedure is rehashed. There is an average auxiliary contrast in the system with respect to the usage of dropout layer.

3.4 RECURRENT NEURAL NETWORK

Recurrent Neural Network is a variant of the deep neural network which is suitable for processing sequence of data that is dependent on temporal nature. In this network, each hidden state is connected with the hidden state of the previous time step and also with the input and output layers. This helps in extensively capturing the time sequence based relation among the input sequence. Since there is a feedback loop in the hidden layer, the network is capable of performing retrospection over the high level feature extracted over the inputs at the previous time steps.

The cascade of network sequence (Fig 3.4) is helpful in empowering the network to remember long-term dependencies. This typical network model is inspired from the way humans read long sequences of texts such that every input sample is analysed and the features extracted at each step is used for future analysis.

$$\boldsymbol{h_t = \tanh\left(W * h_{t-1} + U * x_t + b\right)} \quad \textbf{... (3.1)}$$

$$\boldsymbol{y_t = softmax\left(V * h_t + c\right)} \quad \textbf{... (3.2)}$$

This also helps the network to correlate events at various time steps and is responsible for bringing out long term dependencies among the input series. This network uses Delta Learning law in order to comprehend the features and to correlate them. The hidden and output layers are represented as functions in Eqn 3.1 and 3.2 respectively.

It has been found that this temporally dynamic network is helpful for character level language modelling, weather forecasting, audio data processing and even stock price calculation. But this network suffers from the defect of assigning high bias to the later input sequences in comparison with that of former inputs.

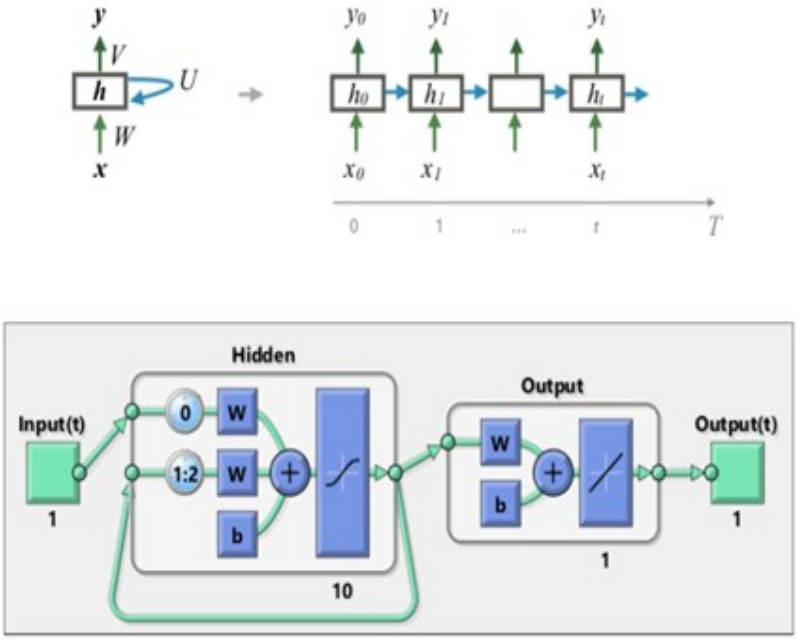


Fig 3.4 RECURRENT NEURAL NETWORK [56]

This bias leads to vanishing gradient problem when large sequences of inputs are fed into the network. There are several variants of this recurrent network that outperform the simpler version. They are as follows.

1. Long Short Term Memory
2. Gated Recurrent Unit
3. Bidirectional RNNs
4. Attention model based RNN.

3.4.1 LONG SHORT TERM MEMORY

One of the main drawbacks of the recurrent neural networks is the limited capability of the memory to remember long sequences of inputs. As the input sequence exponentially rises, the long period dependencies are not stored in the network. Hence in order to overcome this defect, the long short term memory (Fig 3.5) concept was introduced. To combat the drawbacks of the RNN, the long short term memory mainly plans to use a combination of properly

regulated cells that are capable of holding more features.

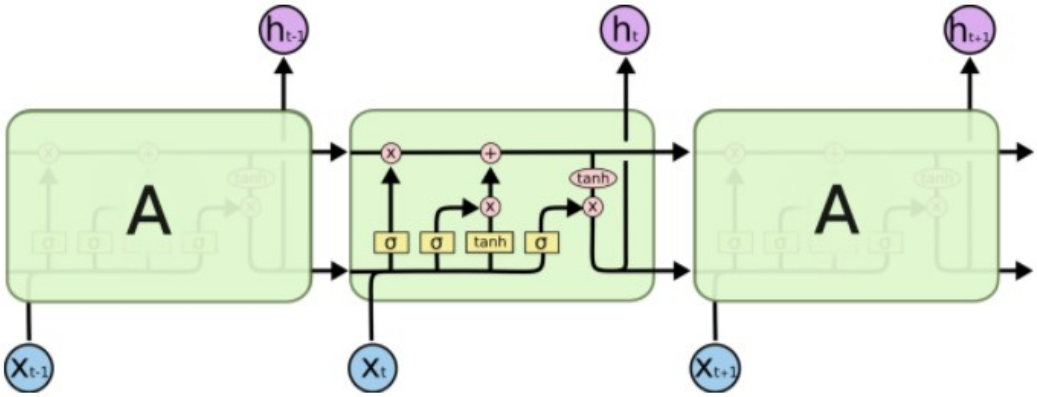


Fig 3.5 LONG SHORT TERM MEMORY [57]

The main purpose of these cells is to regulate the quantity of previously collected hidden state and the current input state which will be deposited in the current cell state. The various members of this cell combination include gating variables, candidate cell state and also the current hidden and cell state. One of the primary components of a long short term memory network is a Gate. A Gate is a selective filter that permits the flow of data. They composition of a gate includes a sigmoid based neural layer convoluted with point wise multiplication. The actual purpose of using sigmoid is to scale down the inputs on a scale between zero and one, which indirectly indicates the fraction of data to let through. For the purpose of interpretation, we can assert that the value of zero indicates the flow of no information while the other extreme of one indicates the flow of all information. As a result it acts as a kind of a regulated barrier to allow the passing of information.

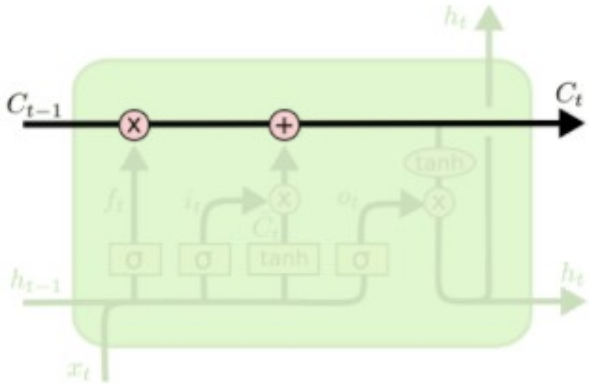


Fig 3.6 MEMORY CELL [57]

A memory cell (Fig 3.6) is the memory hub in a long short term memory network which is actually the key holder of values. This capability is empowered to the cells by the gates as they can regulate the flow of features in a network. In order to accomplish this purpose, the long short term memory network has different types of gates each with a specific purpose. One of them is the Forget Gate (Fig 3.7). The functioning of the gates are depicted in the form of equation in Eqn. 3.3.

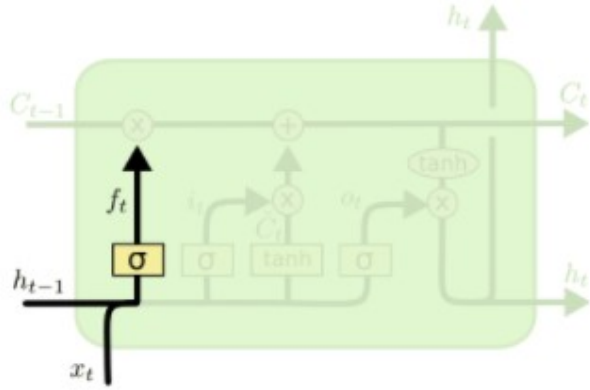


Fig 3.7 LSTM FORGET GATE [57]

$$\boldsymbol{f_t = \sigma(W_f * [h_{t-1}, x_t] + b_t)} \quad \textbf{... (3.3)}$$

This gate is responsible for deciding how much of the information from the previous cell state must be forgotten when we are processing in the current

state. This is indispensible because to remember the contents of current input a fraction of the previous memory must be erased to make space for this addition.

$$\boldsymbol{i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i)} \quad \textbf{... (3.4)}$$

This also uses a sigmoid based layer in order to assert the extent of content that must be forgotten.

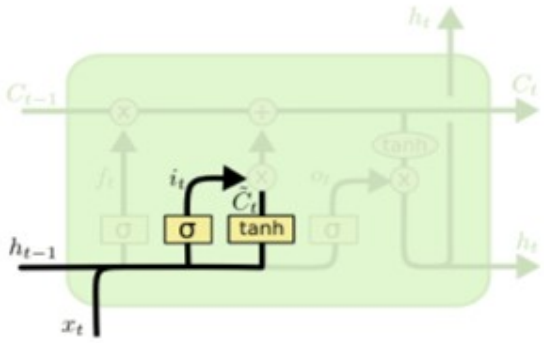


Fig 3.8 LSTM INPUT GATE [57]

The next step will be to ponder over the information that must be stored in the cell state. This is a two step procedure. First, the Input Gate (Fig 3.8) is another type of gate that plays the role of deciding which values to be updated. This is different from the Forget gate because a forget gate only regulates the fraction of things to be updated but the actual contents that must be updated are all proposed by the Input gate. This is also depicted in Eqn 3.4. Second, the Candidate cell is responsible for proposing the possible candidate to occupy the positions of the previous cell states that are currently subjected to replacement. The corresponding equation is given in Eqn 3.5.

$$\boldsymbol{o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o)} \quad \textbf{... (3.5)}$$

$$\check{c}_t = \boldsymbol{tanh(W_c * [h_{t-1}, x_t] + b_c)} \quad \textbf{... (3.6)}$$

Then the hyperbolic tangent based neural layer creates the candidate vector from the current input and hidden state outputs. The equation for this updation is shown in Eqn. 3.6.

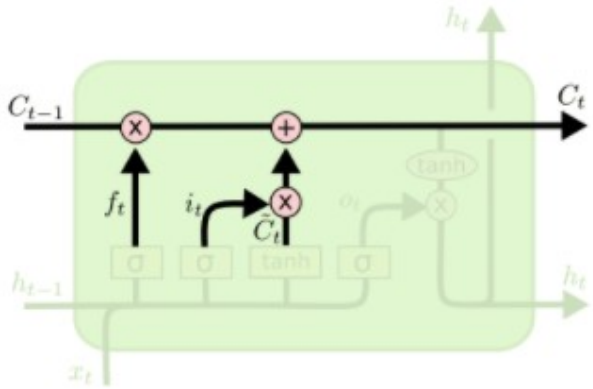


Fig 3.9 LSTM UPDATE CELL STATE [57]

The final step will be to amalgamate the forget gate, input gate and the candidate cells in order to arrive at the new candidate structure. This is done by multiplying the forget gate with the old cell state and the new candidate with the input gate as shown in Fig 3.9. These two outputs are then integrated to arrive at the new cell state of the current time step. The corresponding updation equations for these two cells is represented in Eqn. 3.7 and 3.8 respectively.

$$\boldsymbol{c_t = f_t \circ c_{t-1} + i_t \circ \check{c}_t} \quad \textbf{... (3.7)}$$

$$\boldsymbol{h_t = o_t \circ \tanh(c_t)} \quad \textbf{... (3.8)}$$

To output any value of the network at any time step we need to consider a fraction of the current cell state and the hidden layer content so that a proper representation of the current state of the network considering input sequences.

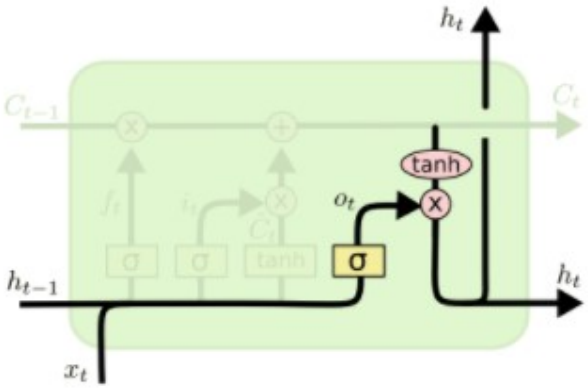


Fig 3.10 LSTM OUTPUT GATE [57]

Due to the presence of an inherent cell state other than the hidden state, the long short term memory is capable of storing more sequences of input in comparison with the traditional recurrent network. But as a consequence of many gates, more processing time is involved. The output of the layer is finally represented in Fig 3.10. There exists a variety of applications for long short term memory network including Text classification, Text summarization, Question answering system, Machine translation.

3.4.2 GATED RECURRENT UNIT

One of the main drawbacks of the recurrent neural networks is the limited capability of the memory to remember long sequences of inputs. As the input sequence exponentially rises, the long period dependencies are not stored in the network. Hence in order to overcome this defect, the long short term memory concept was introduced. But because of the presence of long sequences of processing, the time taken to run a long short term memory is very high. In order to remember large sequences of input along with providing quick performance, the gated recurrent units were introduced. One of the main differences between the long short term memory network and the gated recurrent units is that there exists no memory cell in the gated recurrent units. Obviously this leads to an increase in processing speed and even though there is

no memory cell, we have gates that regulate the flow of information among the previous and current states. Hence the accuracy of the model is maintained as similar to that of long short term memory.

$$z_t = \sigma(W_z * [h_{t-1}, x_t]) \quad \text{... (3.9)}$$

$$r_t = \sigma(W_r * [h_{t-1}, x_t]) \quad \text{... (3.10)}$$

Similar to long short term memory, the gated recurrent unit (Fig 3.11) is composed of gates. A Gate is a selective filter that permits the flow of data. They composition of a gate includes a sigmoid based neural layer convoluted with point wise multiplication. The actual purpose of using sigmoid is to scale down the inputs on a scale between zero and one, which indirectly indicates the fraction of data to let through. For the purpose of interpretation, we can assert that the value of zero indicates the flow of no information while the other extreme of one indicates the flow of all information. As a result it acts as a kind of a regulated barrier to allow the passing of information.

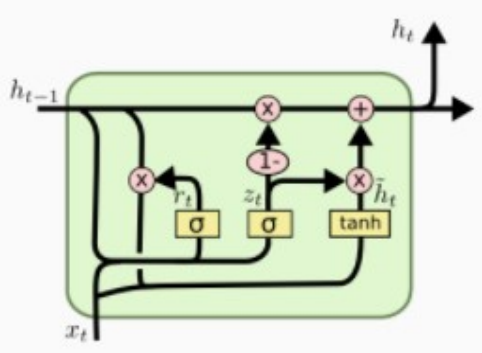


Fig 3.11 GATED RECURRENT UNIT [57]

The two types of gates used in gated recurrent units are the reset gate and the update gate. The reset gate is responsible for handling how much of the previous state must be considered for the current time step. This is essential to maintain the continuity of the time based sequential input. Hence this layer utilizes the sigmoid based neural layer for deciding the fraction of content to be

regulated. The equation is depicted for this gate in Eqn 3.9. The second type of gate is the update gate. The purpose of this gate is to collectively analyze both the previous fraction of hidden state and the input of the current state. The equation is depicted for this gate in Eqn 3.10. This acts as the judging factor that determines the performance of the gated recurrent unit. The output of the network is influenced by the hidden states and the output gate. The candidate for the hidden memory is calculated based on the Eqn 3.11 while the actual updation of the hidden layer is performed using Eqn 3.12.

$$\hat{h}_t = \boldsymbol{tanh}(\boldsymbol{W} * [\boldsymbol{r_t} * \boldsymbol{h_{t-1}}, \boldsymbol{x_t}]) \quad \textbf{... (3.11)}$$

$$\boldsymbol{h_t} = (\boldsymbol{1} - \boldsymbol{z_t}) * \boldsymbol{h_{t-1}} + \boldsymbol{z_t} * \hat{h}_t \quad \textbf{... (3.12)}$$

Since gated recurrent unit uses only two gates in comparison with the 3 gates and memory cell of the long short term memory, the gated recurrent unit consumes less memory space and the processing speed is higher. There exists a variety of applications for long short term memory network including Text classification, Text summarization, Question answering system, Machine translation.

3.4.3 BIDIRECTIONAL RNN

The Bidirectional Recurrent Neural Network (Fig 3.12) is another variant of the recurrent neural network. The idiosyncrasy of this network is its capability to retrospect both the past and the future sequences in order to contextually analyse the current input. Traditional RNNs only consider the previous input sequence while that of the Bi-directional RNNs both the former and the latter sequences.

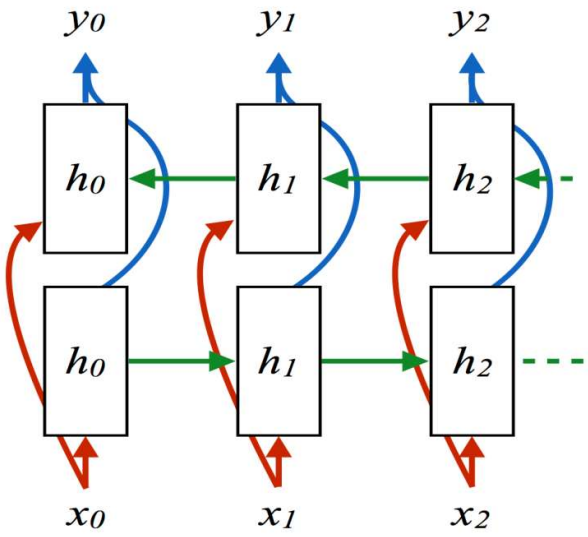


Fig 3.12 BIDIRECTIONAL RNN [58]

The concept of Bidirectional Recurrent Neural Network is to divide the neurons of a normal Recurrent Neural Networks into two directions, one for positive time direction (forward states), and another for negative time direction (backward states). But those two states output in that network are not in contact to inputs of the opposite direction states. Applications of bidirectional recurrent neural network include machine translation, handwritten recognition, speech recognition and text processing.

3.4.4 ATTENTION MODEL

Attention model is a relatively new technique that has lot of attention in the domain of the recurrent neural networks. The purpose of this technique is to ensure that the network prioritizes only in certain parts of the network giving less preference to irrelevant content. This is designed based on the inspiration from the way humans perceive information. Focussing only on certain parts of data is a part of prioritization is useful in understanding the context of the input. This was designed to improve the performance of encoder-decoder sequences.

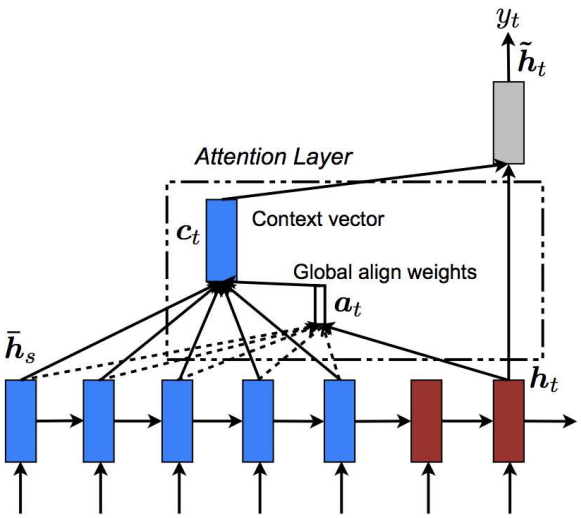


Fig 3.13 ATTENTION MODEL [59]

The attention mechanism (Fig 3.13) of the recurrent network consists of context vector, align weights and attention vector. In order to grasp the contextual information from the data, the input is initially sent to a recurrent neural network.

$$\boldsymbol{a_t(s) = align(h_t, \hbar_s)} \quad \textbf{... (3.13)}$$

$$\boldsymbol{a_t(s) = \frac{e^{score(h_t, h_s)}}{\sum e^{score(h_t, h_s)}}} \quad \textbf{... (3.14)}$$

$$\boldsymbol{h_t = \tanh(W_c * [c_t, h_t])} \quad \textbf{... (3.15)}$$

$$\boldsymbol{p(y_t | y_{<t}) = softmax(W_s * h_t)} \quad \textbf{... (3.16)}$$

Followed by this, the processed input is sent to the context vector within the attention layer. This layer tries to extrapolate the underlying features from the processed data. The corresponding equation for alignment calculation, attention calculation, hidden layer updation and probabilistic attentionestimation are depicted in Eqn 3.13, 3.14, 3.15 and 3.16 respectively.

$$score(h_t, h_s) = \begin{cases} h_t^T h_s \ \ dot \\ h_t^T W_a h_s \ \ general \\ v_a^T \tanh(W_a[h_t; h_s]) \ \ concat \end{cases} \quad \text{... (3.17)}$$

$$a_t = softmax(W_a * h_t) \quad \text{... (3.18)}$$

An attention model in a recurrent neural network lets the network specifically shift its focus on a particular subset of the input, analyse it, and then change its focus to some other part of the input. The final outcome score and the attention vectors are calculated using Eqn 3.17 and 3.18 respectively. The align weights are later used after the context vector that actually determines which process of the input to focus on. Hence this is an efficient mechanism to collect the relevant features. Applications of attention model include visual recognition, handwritten recognition, speech recognition and text summarization and text classification.

3.5 SYSTEM ARCHITECTURE

3.5.1 RECURRENT AND CONVOLUTIONAL PARALLEL SEQUENTIAL NEURAL NETWORK

3.5.1.1 ARCHITECTURE

In the first variant of the proposed parallel architectures, the recurrent and the convolutional neural network components are placed separately. This connotes that the processed input data is fed independently to both the serial convolutional and the serial recurrent network parts. Hence this model is termed as Recurrent And Convolutional Parallel Sequential Neural Network (RCPSNN) (Fig 3.14)

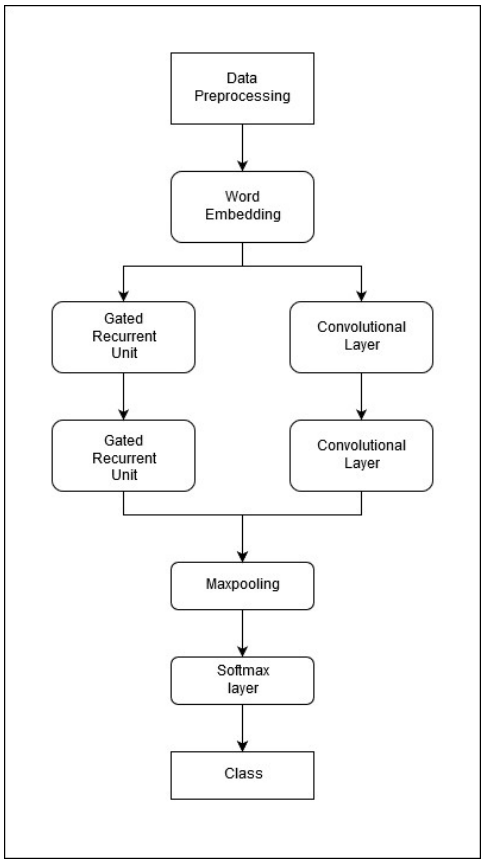


Fig 3.14 RCPSNN ARCHITECTURE

The outputs of these two serial networks are then amalgamated by applying maxpool function. Thus only the prominent features of both the networks are taken into consideration thus eliminating the redundant insignificant features.

3.5.1.2 ALGORITHM

In the first variant of the parallel architectures, the recurrent and the convolutional neural network components are placed separately. This connotes that the processed input data is fed independently to both the serial convolutional and the serial recurrent network parts. The outputs of these two serial networks are then amalgamated by applying maxpool function. Thus only the prominent features of both the networks are taken into consideration thus eliminating the redundant and insignificant features. The abstract algorithmic

representation ofrecurrent and convolutional parallel sequential neural network architecture is depicted as follows in Fig 3.15.

Input: Dataset that is not pre-processed

Output: A list of class vectors for each input sample

Step 1: Receive the dataset and split it into input and output lists

Step 2: Pre-process the input list by performing tokenization, punctuation removal, stop words removal, normalization, unknown words removal and padding the length of the input sequence.

Step 3: Perform Word Embedding over the input sequences using Google Word vectors.

Step 4: Pre-process the output list by replacing class labels with their indices. Then perform one-hot encoding over the class indices to get the target output.

Step 5: Split the pre-processed input-output pair into training, validation and testing data pairs.

Step 6: Now start the training phase by initialising the number of epochs for which the algorithm must be trained.

Step 7: For each epoch e, do the following.

Step 8: For each training input sample i do the following.

Step 9: Feed the sample i into the first convolutional and first recurrent neural network independently.

Step 10: Take the output of the first convolutional network and feed it as input to the second convolutional network. Similarly, take the output of the first recurrent network and pass it as input to the second recurrent network.

Step 11: Take the outputs of the second convolutional and second recurrent neural network and apply maximum merge function over them.

Step 12: Take the output of this merge function and apply softmax function to get the actual output class vector for that sample.

Step 13: Now perform back propagation by comparing the difference between

the expected and actual class vector and pass this as loss onto the second convolutional and second recurrent network.

Step 14: Calculate the error in the second convolutional and second recurrent networks respectively and adjust the weights and bias values in accordance with this error. Pass on these error values of second convolutional and second recurrent network to first convolutional and first recuwrrent netorks.

Step 15: Calculate the error in the first convolutional and first recurrent networks respectively and adjust the weights and bias values in accordance with this error.

Step 16: Choose the next input-output pair and go back to Step 9.

Step 17: Validate each of the validation dataset pairs by repeating the steps from 9 to 12.

Step 18: Plot the accuracy and loss graphs for both the training and validation pairs and analyse them.

Step 19: Repeat the step 8.

Step 20: Perform the testing phase over the trained model by performing steps 9 to 12 for each dataset pair in the testing dataset.

Step 21: Compute the evaluation metrics such as F1, accuracy, Precision, Recall over the output of the testing dataset and analyse them.

Fig 3.15 RCPSNN ALGORITHM

3.5.2 RECURRENT AND CONVOLUTIONAL PARALLEL CROSS NEURAL NETWORK

3.5.2.1 ARCHITECTURE

In the second variant of the parallel architectures, the recurrent and the convolutional neural network components are combined sequentially in two ways. Similar to previous architecture, the processed input data is fed independently to both the branches. In one branch, the convolutional network's output is fed immediately to that of the recurrent part. On the other branch, the

recurrent part's output is fed into the convolutional part. Hence this model is termed as Recurrent And Convolutional Parallel Cross Neural Network (RCPCNN) (Fig 3.16)

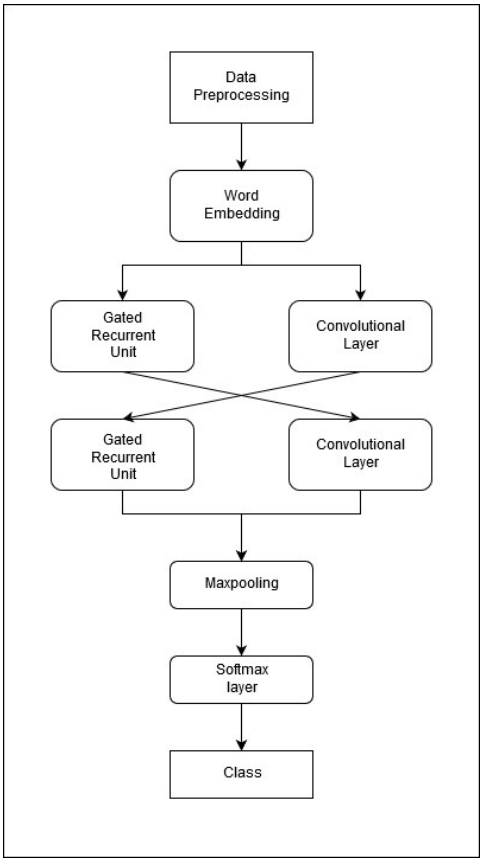


Fig 3.16 RCPCNN ARCHITECTURE

The outputs of these two serial networks are then mixed by applying maxpool function. Thus only the prominent features of both the networks are taken into consideration thus eliminating the redundant and insignificant features.

3.5.2.2 ALGORITHM

In the second variant of the parallel architectures, the recurrent and the convolutional neural network components are combined sequentially in two ways. Similar to previous architecture, the processed input data is fed independently to both the branches. In one branch, the convolutional network's

output is fed immediately to that of the recurrent part. On the other branch, the recurrent part's output is fed into the convolutional part. The outputs of these two serial networks are then mixed by applying maxpool function. The algorithm is as follows in Fig. 3.17

Input: Dataset that is not pre-processed

Output: A list of class vectors for each input sample

Step 1: Receive the dataset and split it into input and output lists

Step 2: Pre-process the input list by performing tokenization, punctuation removal, stop words removal, normalization, unknown words removal and padding the length of the input sequence.

Step 3: Perform Word Embedding over the input sequences using Google Word vectors.

Step 4: Pre-process the output list by replacing class labels with their corresponding indices. Then perform one-hot encoding over the class indices to get the target output.

Step 5: Split the pre-processed input-output pair into training, validation and testing data pairs.

Step 6: Now start the training phase by initialising the number of epochs for which the algorithm must be trained.

Step 7: For each epoch e, do the following.

Step 8: For each training input sample i do the following.

Step 9: Feed the sample i into the first convolutional and first recurrent neural network independently.

Step 10: Take the output of the first convolutional network and feed it as input to the second recurrent network. Similarly, take the output of the first recurrent network and pass it as input to the second convolutional network.

Step 11: Take the outputs of the second convolutional and second recurrent neural network and apply maximum merge function over them.

Step 12: Take the output of this merge function and apply softmax function to get the actual output class vector for that sample.

Step 13: Now perform back propagation by comparing the difference between the expected and actual class vector and pass this as loss onto the second convolutional and second recurrent network.

Step 14: Calculate the error in the second convolutional and second recurrent networks respectively and adjust the weights and bias values in accordance with this error. Pass on these error values of second convolutional and second recurrent network to first recurrent and first convolutional networks respectively.

Step 15: Calculate the error in the first convolutional and first recurrent networks respectively and adjust the weights and bias values in accordance with this error.

Step 16: Choose the next input-output pair and go back to Step 9.

Step 17: Validate each of the validation dataset pairs by repeating the steps from 9 to 12.

Step 18: Plot the accuracy and loss graphs for both the training and validation pairs and analyse them.

Step 19: Repeat the step 8.

Step 20: Perform the testing phase over the trained model by performing steps 9 to 12 for each dataset pair in the testing dataset.

Step 21: Compute the evaluation metrics such as F1, accuracy, Precision, Recall over the output of the testing dataset and analyse them.

Fig 3.17 RCPCNN ALGORITHM

3.5.3 RECURRENT AND CONVOLUTIONAL PARALLEL SEQUENTIAL CROSS NEURAL NETWORK

3.5.3.1 ARCHITECTURE

In the third variant of the parallel architectures, the objective is to integrate the features of the previous two architectures. Similar to the previous architectures, the processed input data is fed independently to both the independent branches.

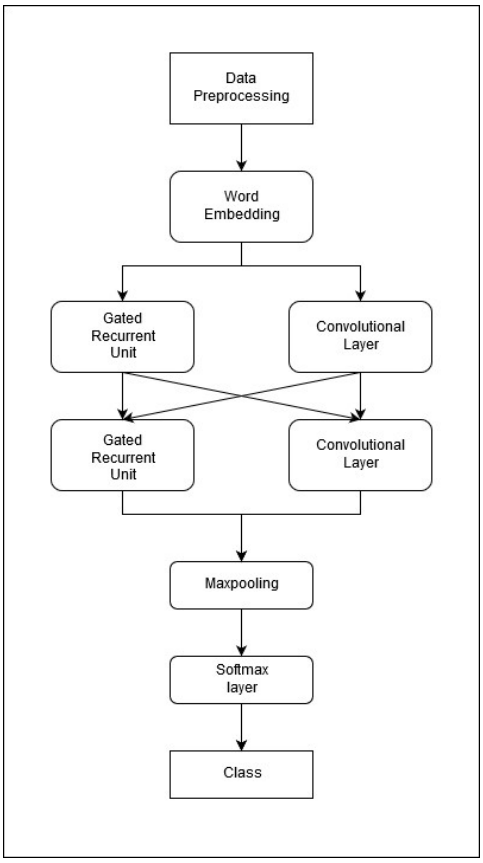


Fig 3.18 RCPSCNN ARCHITECTURE

In the first step, the output of the convolutional part and the recurrent part is interspersed to get an intermediate output. This output is fed into a separate convolutional and recurrent network in the next time step. Hence this model is termed as Recurrent And Convolutional Parallel Sequential Cross Neural Network (RCPSCNN) (Fig 3.18). The outputs of these two serial networks are

then amalgamated by applying maxpool function. Henceforth only the significant features of both the networks are taken into consideration thus eliminating the useless and irrelevant features.

3.5.3.2 ALGORITHM

In the third variant of the parallel architectures, the objective is to integrate the features of the previous two architectures. Similar to the previous architectures, the processed input data is fed independently to both the independent branches. In the first step, the output of the convolutional part and the recurrent part is interspersed to get an intermediate output. This output is again fed into a separate convolutional and recurrent network in the next time step. The outputs of these two serial networks are then amalgamated by applying maxpool function. Henceforth only the significant features of both the networks are taken into consideration thus eliminating the useless and irrelevant features. The algorithm is as follows in Fig 3.19.

Input: Dataset that is not pre-processed

Output: A list of class vectors for each input sample

Step 1: Receive the dataset and split it into input and output lists

Step 2: Pre-process the input list by performing tokenization, punctuation removal, stop words removal, normalization, unknown words removal and padding the length of the input sequence.

Step 3: Perform Word Embedding over the input sequences using Google Word vectors.

Step 4: Pre-process the output list by replacing class labels with their corresponding indices. Then perform one-hot encoding over the class indices to get the target output.

Step 5: Split the pre-processed input-output pair into training, validation and testing data pairs.

Step 6: Now start the training phase by initialising the number of epochs for which the algorithm must be trained.

Step 7: For each epoch e, do the following.

Step 8: For each training input sample i do the following.

Step 9: Feed the sample i into the first convolutional and first recurrent neural network independently.

Step 10: Take the output of the first convolutional and first recurrent networks and apply maximum merge function over them. Take this merged output and feed it as input to the second convolutional network and second recurrent network.

Step 11: Take the outputs of the second convolutional and second recurrent neural network and apply maximum merge function over them.

Step 12: Take the output of this merge function and apply softmax function to get the actual output class vector for that sample.

Step 13: Now perform back propagation by comparing the difference between the expected and actual class vector and pass this as loss onto the second convolutional and second recurrent network.

Step 14: Calculate the error in the second convolutional and second recurrent networks respectively and adjust the weights and bias values in accordance with this error. Pass on both these error values of second convolutional and second recurrent network to thefirst convolutional and first recurrent networks.

Step 15: Calculate the error in the first convolutional and first recurrent networks respectively and adjust the weights and bias values with this error.

Step 16: Choose the next input-output pair and go back to Step 9.

Step 17: Validate each of the validation dataset pairs by repeating the steps from 9 to 12.

Step 18: Plot the accuracy and loss graphs for both the training and validation pairs and analyse them.

Step 19: Repeat the step 8.

Step 20: Perform the testing phase over the trained model by performing steps 9 to 12 for each dataset pair in the testing dataset.

Step 21: Compute the evaluation metrics such as F1, accuracy, Precision, Recall over the output of the testing dataset and analyse them.

Fig 3.19 RCPSCNN ALGORITHM

3.6 WORK FLOW

The workflow of the text classification system (Fig 3.20) is split into various modules and the output of each module is collectively integrated. In the pre-processing phase (1), the document collection is fed into the model as input and output. The input collection is a series of text documents that requires extensive pre-processing. In each sample of the collection, the document is partitioned into a sequence of tokens. Each token is converted into lowercase representation and all the punctuations are removed from the document.

The next step is to remove all the frequently used stop words in the English language that don't contribute any significant value for the purpose of classification. Finally the sequence is trimmed to a maximum length that is required. After padding the sequence of the required length, the word embedding (2) is performed, such that a pre-trained word vector such as Google word vectors or GloVe embeddings are used. The purpose of these embeddings is to convert each word into an n-dimensional vector array. This is helpful in processing the text input as a series of numerical vectors. At the end of this phase, we receive a k*n dimensional real valued matrix in which each of the k-words are represented as the n-dimensional array.

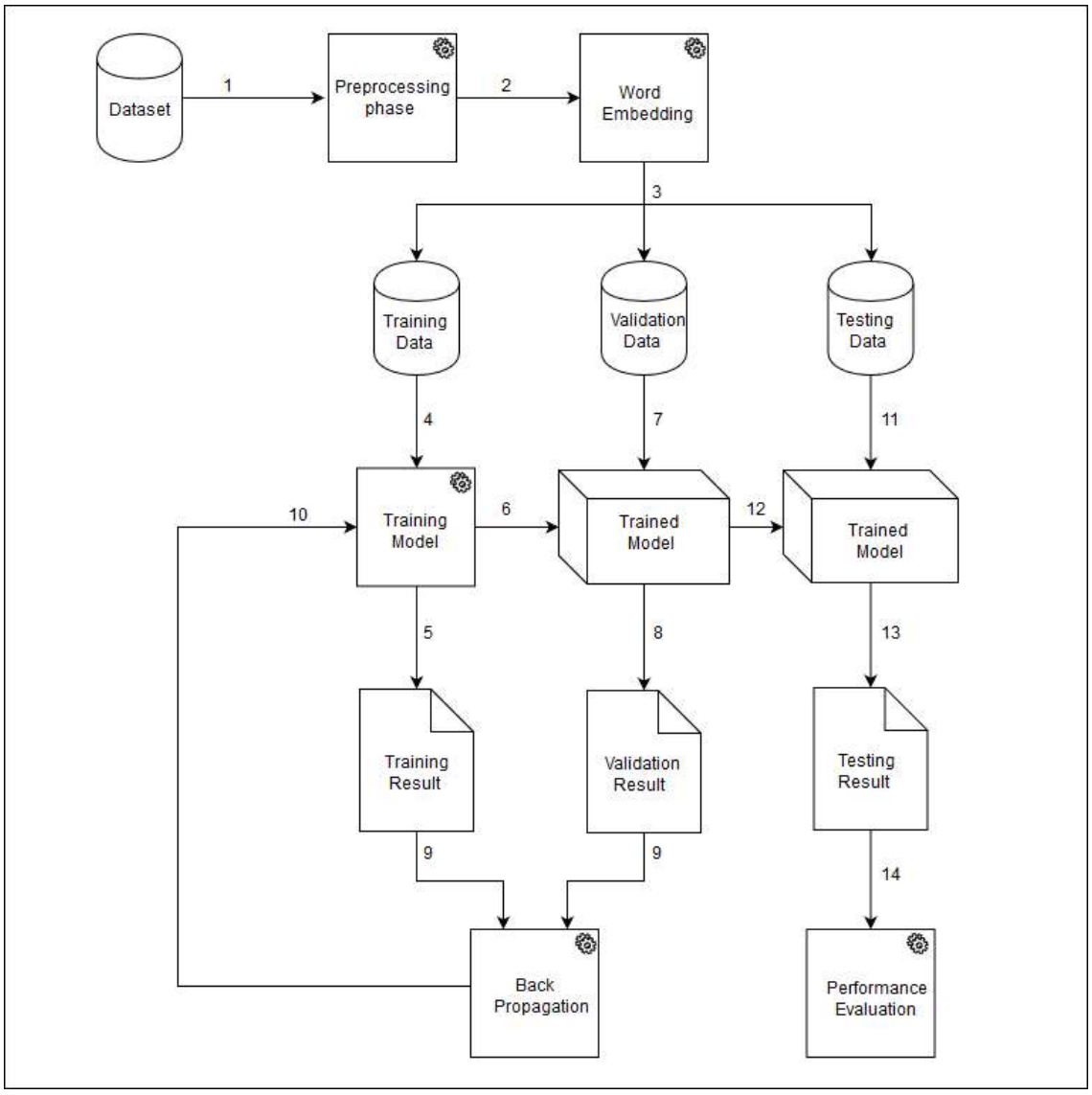


Fig 3.20 WORKFLOW OF THE TEXT CLASSIFICATION SYSTEM

In the one-hot encoding phase, the output file is converted into a representation that is helpful for the model to process, analyze and evaluate. The unprocessed output file consists of integer class values ranging from 0 to m class values. The process of one-hot encoding involves taking each class value and converting it into an m-dimensional vector which contains binary values such as 0 or 1. Thus for each document there is an m-size vector where only the ith position in the vector is set to 1 while rest is set to 0 enunciating that the document belongs to the ith class. This representation is commonly preferred

for neural network based models that involve classification. The dataset is then split into training data, validation data and testing data (3).

In the feed-forward phase, initially the pre-processing and one-hot encoding stages are completed. After that, these tuned input and output are fed into the network model. The input matrix is ingested either into the convolutional or recurrent neural sub network. When an input is given into a convolutional neural network, the matrix of the input is taken and filters of varying sizes are applied to it. As a consequence, we obtain a contiguous sequence of processed layers are obtained. Each of the layers is a spatial representation of the features that are normalized by the activation function. Within the convolutional network, we apply a maxpooling layer that integrates the stacks of layers into a higher level representation. Thus the convolution based network helps to capture the spatial relationship among the input features.

When we shift out focus to the specific case of recurrent neural network (RNN) such as Long Short Term Memory (LSTM) or the Gated recurrent unit (GRU) then it is perceivable that the main idiosyncrasy of these nets are the temporal relation that they capture. Given an input sequence, the RNNs take one segment of input at a time and processes one input at a time step. In each time step, the hidden state of the network possesses a memory cell that recollects the previous sequence of input states and is able to bind those representations with the current input state. Thus the time-series relation of the input is preserved by the recurrent network.

Also in the back-propagation phase (9), the learning section is similar in both the convolutional and recurrent network phases. In any neural network, the actual output produced by the network is compared with the expected output to generate the error feature. This error feature at the k-th layer is passed on to the (k-1)th layer and the corresponding weights and biases of that layer are updated based on this error. Similarly the error feature of the (k-1)th layer is propagated to the (k-2)th layer and so on till we reach the input layer. In this manner, we

train the weights and biases of the network and the learning is improved (10). This indirectly reduces the error rate and the gradient values are minimised.

In the training procedure (4), we first pre-process the input and one-hot encode the output. These set of parameters are fed into the feed-forward phase to generate the actual output. After augmenting it with the expected output, the back-propagation phase is performed for each collection. The combination of feed-forward and back-propagation phase over all the input-output samples is termed as a single epoch. Training phase mainly involves performing several epochs till the model is learnt.

In the validation procedure (7), the fraction of unseen sample is taken and a feed-forward phase is executed. The output of this phase is evaluated and compared in juxtaposition with the training phase in order to ensure the model is learning the features properly at the end of each epoch.

In the testing phase (11), the feed-forward pass is done after the pre-processing and one-hot encoding passes. After the feed-forward pass, the actual output is taken and compared with the expected output by using several evaluation metrics such as precision, accuracy, F-1 score, error and recall (14).

3.7 MODULE DESCRIPTION

There are four main modules in the system. They are,

1. Data Pre-processing
2. Word Embedding
3. Convolutional Neural Network Module
4. Gated Recurrent Unit Module

3.7.1 DATA PRE-PROCESSING

The pre-processing phase is executed before the document collection is fed into the model as input and output. The dataset collection is a series of text documents that requires extensive pre-processing. In each sample of the

collection, initially a process named as tokenization is performed. The process of Tokenization mainly involves chunking a long sequence of any data into smaller sub sequences. In the case of Text processing, the collection of document is taken one by one each of which has large amount of sentences. Each sentence of each document is taken and it is split further into an array of words. Thus each of the documents is further sub-divided into sentences and words. This process of tokenization in natural language processing is also termed as Lexical Analysis.

After the process of tokenization, we perform another task known as Normalization. The term Normalization in natural language processing usually means that the texts represented are on a same scale. This implicates that either all the text tokens are in lower case or else in upper case. This is useful because the normalization helps the processor to provide equal focus and perform uniform processing over all the words without any predisposition.Hence as a result, each token is converted into lowercase representation.Further, the process termed as Stemming is performed in order to eliminate all the futile affixes such as all the prefixes, infixes, suffixes. This is done in order to unify contextually similar words having varied grammar tenses.

Finally, all the punctuations are removed from the document. Along with that the stop words of the language are considered. Stop words are most frequently used common words that do not provide any significant contribution to the actual representation of the content. They are rather placed in order to grammatically assert the correct content. The next step is to remove all the frequently used stop words in the English language that don’t contribute any significant value for the purpose of classification. Finally the sequence is trimmed to a maximum length that is required. This final step ensures that all the input sequences fed are of equal length and hence uniformity is assured.

3.7.2 WORD EMBEDDING

Word Embedding is a feature learning and language modelling technique that is typically used in natural language processing with the assistance from deep learning tools in order to mathematically represent lexical content. This process mainly refers to defining a vector based approach for contextually exploiting the similarity among various words. Theoretically the objective of word embedding is to represent each word as a contiguous vector of a particular size in a mathematical space. This technique is mainly helpful because the contextually simulated vector representation of lexemes is effective to process and identify the inherent patterns in them.

The technique of word embedding can be performed using deep neural networks. This model is known as Word2Vec. Word2Vec which typically means word to vector is a machine learning model that typically has three layers of neural network.

In order to preserve the contextual integrity among the words, the model mainly aims to predict the surrounding words in a phrase from the centre word in the phrase. This model is also known as Skip-gram model (Fig 3.21). The main purpose and objective behind the skip-gram based model is to take every token in a collection and to evaluate its relation with the surrounding n-words. Thus in some sense, we try to define a window around each word and evaluate the contextual significance of that word in that window frame. In the neural network, each word is one-hot encoded and then given as input. The hidden layers are randomly initialized in the beginning of the training phase. The expected output will be the set of one-hot encoded k-words surrounding that word. Hence the model is made to learn about the contextual similarity around these words and hence can easily be used to represent the mathematical vector for each word.

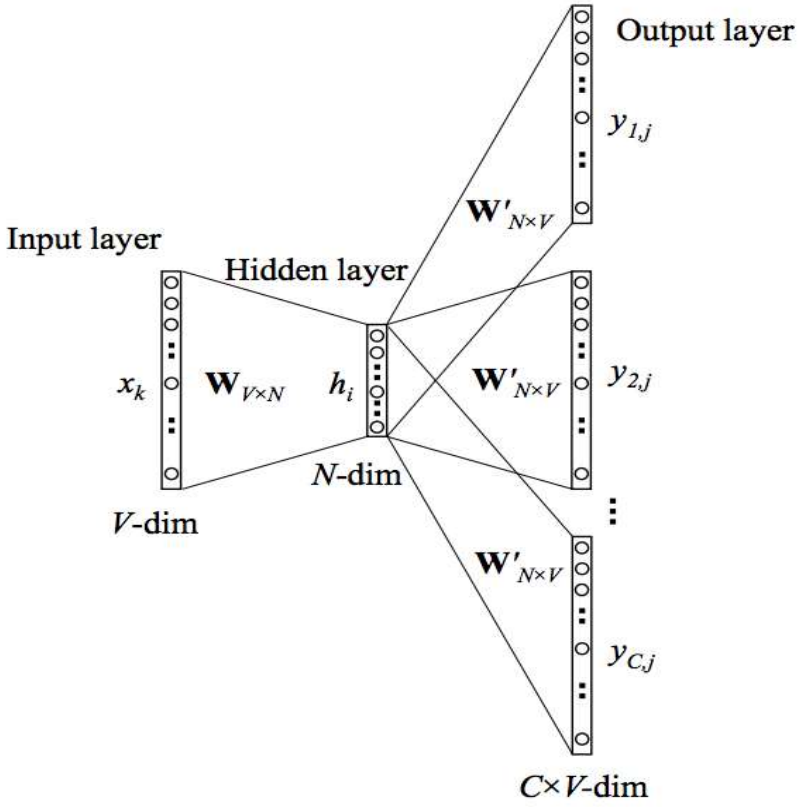


Fig 3.21 SKIP-GRAM MODEL [21]

Google's pre-trained word vector is one of the Word2Vec models. This is nothing but a dictionary of (key, value) pair in which for each word, there exist a k-dimensional numeric vector. This pre-trained model has a vocabulary of around three million lexemes and each token is represented by 300 numerical features. This model is trained over Google news dataset that has about 100 billion words and phrases. Hence it is a standard model to interpret the contextual significance of each word as a mathematical representation.

3.7.3 CONVOLUTIONAL NEURAL NETWORK MODULE

The convolutional neural network is a type of deep neural network that helps in capturing feature by using spatial analytics. Within a CNN there are several kinds of layers each with a distinct function. When the CNN is perceived in a mathematical dimension, there exists a cross-correlation among the inputs and

the output generated by the network. One of the key specialities of using convolutional neural network is its ability to capture features for every n-gram of words.

Table 3.1 - CONVOLUTION NEURAL NETWORK PARAMETERS

COMPONENT	QUANTITY
CONVOLUTION - 1	
Output units	64
Output dimension	(746,64)
Weight dimension	(4,300,64)
Number of Filters	64
Filter Size	4300
Strides	1
Bias dimension	(64)
MAXPOOL - 1	
Pool size	(4)
CONVOLUTION - 2	
Output units	64
Output dimension	(186,64)
Weight dimension	(4,300,64)
Number of Filters	64
Filter Size	4300
Strides	1
Bias dimension	(64)
MAXPOOL - 2	
Pool size	(4)

This window based approach is useful in capitulating the interdependent relation among consecutive words in a sequence. In our proposed architecture we use 2 convolutional neural network with the following specifications as depicted in Table 3.1.

3.7.4 GATED RECURRENT UNIT MODULE

The gated recurrent unit is a special type of recurrent neural network that captures feature by using temporal analytics. Within a GRU there are several kinds of gates each with a distinct function. When the GRU is perceived in a mathematical dimension, there exists a time series based relation among the inputs and the output generated by the network.

Table 3.2 – GATED RECURRENT UNIT PARAMETERS

COMPONENT	QUANTITY
GRU - 1	
Output units	64
Input Weight dimension	(300,192)
Recurrent Weight	(64,192)
Bias dimension	(1,192)
GRU - 2	
Output units	64
Input Weight dimension	(64,192)
Recurrent Weight	(64,192)
Bias dimension	(1,192)

One of the key specialities of using recurrent neural network is its ability to comprehend long sequences of input rather than a specific window based approach. This approach catalyzes in capitulating the long term dependencies among consecutive words in a sequence. In our proposed architecture we use 2 gated recurrent units with the following specifications as depicted in Table 3.2.

3.8 CHAPTER SUMMARY

In this chapter, the overall architecture of the proposed deep neural network based text classifier system and its components are explained. All the modules of the system are enunciated and the features of the convolutional neural network and recurrent neural network are explained elaborately. The important activation functions such as sigmoid, hyperbolic tangent, rectified linear unit and their variants are discussed. In next chapter, the implementation details and results obtained will be discussed.

CHAPTER 4
ACTIVATION FUNCTION ANALYSIS

4.1 INTRODUCTION

An activation function is a mathematical expression that is used for normalizing the inputs on a uniform scale such that the network is equally influenced by all the input parameters. This transformation function can be either linear or non-linear. The activation function is the essential component of any neural network and it acts as a universal approximation function. It determines the extent to which certain part of the network is involved in contributing to the final output.

4.2 NEED FOR ACTIVATION FUNCTION

A neural network layer without any activation functions is typically a weighted linear combination of its input. Hence only a linear output can be expected from such a network when no activation function is used. Thus in such a case we can expect the network to solve problems only having linear relation among inputs and outputs. But in reality most of the problems that we encounter have non-linear complex correlation among the inputs and outputs. Hence in order to address this issue, we must implement complex activation functions in each layer of the network. Thus activation functions empower the network to map the complex relation among the inputs and outputs to solve non-linear problem. Apart from acting as non-linear approximators, activation functions are also used as ideal decision making functions within each layer so that each layer can filter out only the relevant features and pass them to the successive layers.

4.3 TYPES OF ACTIVATION FUNCTIONS

There are several variants of the activation function that have been used in neural networks.

1. Sigmoid function
2. Hyperbolic Tangent function
3. Rectified linear unit
4. Leaky rectified linear unit
5. Exponential linear unit
6. Proposed Inverse exponential linear unit
7. Softmax function

4.3.1 SIGMOID FUNCTION

A Sigmoid function is a non linear activation function and the graphical representation of the function resembles the shape of the letter 'S'.

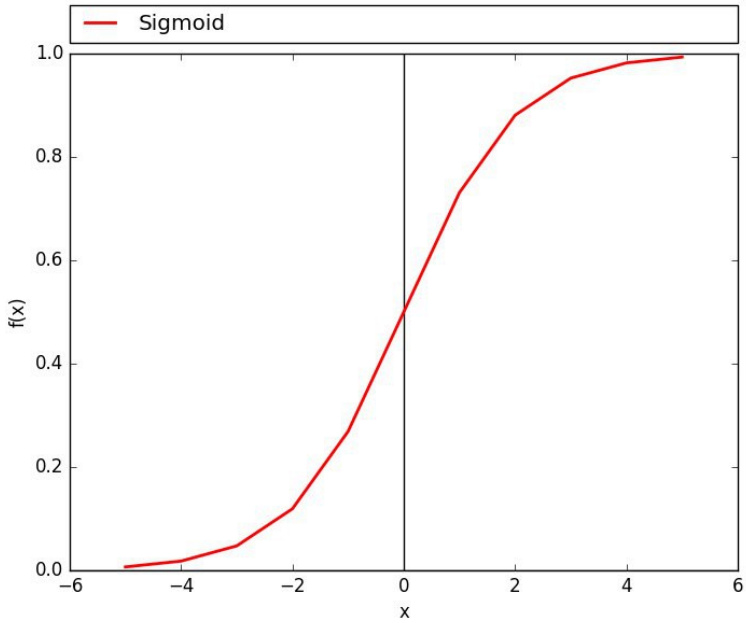


Fig 4.1CHARACTERISTICS OF SIGMOID FUNCTION

It is considered as a unique case of logistic function which is a bounded function that has a limited range between 0 and 1. The characteristic nature of

the sigmoid function in portrayed in Fig 4.1. The sigmoid function is represented in Eqn 4.1 and its first order derivative is given in Eqn 4.2 respectively.

$$f(x) = \frac{1}{1 + e^{-x}} \quad \text{... (4.1)}$$

$$f'(x) = f(x) * \big(1 - f(x)\big) \quad \text{... (4.2)}$$

This is a real valued function which is monotonic in nature such that the output remains within this range for any value of input (positive or negative). The derivative of this function is non-negative and this differential function has a bell shaped graphical representation. This model suffers from the drawback of vanishing gradient in which as the input value increases the output value remains saturated. Hence no learning takes place beyond a certain range.

4.3.2 HYPERBOLIC TANGENT FUNCTION

In spite of having an elegant and simple representation, the sigmoid function is incapable of accounting for the negative part of the input as it always outputs a probabilistic positive value. Hence another activation function termed as hyperbolic tangent function was introduced. Resembling the graphical structure of sigmoid, the hyperbolic tangent function also has a “S” shape structure but the range of the output that this function gives varies from -1 to 1.

Hence in comparison with sigmoid function it is broader and can accommodate more features for representation. This is a real valued function which is monotonic in nature such that the output remains within this range for any value of input (positive or negative). The derivative of this function is non-negative and this differential function has a bell shaped graphical representation.

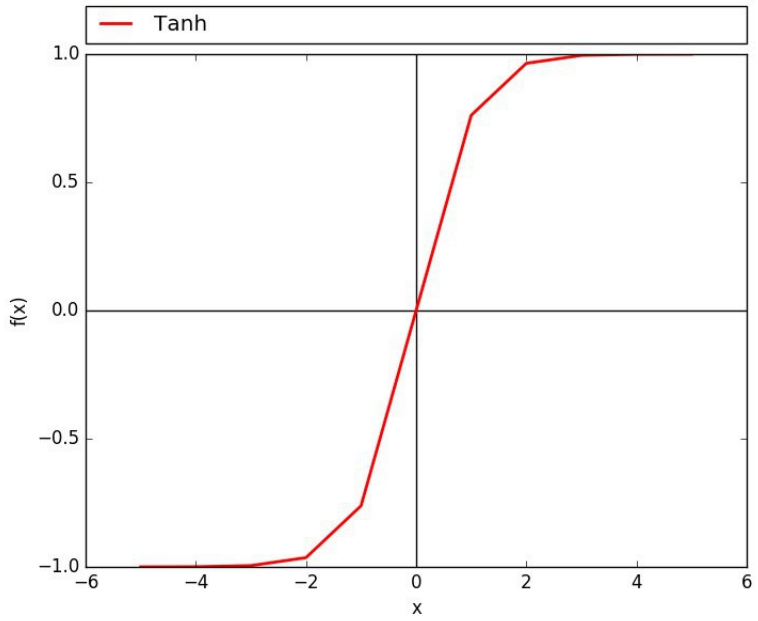


Fig 4.2CHARACTERISTICS OF TANH FUNCTION

The characteristic nature of the tanh function in portrayed in Fig 4.2. The tanh function is represented in Eqn 4.3 and its first order derivative is given in Eqn 4.4 respectively.

$$f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \quad \text{... (4.3)}$$

$$f'(x) = 1 - f(x)^{2} \quad \text{... (4.4)}$$

This model suffers from the drawback of vanishing gradient in which as the input value increases the output value remains saturated. Hence no learning takes place beyond a certain range. But this range is quite higher in comparison with that of sigmoid function.

4.3.3 RECTIFIED LINEAR UNIT

A Rectified linear unit is a unique activation function in the representation of artificial neural network which typically thrusts its focus only over the positive part of the input parameter. Unlike sigmoid or hyperbolic tangent functions the range of rectified linear unit is undefined. It is a ramp type

function that is used in control systems and hence the value escalates continuously for larger inputs. But the lower bound of the function is set to zero which connotes that no negative value will be considered in the network.

In terms of mathematical approximation, the rectifier analytic function which is the soft plus function is the closest representation to the rectified linear unit function. As the name suggests, the graph is linear in nature and the derivative of this function is normal logistic sigmoid function. The characteristic nature of the relu function in portrayed in Fig 4.3. The relu function is represented in Eqn 4.5 and its first order derivative is given in Eqn 4.6 respectively.

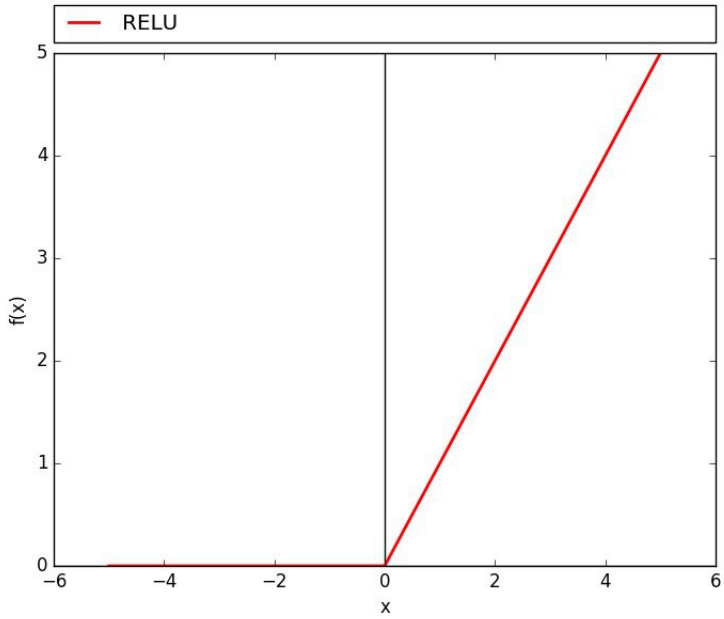


Fig 4.3 CHARACTERISTICS OF RELU FUNCTION

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \quad \text{... (4.5)}$$

$$f'(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases} \quad \text{... (4.6)}$$

This is one of the best performing activation functions in recent times and has been extensively used for deep neural network especially in the convolutional and recurrent domains. Several variants of Rectified linear unit have been used in recent times that outperform it including Leaky rectified linear unit, Parametric rectified linear unit and exponential linear unit.

4.3.4 LEAKY RECTIFIED LINEAR UNIT

The Leaky Rectified linear unit is an extension of the basic rectified linear unit and hence possess all the basic properties of normal rectified linear unit. It has been observed that the rectified linear unit network outputs only positive value and hence the network doesn't take into account the negative segment of units. The characteristic nature of the leaky relu function in portrayed in Fig 4.4. The leaky relu function is represented in Eqn 4.7 and its first order derivative is given in Eqn 4.8 respectively.

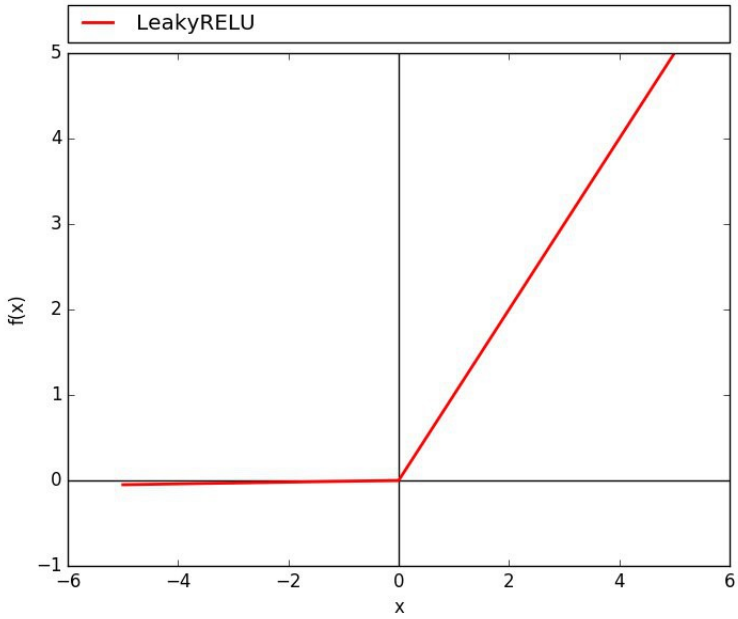


Fig 4.4 CHARACTERISTICS OF LEAKY RELU FUNCTION

This leads to a problem known as “Dying neurons” which typically denotes that the neurons with negative values are typically never fired leading to a bias for the positive values.

$$f(x) = \begin{cases} 0.01x, & x < 0 \\ x, & x \geq 0 \end{cases} \quad \text{... (4.7)}$$

$$f'(x) = \begin{cases} 0.01, & x < 0 \\ 1, & x \geq 0 \end{cases} \quad \text{...(4.8)}$$

The learning of the network is also compromised to an extent because of this bias. Hence to take the negative values into consideration, we use a small, non-zero gradient for the negative inputs. This is termed as leaky rectified linear unit as the negative portion of the graph has values whereas there is no negative portion is rectified linear unit.

4.3.5 EXPONENTIAL LINEAR UNIT

Like corrected direct units (ReLUs), leaky ReLUs (LReLUs) and parameterized ReLUs (PReLUs), ELUs likewise dodge a vanishing angle by means of the character for positive esteems. However ELUs have enhanced learning attributes thought about to the next actuation capacities. Rather than ReLUs, ELUs have negative values which enable them to push mean unit enactments more like zero. Zero implies accelerate learning since they convey the inclination nearer to the unit common angle.

Like cluster standardization, ELUs push the mean towards zero, however, with an essentially littler computational impression. While other actuation capacities like LReLUs and PReLUs additionally have negative esteems, they don't guarantee a clamor powerful deactivation state. ELU represents negative values as an incentive with littler sources of info and along these lines diminish the engendered variety and data.

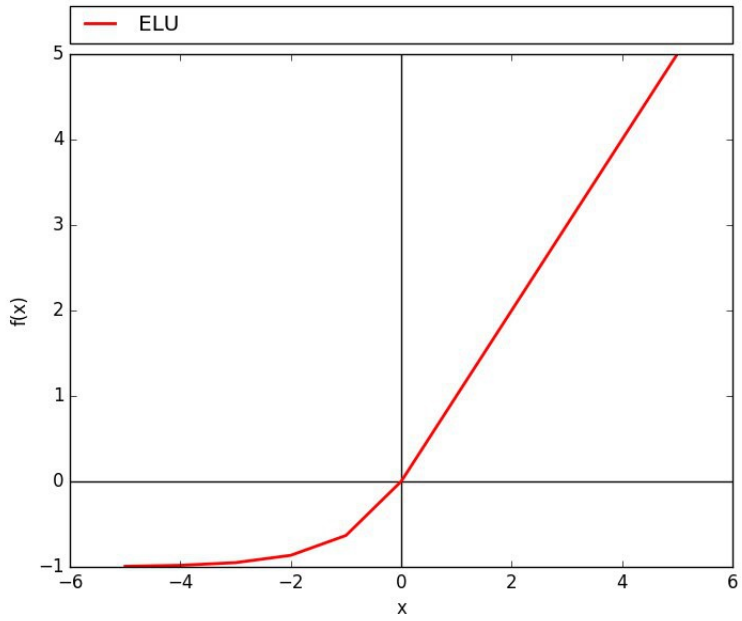


Fig 4.5 CHARACTERISTICS OF ELU FUNCTION

The characteristic nature of the Elu function in portrayed in Fig 4.5. The Elu function is represented in Eqn 4.9 and its first order derivative is given in Eqn 4.10 respectively.

$$f(x) = \begin{cases} \alpha(e^x - 1), & x < 0 \\ x, & x \geq 0 \end{cases} \quad \text{... (4.9)}$$

$$f'(x) = \begin{cases} f(x) + \alpha, & x < 0 \\ 1, & x \geq 0 \end{cases} \quad \text{... (4.10)}$$

In this way ELUs code the level of quality of specific wonders in the input, while they don't quantitatively demonstrate the level of their nonattendance.

4.3.6 PROPOSED INVERSE EXPONENTIAL LINEAR UNIT FUNCTION

The Inverse exponential linear unit function (IELU) that we have defined is inspired from the structure of the Exponential linear unit. The introduction of another exponential component helps in achieving a more generalizable

equation when compared with rectified linear unit and exponential unit. In comparison with ELU, the Inverse exponential linear unit function has a faster learning rate which leads to higher classification accuracies.

Alpha is the hyper parameter and initially the hyperparameter is set to 1.The constraint for the hyperparameter is that it is always greater than or equal to 0.When alpha is 1, the inverse exponential linear unit function is an intermediate representation of both relu and elu thereby combining their desirable features. Along with that,the inverse exponential linear unit function satisfies all the three basic properties- avoiding vanishing, avoiding exploding gradient and is zero centric.

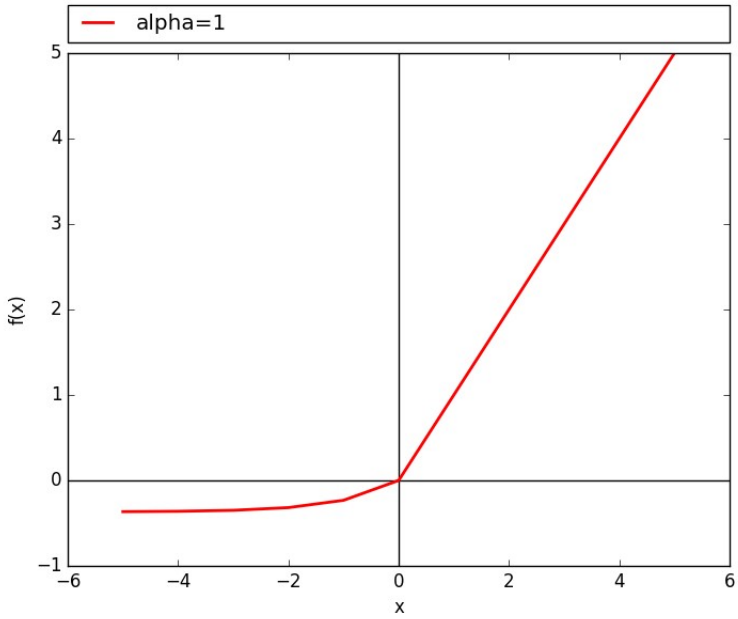


Fig 4.6 CHARACTERISTICS OF IELU FUNCTION

The characteristic nature of the Ielu function in portrayed in Fig 4.6. The Ielu function is represented in Eqn 4.11 and its first order derivative is given in Eqn 4.12 respectively.

$$f(x) = \begin{cases} x, & x \geq 0 \\ e^{-\alpha}(e^{x} - 1), & x < 0 \end{cases} \quad \text{... (4.11)}$$

$$f'(x) = \begin{cases} 1, & x \geq 0 \\ e^{-\alpha+x}, & x < 0 \end{cases} \quad \text{... (4.12)}$$

For different values of the hyper-parameter, the inverse exponential linear unit function exhibits different properties. For greater values of alpha,it functions similar to leakyrelu and relu. For smaller values of alpha,it functions similar to ELU.

4.3.7 SOFTMAX FUNCTION

The Softmax function which in other words is also termed as normalized exponential function is a generalized version of the logistic function for a particular set of inputs. The main purpose of this activation function is to normalize a set of inputs on a scale from 0 to 1 indicating the fraction of their contribution to that set. The Softmax function is represented in Eqn 4.13.

$$f(x) = \frac{e^{x}}{\sum_{i=1}^{k} e^{x_i}} \quad \text{... (4.13)}$$

This is extensively used as a probability based metric for neural networks as it acts as a gradient log based normalization for categorical probability distribution. This function is used at the last layer of most of the classification based network in order to determine the relevant output class of an input. It is specifically helpful in classifiers if they are trained using log based loss or cross-entropy based function.

4.4 ANALYSIS OF INVERSE EXPONENTIAL LINEAR UNIT FUNCTION

As stated before, the inverse exponential linear unit function that we have defined is inspired from the structure of the Exponential linear unit. The introduction of another exponential component helps in achieving a more generalized equation when compared with rectified linear unit and exponential unit. This has been explicitly analysed by comparing the graph structures of all the activation functions in juxtaposition to one another in Fig 4.7.

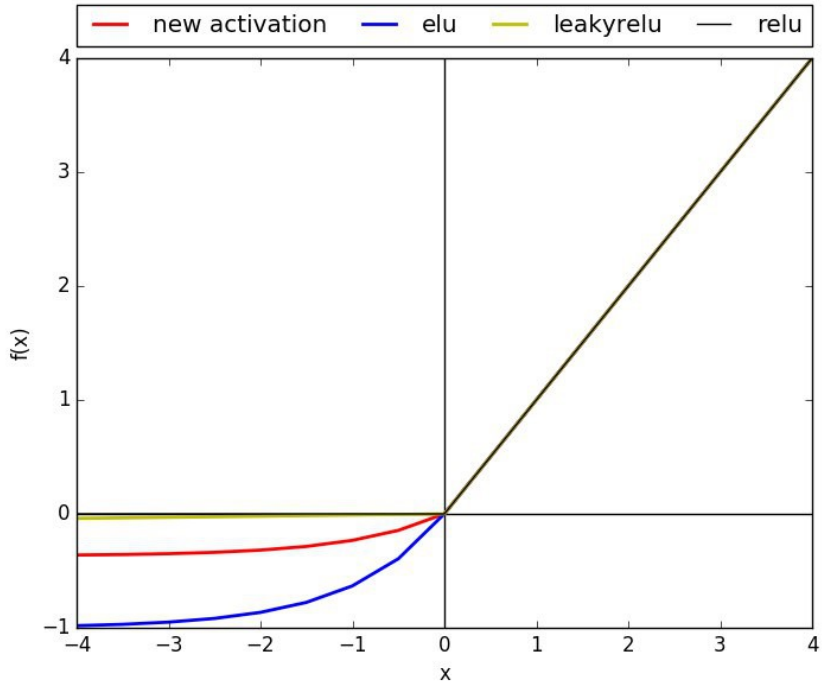


Fig 4.7 CHARACTERISTICS COMPARISON OF VARIOUS ACTIVATION FUNCTIONS

Alpha is the hyper parameter and initially the hyperparameter is set to 1.The constraint for the hyperparameter is that it is always greater than or equal to 0.When alpha is 1, the inverse exponential linear unit function is an intermediate representation of both relu and elu thereby combining their desirable features. Along with that,the inverse exponential linear unit function satisfies all the three basic properties- avoiding vanishing, avoiding exploding

gradient and is zero centric.For different values of the hyperparameter,the new function exhibits different properties. For greater values of alpha,it functions similar to leakyrelu and relu. For smaller values of alpha,it functions similar to ELU. This can be seen in Fig 4.8.

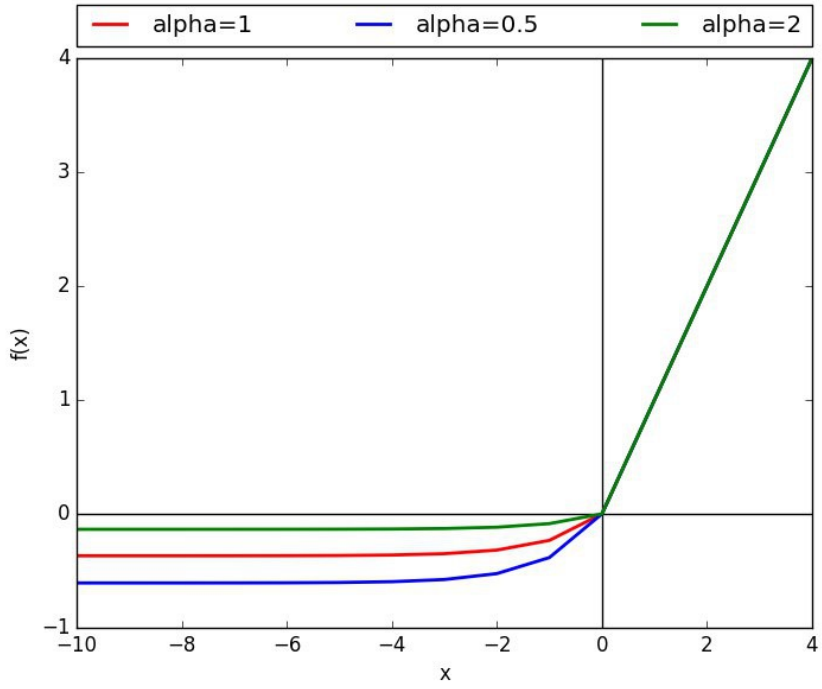


Fig 4.8 VARYING THE HYPER PARAMETERS OF IELU FUNCTION

The hyperparameter α controls the value to which the inverse exponential linear unit function saturates for negative inputs. Inverse exponential linear unit and ELUs have very parallel curves so at a high level one would expect tosee the same general characteristics in most cases. Inverse exponential linear unit function have smooth and continuous first derivative. In contrast, ReLU is non-differentiable at zero. Since Inverse exponential linear unit and ELUs share most of the same characteristics we use the (e^-alpha) times weight initialization guidelines as are used for ELUs.

The advantage of inverse exponential linear unit is its reduced exponentially compared to ELU. When calculating, inverse exponential linear unit for negative inputs, first we calculate e^ -1. Multiplying this function by e^-

α provides the value for theforward calculation. While the other activation functions have complex back-propagation function, by adding the e^-α value alone from the original equation,we get the value for back propagation, so the back propagation can be calculated faster. ELU saturates at 1 when its α = 1.

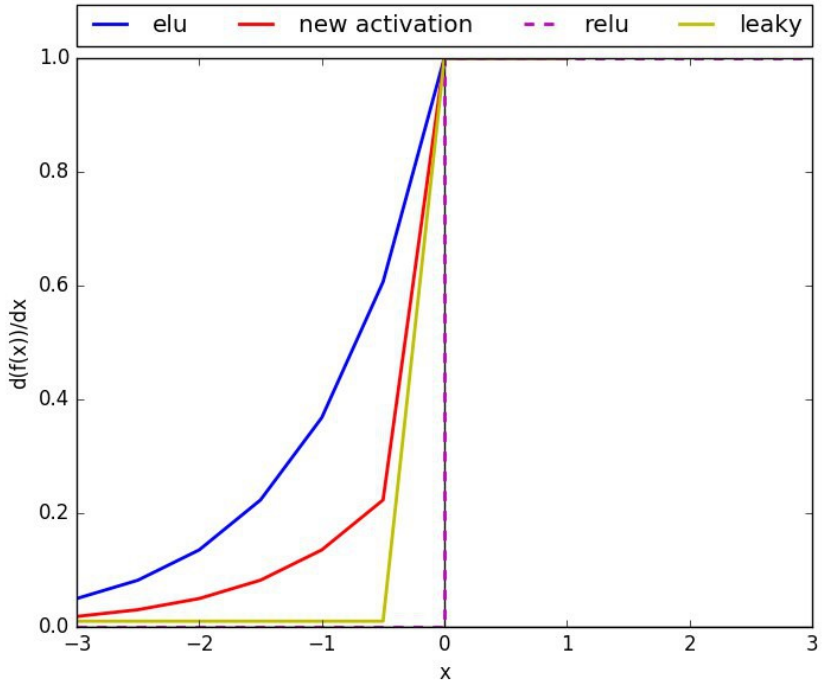


Fig 4.9 DERIVATIVE ANALYSIS OF ACTIVATION FUNCTIONS

When we interpret the derivative analysis of each activation function as depicted in Fig 4.9 we can see that with α = 1, Inverse exponential linear unit saturation approaches -0.37. With α = 0.5, the negative saturation is reduced, so a smaller portion of the back-propagated error signal will pass to the next layer. This allows the network to output sparse activations while preserving its ability to reactivate dead neurons. Note that under variations of theα parameter,Inverse exponential linear unit is a family of continuous curves which preserve their smooth derivatives, and transition between the functional bounds of ($\alpha \rightarrow -e^{-\alpha}$) and linear ($\alpha \rightarrow \infty$).

4.5 CHAPTER SUMMARY

Conclusively, we can assert that the Inverse exponential linear unit function that we have defined is inspired from the structure of the Exponential linear unit. This is because of the introduction of another exponential component which helps in achieving a more generalizable equation when compared with rectified linear unit and exponential unit. Hence, In comparison with Rectified linear unit, the Inverse exponential linear unit function has a faster learning rate which leads to high classification accuracies.

CHAPTER 5
IMPLEMENTATION AND RESULTS

5.1 IMPLEMENTATION OVERVIEW

Chapter 5 discusses the implementation details and results of the proposed deep learning based text classification system implemented with the inverse exponential linear unit function. It describes various tools used to develop the system along with the analysis of the output. The three variants of the parallel architectures are designed using the Tensoflow library and the datasets that are used have also been discussed. The extensive performance analysis that has been conducted is also reported.

5.2 TOOLKIT

The Toolkit required to design the deep learning based parallel architectures involving recurrent and convolutional neural networks involve several library packages like Tensorflow, Keras, Sci-kit learn, Natural Language toolkit and also Matplotlib packages. All the codes are written using Python programming language by utilising the above packages.

5.2.1 TENSORFLOW

Tensorflow is an open source software library written in Python, C++ and CUDA. It is fast, easy to install, and supports CPU and GPU computation. The purpose of this library is for empowering users to write dataflow based programming across a range of tasks especially for applications pertaining to machine learning such as construction of Neural Networks, KNN classifiers, Support Vector machines. The Deep Neural Network is run on theTensorflow library. The Tensorflow evaluates and empowers researchers to train any neural network based on specific requirements. The final trained network can be used totest over to measure the performance.

5.2.2 KERAS

Though Tensorflow allows the researchers and students to run and test neural network based models, the level of abstraction and representation is convolved. Keras is a high-level Application Programming Interface that is solely designed for the purpose of implementing neural network based architectures. Keras specifically focuses on the Neural network segment of the machine learning domain whereas Tensorflow covers other domains too. This high-level API is compatible with all the famous software libraries like Theano, Tensorflow and also Cognitive Toolkit. It caters to the need of fast experimentation empowering analysts to quickly design, train and test various models. Also Keras is responsible for swift and easy prototyping of models and supports any combination of convolutional and recurrent neural networks. Few of the noteworthy properties of Keras include User friendliness, modularity, compatibility with python and extensibility with other modules.

5.2.3 SCIKIT LEARN

Scikit-learn is an open source machine learning based software library for Python. It contains a collection of packages for performing machine learning based tasks like regression, categorization, clustering which in turn contains support vector machines, k-means, gradient boosting and even random forests. One of the unique features of this library is that it is highly compatible with other python libraries such as SciPy and NumPy. This library also contains various collections of standard datasets such as the 20 Newsgroup dataset for running the deep learning model. Few other features that can be performed by this library include data pre-processing, model selection and also dimensionality reduction.

5.2.4 NATURAL LANGUAGE TOOLKIT

Natural Language Toolkit is an open source library for statistical natural language processing. It is written in python programming language and mainly aims to promote research and learning in the field of natural language processing with major focus in regions pertaining to linguistics, information retrieval, machine learning, cognitive sciences and also artificial intelligence. It has also been used as a platform for prototype development and testing of systems.

5.2.5 MATPLOTLIB

Matplotlib is a two dimensional plotting library for Python programming language which is employed to generate professional quality figures in a variety of softcopy and hardcopy formats. Data visualization is an integral part of any research practise. Hence it becomes almost indispensible to utilize resources like this to comprehend and contemplate over the outcomes and by-products of any research experiment. It encompasses an object-oriented application programming interface for embedding the generated plots into applications including general purpose graphical user interfaces. Matplotlib is one of the excellent visualization tools to comprehend the relation among various data. Few of the variety of generations of Matplotlib include plots, histograms, bar charts, scatter plots and even error charts.

5.3 DATASET

A Dataset typically refers to an assortment or accumulation of data that are similar to each other in some aspect. They are usually collected in order to identify the common or disparate aspects among them and to reason the empirical observation. A dataset can be in many forms like a database table or a statistical data matrix. These datasets are collected manually throughout time as a temporal sequence of collection which are then analysed for reasoning. The

dataset may contain any type of attributes like real numbers, integers, text, image or even audio signals. It is necessary to choose the correct dataset depending on the purpose of the dataset.

5.3.1 REUTER NEWS ARTICLES

Reuters is an ensemble of business oriented articles and it can also be perceived asa high preferred benchmark dataset when it comes to document categorization. Reuters Ltd has provided a huge collection of Reuter stories for utilization in the research and development of natural language processing, information retrieval, and machine learning systems. Thedataset of 11,228 newswires from Reuters are used and they are labelled over 46 topics. In this dataset, the maximum length of an input sample is 15771 characters while on the contrary minimum length of an input sample is 2372 characters. The document collection is split as 8982 documents for training, 1123 documents for validation and 1123 documents for the purpose of testing the model. The various classes in the Reuter dataset are:

[earn,acq, money-fx, grain, crude, trade, interest, wheat, ship, corn, money-supply, dlr, sugar, oilseed, coffee, gnp, gold, veg-oil, soybean, livestock, nat-gas, bop, cpi, cocoa, reserves, carcass, copper, jobs, yen, ipi, iron-steel, cotton, barley, gas, rubber, alum, rice, palm-oil, meal-feed, sorghum, retail, zinc, silver, pet-chem, wpi, tin]

5.3.2 20-NEWSGROUP ARTICLES

The 20 Newsgroups data set is a corpora of articles that is concerned with news related issues and there are about 19000literatures, splitequally across 20 different sections. The 20 newsgroups collection is an important benchmark dataset for experiments in text applications of machine learning techniques, such as text classification and text clustering. In this dataset, the maximum length of an input sample is 160616 characters and the minimum length of an

input sample is 115 characters. The document collection is split as 11307 documents for training, 3770 documents for validation and 3769 documents for the purpose of testing the model. The various classes in the 20 Newsgroup dataset are : [‘graphics', ‘pc.hardware ', ‘mac.hardware’, ‘windows.x', ‘ms-windows.misc', ‘forsale', ‘autos', ‘motorcycles', ‘baseball', ‘hockey', ‘guns', ‘mideast', ‘crypt', ‘electronics',‘med‘, ‘space‘, ‘atheism‘, ‘christian‘, ‘religion.misc‘, ‘politics.misc']

5.3.3 TECHCRUNCH ARTICLES

TechCrunch is a technology related article publishing firm that is mainly concerned with emerging start-ups, cutting edge technology and other technical news. It contains 39000 samples of English-language text, collected from articles published in the Tech Crunch website which is split across 7 classes. In the dataset, the maximum length of an input sample is 31948 characters while the minimum length of an input sample is 18 characters. The document collection is split as 23400 documents for training, 7800 documents for validation and 7800 documents for the purpose of testing the model. The various classes in the TechCrunch dataset are:['startups', 'mobile', 'gadgets', 'enterprise', 'social', 'europe', 'asia‘]

5.3.4 BROWN CORPUS

The Brown University in the previous century have constructed a standard corpus of English oriented literatures and has been deployed as one of the most preferred dataset collection in the division of linguistics and is also perceived as one of the benchmark datasets for text segregation. It proposes to have around five hundred samples of English based texts, totalling approximatelyten lakh words, gathered from works deployed in the United States that is split across 15 different classes. Within the dataset, the maximum length of an input sample is 14653 characters and the minimum length of an

input sample is 10469 characters. The document collection is split as 300 documents for training, 100 documents for validation and 100 documents for the purpose of testing the model. The various classes in the Brown Corpus dataset are:['adventure', 'belles_lettres', 'editorial', 'fiction', 'government', ‘hobbies', 'humor', 'learned', 'lore', 'mystery', 'news', 'religion', 'reviews', 'romance', 'science_fiction']

5.3.5 OHSUMED MEDICAL DATASET

Ohsumed dataset is a collection of medical records that maintains the case histories of patients suffering from cardiovascular diseases. It is a peer-reviewed medical literature that holds all the required records of the 7 important cardio-vascular diseases. The collection consists of almost 5000 documents and each one of them classified under one of the 7 classes. Within the dataset, the maximum length of an input sample is1830 characters and the minimum length of an input sample is 68 characters. The document collection is split as 3000 documents for training, 1000 documents for validation and 1000 documents for the purpose of testing the model. The various classes in the Ohsumed dataset are:['Neg-','Pos-Hyperplasia','Pos-Mitosis', 'Pos-Necrosis','Pos-Pediatrics','Pos-Preg','Pos-Rats']

5.4 SOFTWARE INSTALLATION

Installing software such as NVIDIA Drivers, cuDA, cuDNN, Tensorflow, Keras, NumPy, Matplotlib, SciPy, Scikit-learn and Graphviz are done in Ubuntu environment can be done by following the set of commands mentioned in the Annexe I.

5.5 CODE ORGANISATION

The codes written for the implementation of the proposed three architectures are written in python programming language. Each of the parallel

architectures is trained on all the mentioned datasets and the corresponding performance of each of the models is evaluated. The code written is compatible with normal CPU based system. At the same time, the same code can be run on a GPU processor like NVIDIA cuDA. The datasets used are either directly downloaded from the python library natural language toolkit or it is stored in a comma separated value file. The word embedding vectors are stored in a bin file for quick loading and access. The performance analysis of all the models are stored and visualized from an excel file.

The following Table 5.1contains the description of the functionality of all the files in the system.

Table 5.1 CODE ORGANISATION OF FILES

FILE NAME	DESCRIPTION
GoogleNews-vectors-negative300.bin	GoogleNews vectors that are used for the purpose of word embeddings. Contains 3 million words and each word has a 300 dimensional vector.
Reuter-Architecture1.py	The first architecture model analysed using Reuter dataset
Reuter-Architecture2.py	The second architecture model analysed using Reuter dataset
Reuter-Architecture3.py	The third architecture model analysed using Reuter dataset
20NG-Architecture1.py	The first architecture model analysed using 20 NewsGroup dataset
20NG-Architecture2.py	The second architecture model analysed using 20 NewsGroup dataset
20NG-Architecture3.py	The third architecture model analysed using 20 NewsGroup dataset

TechCrunch-Architecture1.py	The first architecture model analysed using TechCrunch dataset
TechCrunch-Architecture2.py	The second architecture model analysed using TechCrunch dataset
TechCrunch-Architecture3.py	The third architecture model analysed using TechCrunch dataset
Brown-Architecture1.py	The first architecture model analysed using Brown dataset
Brown-Architecture2.py	The second architecture model analysed using Brown dataset
Brown-Architecture3.py	The third architecture model analysed using Brown dataset
Ohsumed-Architecture1.py	The first architecture model analysed using Ohsumed dataset
Ohsumed-Architecture2.py	The second architecture model analysed using Ohsumed dataset
Ohsumed-Architecture3.py	The third architecture model analysed using Ohsumed dataset
Ohsumed.csv	The dataset file that contains all the medical text records along with the corresponding class labels.
Techcrunch.csv	The dataset file that contains all the technical news articles along with the corresponding class labels.
Performance_analysis.xls	The complete set of documented analysis of the performance of each model over every dataset.

5.6 PERFORMANCE EVALUATION

Our proposed approach is predicted to be better because of the following reasons:

Compared with the existing sequential architecture, a parallel model is capable of utilizing the unique structure of two sequential architectures in such a manner that only the prominent features among the branches will be utilized. Thus the parallel architecture has the opportunity to capture more features of varying architectures and hence the ability to comprehend more parameters and thus the classification performance can be improved.

Also, while combining a convolutional neural network and recurrent neural network there is a significant feature reduction technique that can be utilised. Since a convolutional neural network is a window frame based feature extractor, if we assign the window size as n, then each n-gram words will be condensed into a high level representation. When this n-frame condensed output is fed into the recurrent neural network which is capable of learning long sequences of temporal input, features are better collected by the network. Also, since convolutional neural network captures spatial relation among the inputs while the recurrent neural network captures temporal relation among the inputs, amalgamating them will certainly preserve the spatial-temporal relation among the input data.

Hence combining these results, the proposed approach is hypothesised to be better for improving the performance of the text classification of large datasets.

5.6.1 EXPERIMENTAL SETUP

The experimental dataset involved for the purpose of training and testing the validity of any of the proposed models are a collection of text passages where every document identifies itself with one specific category. Once the program is loaded into the processor, the dataset to be utilized is loaded into the

memory as a pair of input-output list where for each entry in the input list there exist a counterpart in the output list. After loading the dataset into the memory, the next task is to trim the number of lexemes in the input documentto a standard fixed range. With the help of all the inbuilt pre-processing libraries in python programming language we tune the input and output lists into the desired format.

Eventually this processed dataset is split into training, validation and testing dataset samples. Then the architecture model which is to be analysed is defined by enunciating all the required convolutional neural network models and the recurrent neural network models. After defining the architecture, the next rational step will be to compile the designed agglomerate network. The compilation phase is to ensure that the network is actually created based on the description given. After compilation, the next step is to train the designed model. This training phase is used to ensure that the model analyzes the content of the input, to gather features and interpolate the relation among the feature extracted with respect to the class to which each sample belongs to. For each epoch, along with the training we perform a task known as Validation.

In the Validation phase, we run the validation data in the trained model to ensure that the training features don't narrow down the feature range of the values in the network model and the generality of the features remains intact. When we find a middle ground between the validation and training outputs, we lock the optimum point where learning has been completed for that model. The testing phase is the actual challenge posed to the model where the practical performance of the model is measured and analyzed. Finally the actual output is evaluated and compared in juxtaposition with the expected output to analyze the performance of the network. This performance is analysed by using the scikit-learn library package in python.

5.6.2 PERFORMANCE METRICS

The performance of the system as a whole is analysed by implementing the system in a Linux environment supported by NVIDIA GPU and cuDNN. The datasets are evaluated based on certain specific parameters like Accuracy, Loss, Precision, Recall and F-1 Score. In order to calculate the above performance benchmarks, we need to define several terms such as true positive, true negative, false positive and false negatives.In classification zone of machine learning the below four concepts are of prime importance.

True Positive TP – When the predicted class is same as that of the ideal truth.

True Negative TN – When the algorithm fails to identify the ideal truth class.

False Positive FP – When the predicted class identifies non-existent truth class.

False Negative FN – When the algorithm doesn't identify non-existent classes.

When focussing the field of data extrapulation, the concept of precision is deeply associated with the fraction of documents retrieved that are actually congruent to the question proposed. In empirical terms, it is the quantity ofcongruent documents extracted divided by the total quantity of extracted documents. Mathematically, this is represented in Eqn 5.1.

$$\textbf{Precision} = \frac{\mathbf{TP}}{\mathbf{TP+FP}} \qquad \textbf{... (5.1)}$$

When considering the field of information retrieval, the concept of recall is linked with the total fraction of congruenttext collection that are successfully extracted. In empirical terms, it is the quantity of required documents extracted divided by the total quantity of required literatures. Mathematically, this is represented in Eqn 5.2.

$$\textbf{Recall} = \frac{\mathbf{TP}}{\mathbf{TP+FN}} \qquad \textbf{... (5.2)}$$

True Negative Rate can be defined as the instances where the systems have not classified an anomalous entity. Mathematically, this is represented in Eqn 5.3.

$$\textbf{True Negative Rate} = \frac{\mathbf{TN}}{\mathbf{TN+FP}} \quad \textbf{... (5.3)}$$

Accuracy can be defined as the total number of relevant documents successfully retrieved based on the query proposed. Mathematically, this is represented in Eqn 5.4.

$$\textbf{Accuracy} = \frac{\mathbf{TP+TN}}{\mathbf{TP+TN+FP+FN}} \quad \textbf{... (5.4)}$$

A method of successfully combining precision and recall so that both the parameters are equally involved is the F-1 score. It can be considered as a mathematical harmonic mean of the precision and recall so that they are evenly weighed. It is a technique to analyse the effectiveness of the retrieval of a system where precision is given as much as importance as that of recall. Mathematically, this is represented in Eqn 5.5.

$$\textbf{F-1 score} = \frac{\mathbf{2 * Precision * Recall}}{\mathbf{Precision+Recall}} \quad \textbf{... (5.5)}$$

5.6.3 ANALYSIS OF EXPERIMENTAL RESULTS

The analysis part of the thesis involves carefully considering the way in which every model that has been designed works and the methodology that each of them adapt in order to accumulate and utilize the various features of the input data in order to derive the conclusion regarding the category to which that sample belongs to.

We also analyse the performance of all the three architecture in juxtaposition with the traditional model so that we can comprehend the

functioning of these models better. It also involves recording the variations that the models show when subjected to different activation functions. This is important because every activation function is a normalized regulator but the way in which they normalize data differs from one function to another. This is because every function has certain idiosyncrasy that can be attributed to only that function. Hence it is prominent to analyse how each of the model performs when subjected to those activation functions. This also caters to our needs of researching about the performance of the newly defined activation function and the rate at which it improves the learning ability of the models. Hence we will consider few of the high performing activation functions like rectified linear unit, leaky rectified linear unit and exponential linear unit along with our inverse exponential linear unit function. For the purpose of understanding we will refer to the traditional sequential combination of recurrent and convolutional neural network as RCSNN in our analysis.

5.6.3.1 ANALYSIS OF REUTER DATASET

When we consider the Reuter dataset that has about 11,000 documents split across 46 categories as the input to all the three models plus the sequential model then it has been inferred that sequential model is clearly outperformed by all the parallel architectures in terms of all the evaluation metrics.

When we analyse the performance of the various activation functions it has been found that our inverse exponential linear unit function outperforms all the other three standard activation functions in terms of precision, recall and f1 score and conclusively it has been identified that our parallel cross architecture when embedded with our inverse exponential linear unit function gives the highest performance among all the twelve models. This is clearly observable from the graphs illustrated.

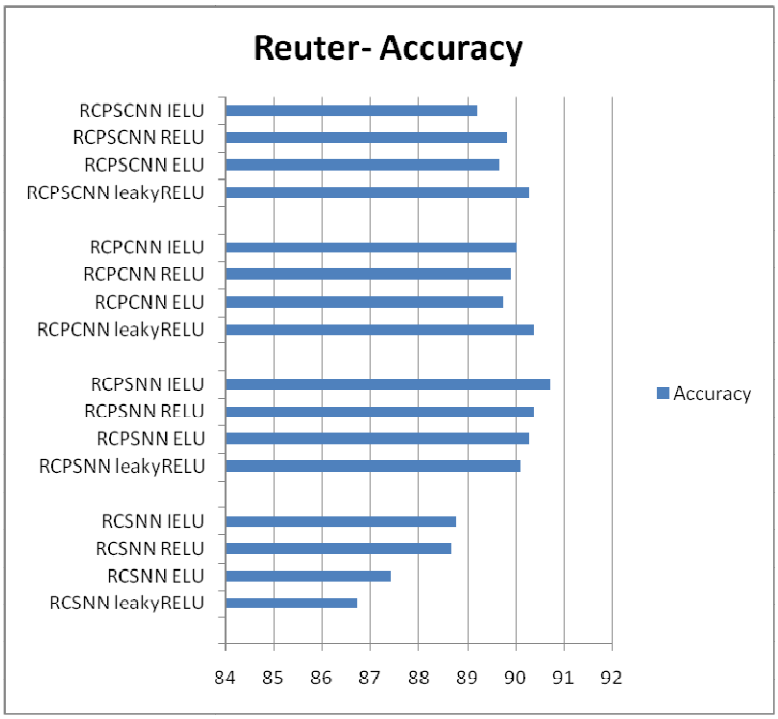


Fig 5.1 REUTER DATASET – ACCURACY ANALYSIS

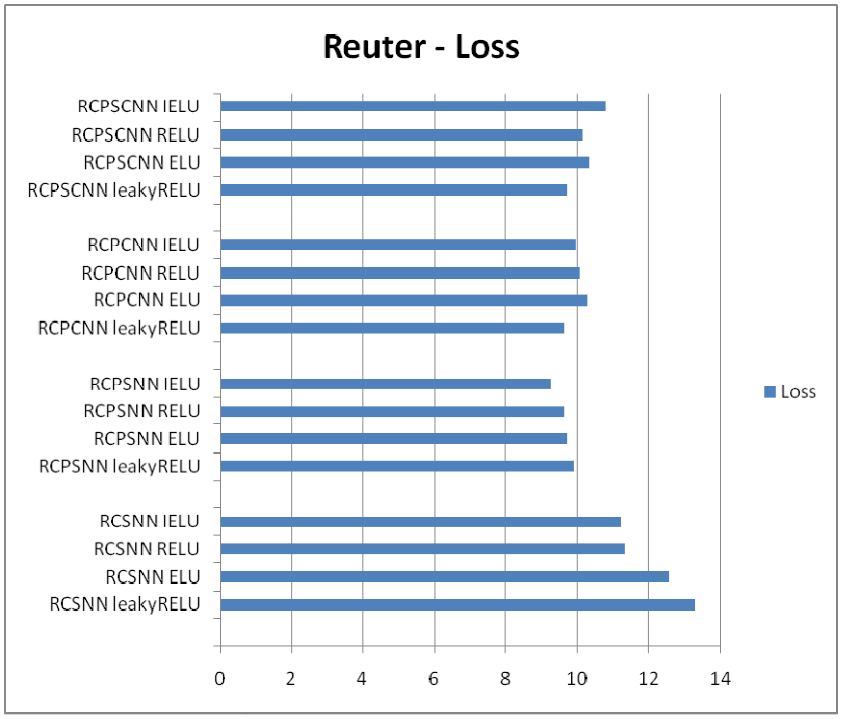


Fig 5.2 REUTER DATASET – LOSS ANALYSIS

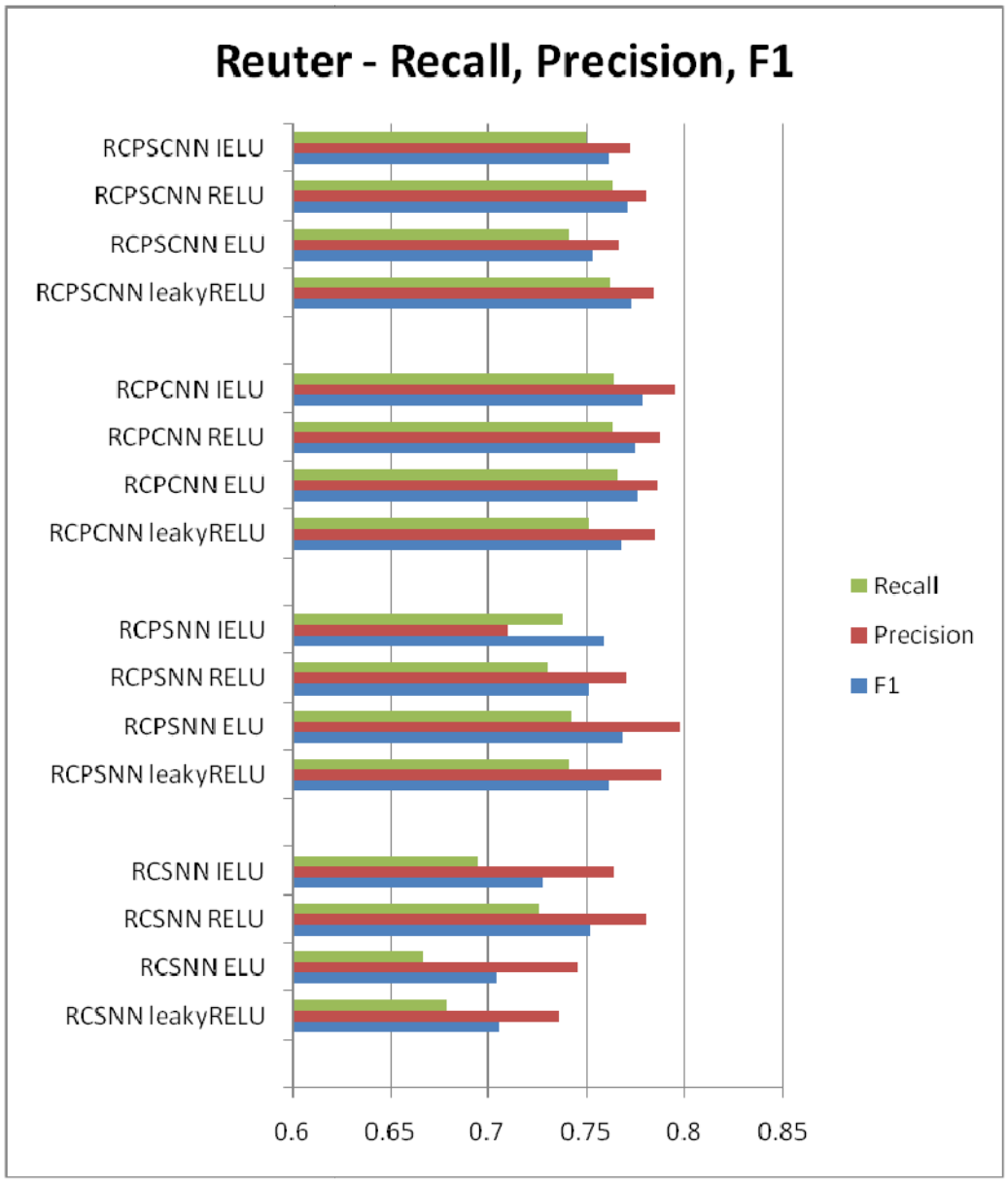


Fig 5.3 REUTER DATASET – RECALL, PRECISON, F1 SCORE ANALYSIS

We represent the performance analysis done graphically from the Fig 5.1, 5.2, 5.3 corresponding to the analysis of Accuracy, Loss, Recall, Precision and F-1 Score. Thus the analysis of Reuter dataset clearly implies that all the parallel architecture models have outperformed the basic sequential model and particularly the second parallel architecture model having the sequential as well

as the parallel mixture of convolutional and recurrent neural network is the best among these three designed models.

5.6.3.2 ANALYSIS OF 20 NEWSGROUP DATASET

When we consider the 20 Newsgroup dataset that has about 20,000 documents split across 20 categories as the input to all the three models plus the sequential model then it has been inferred that sequential model is clearly outperformed by all the parallel architectures in terms of all the evaluation metrics.

When we analyse the performance of the various activation functions it has been found that our inverse exponential linear unit function outperforms all the other three standard activation functions in terms of precision, recall and f1 score and conclusively it has been identified that our parallel straight-cross architecture when embedded with our inverse exponential linear unit function gives the highest performance among all the twelve models. This is clearly observable from the following graphs.

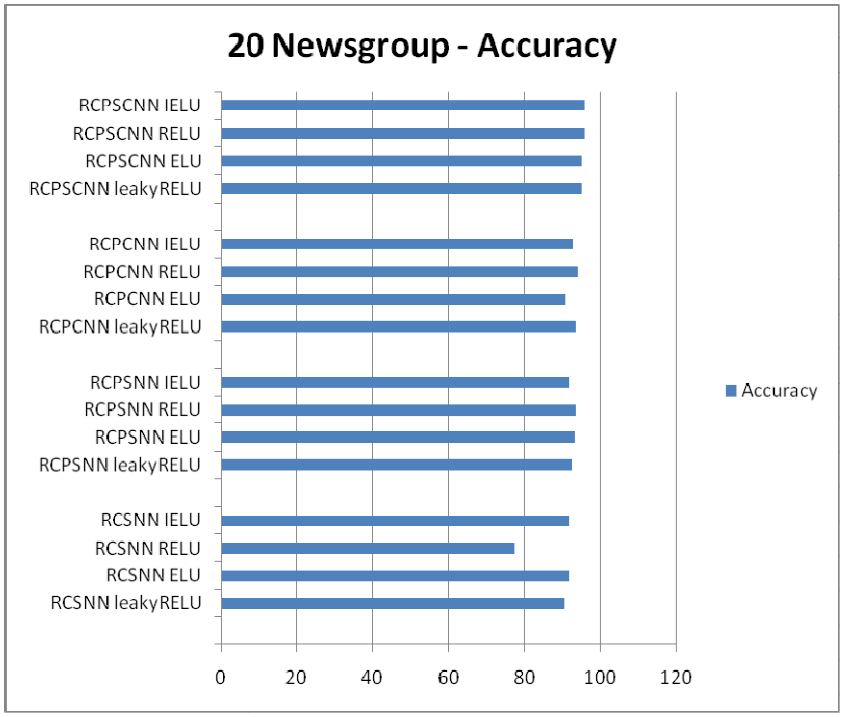


Fig 5.4 20NEWSGROUP DATASET – ACCURACY ANALYSIS

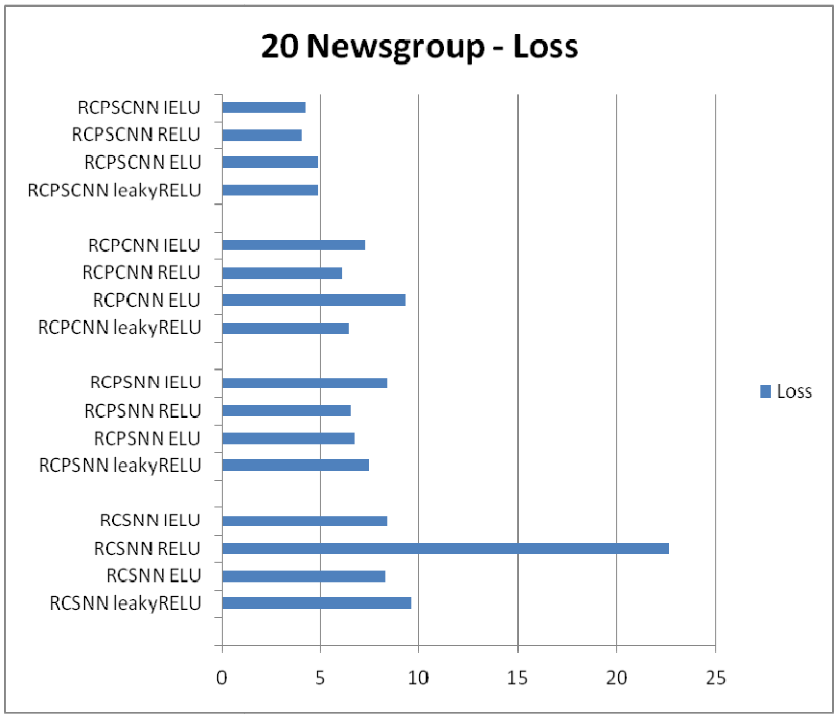


Fig 5.5 20NEWSGROUPDATASET – LOSS ANALYSIS

We represent the performance analysis done graphically from the Fig 5.4, 5.5, 5.6 corresponding to the analysis of Accuracy, Loss, Recall, Precision and F-1 Score. Thus the analysis of 20Newsgroup dataset clearly implies that all the parallel architecture models have outperformed the basic sequential model and particularly the third parallel architecture model having the sequential as well as the parallel mixture of convolutional and recurrent neural network is the best among these three designed models.

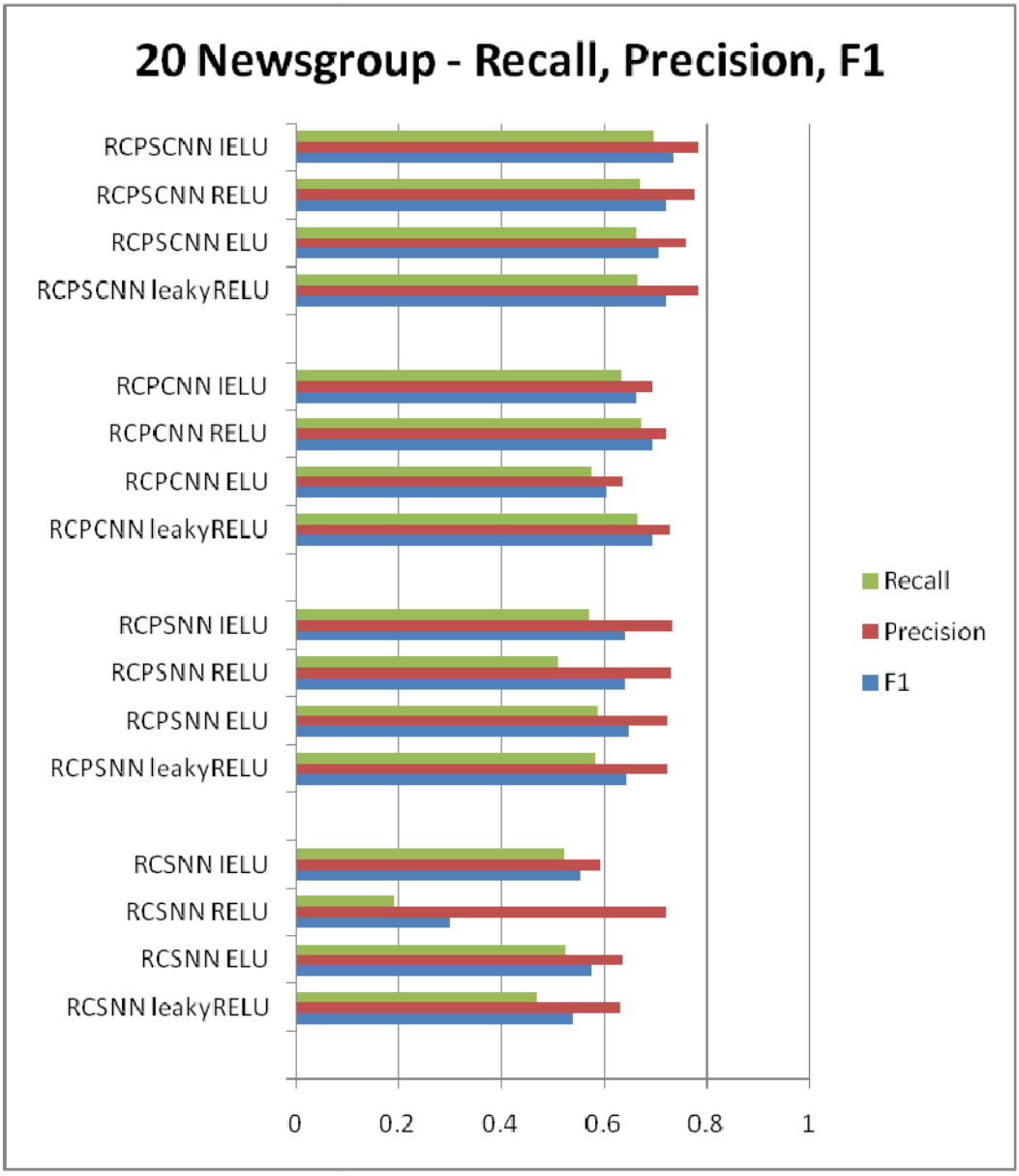


Fig 5.6 20NEWSGROUPDATASET – RECALL, PRECISON, F1 SCORE ANALYSIS

5.6.3.3 ANALYSIS OF TECHCRUNCH DATASET

When we consider the Tech-Crunch dataset that has about 39,000 documents split across 7 categories as the input to all the three models plus the sequential model then it has been inferred that sequential model is clearly outperformed by all the parallel architectures in terms of the evaluation metrics.

When we analyse the performance of the various activation functions it has been found that our inverse exponential linear unit function outperforms all the other three standard activation functions in terms of precision, recall and f1 score and conclusively it has been identified that our parallel straight cross architecture gives the highest performance among all the twelve models. This is clearly observable from the graphs illustrated.

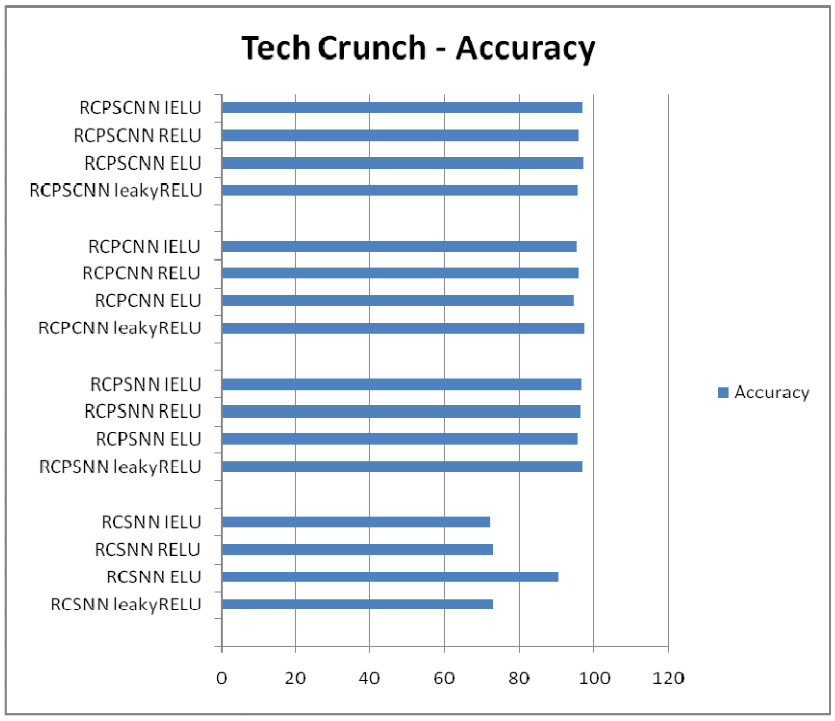


Fig 5.7 TECHCRUNCH DATASET – ACCURACY ANALYSIS

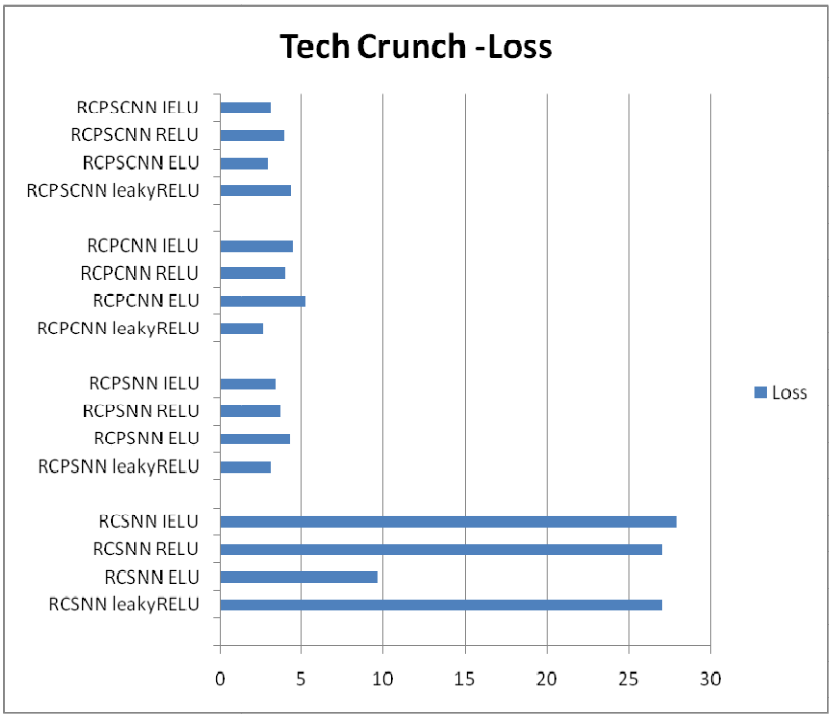


Fig 5.8 TECHCRUNCHDATASET – LOSS ANALYSIS

We represent the performance analysis done graphically from the Fig 5.7, 5.8, 5.9 corresponding to the analysis of Accuracy, Loss, Recall, Precision and F-1 Score. Thus the analysis of Techcrunch dataset clearly implies that all the parallel architecture models have outperformed the basic sequential model and particularly the third parallel architecture model having the sequential as well as the parallel mixture of convolutional and recurrent neural network is the best among these three designed models.

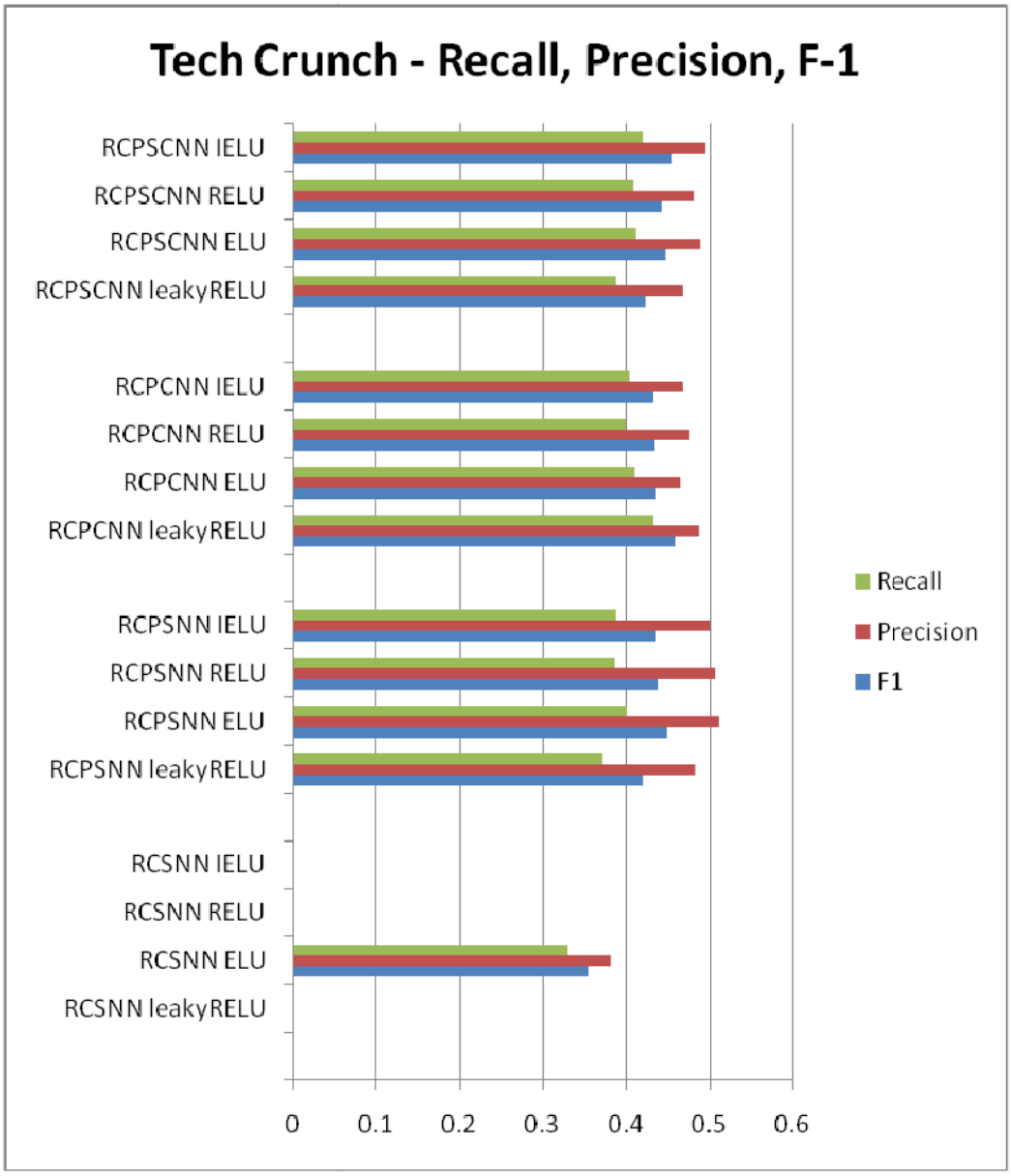


Fig 5.9 TECHCRUNCHDATASET – RECALL, PRECISION, F-1 SCORE ANALYSIS

5.6.3.4 ANALYSIS OF BROWN CORPUS

When we consider the Brown corpus that has about 500 documents split across 15 categories as the input to all the three models plus the sequential model then it has been inferred that sequential model is clearly outperformed by all the parallel architectures in terms of all the evaluation metrics.

When we analyse the performance of the various activation functions it has been found that our inverse exponential linear unit function outperforms all the other three standard activation functions in terms of precision, recall and f1 score and conclusively it has been identified that our parallel straight cross architecture when embedded with our inverse exponential linear unit function gives the highest performance among all the twelve models. This is clearly observable from the given graphs.

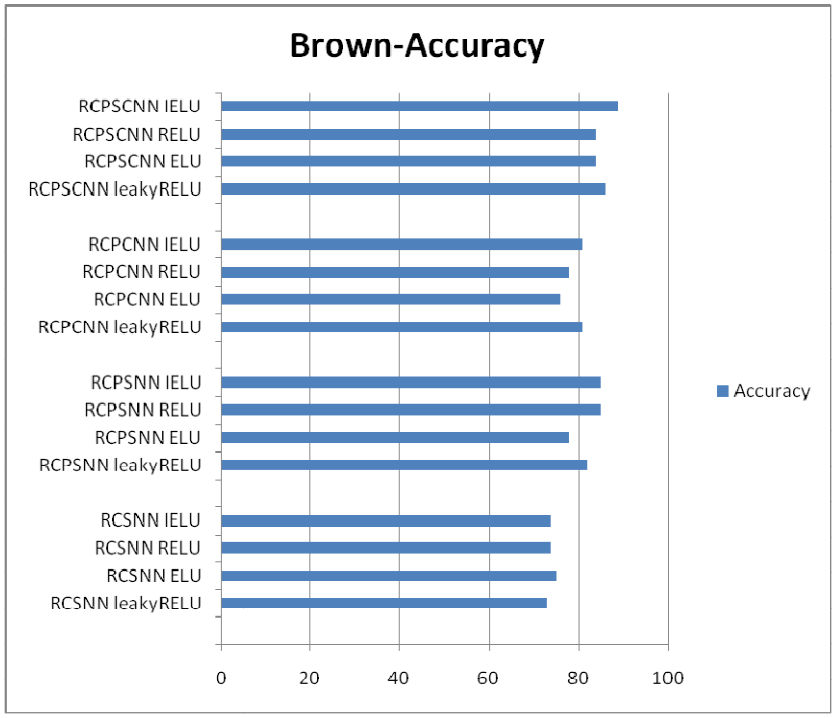


Fig 5.10 BROWN CORPUS– ACCURACY ANALYSIS

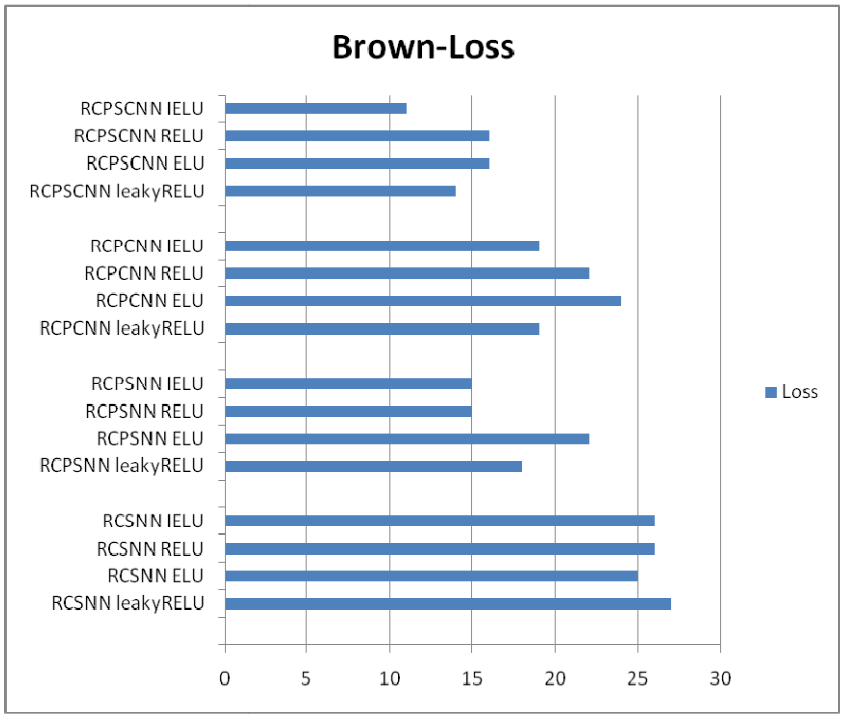


Fig 5.11 BROWN CORPUS– LOSS ANALYSIS

We represent the performance analysis done graphically from the Fig 5.10, 5.11, 5.12 corresponding to the analysis of Accuracy, Loss, Recall, Precision and F-1 Score. Thus the analysis of Brown Corpus dataset clearly implies that all the parallel architecture models have outperformed the basic sequential model and particularly the second parallel architecture model having the sequential as well as the parallel mixture of convolutional and recurrent neural network is the best among these three designed models in terms of recall and f1 score nut the third parallel architecture has higher precision based value.

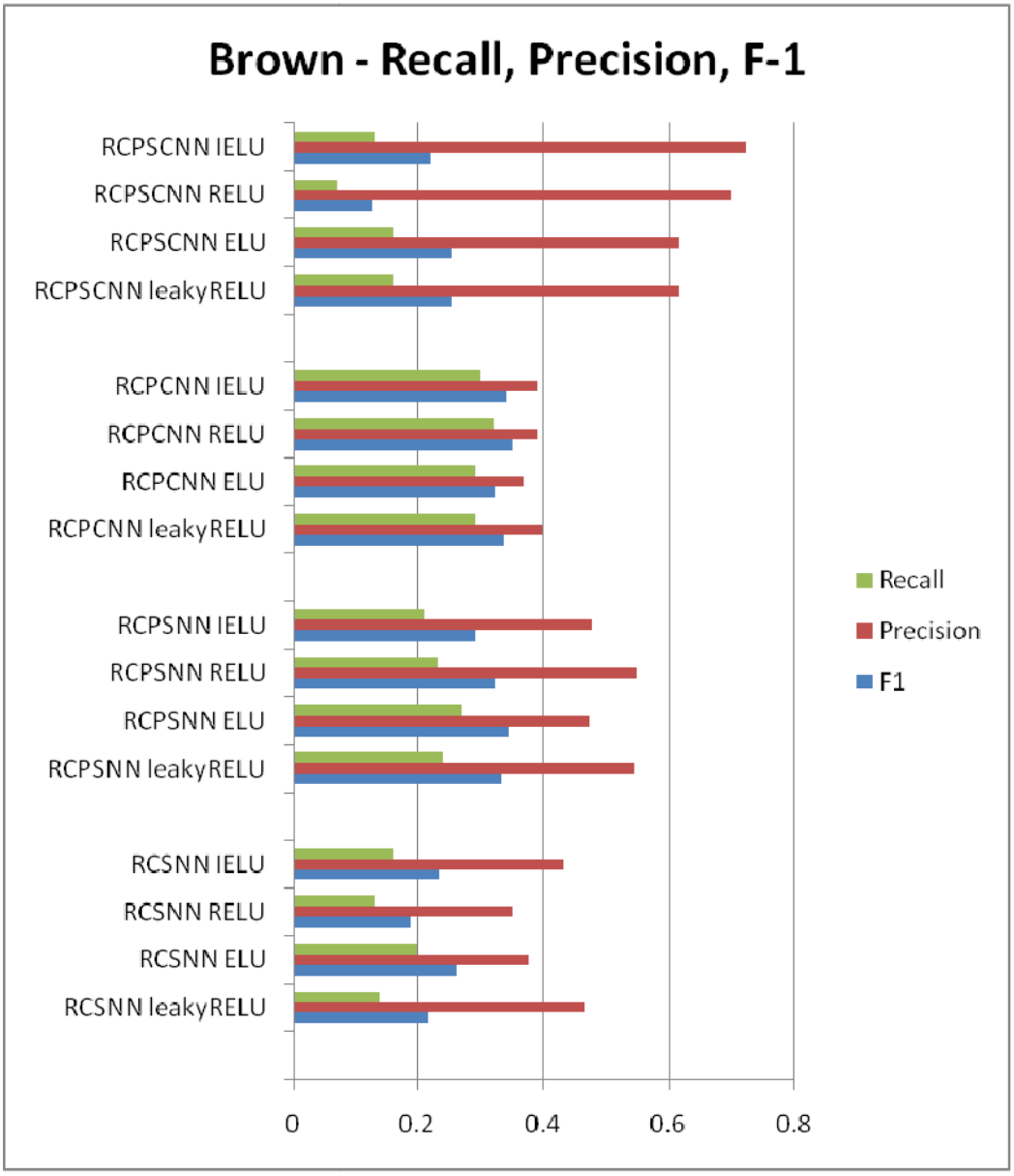


Fig 5.12 BROWN CORPUS– RECALL, PRECISION, F-1 SCORE ANALYSIS

5.6.3.5 ANALYSIS OF OHSUMED DATASET

When we consider the Ohsumed medical dataset that has about 5000 documents split across 7 categories as the input to all the three models plus the sequential model then it has been inferred that sequential model is clearly outperformed by all the parallel architectures in terms of the evaluation metrics.

When we analyse the performance of the various activation functions it has been found that our inverse exponential linear unit function outperforms all the other three standard activation functions in terms of precision, recall and f1 score and conclusively it has been identified that our parallel straight cross architecture when embedded with our inverse exponential linear unit function gives the highest performance among all the twelve models. This is clearly observable from the given graphs.

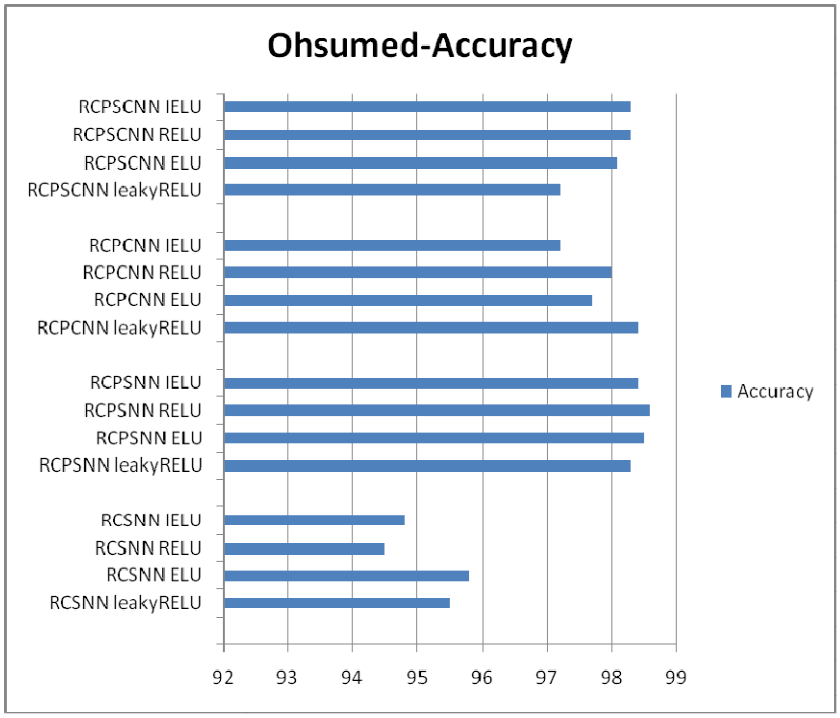


Fig 5.13 OHSUMED DATASET – ACCURACY ANALYSIS

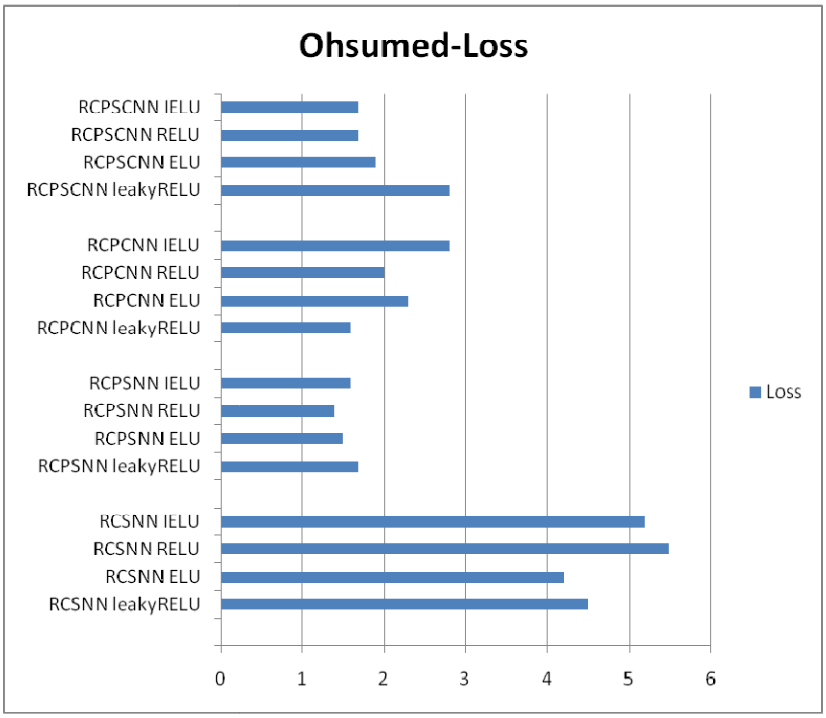


Fig 5.14 OHSUMED DATASET – LOSS ANALYSIS

We represent the performance analysis done graphically from the Fig 5.13, 5.14, 5.15 corresponding to the analysis of Accuracy, Loss, Recall, Precision and F-1 Score.Thus the analysis of Ohsumed dataset clearly implies that all the parallel architecture models have outperformed the basic sequential model and particularly the third parallel architecture model having the sequential as well as the parallel mixture of convolutional and recurrent neural network is the best among these three designed models.

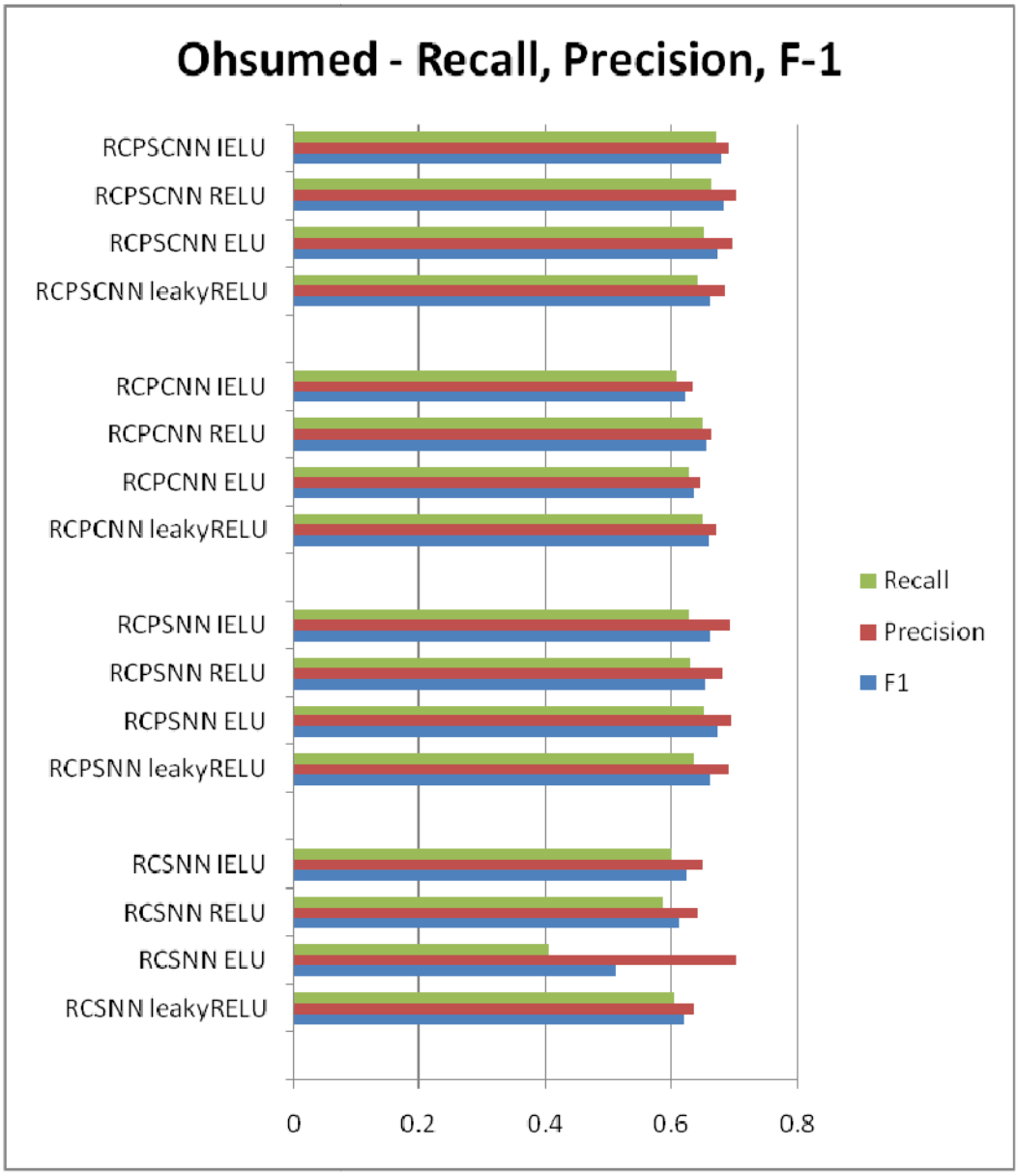


Fig 5.15 OHSUMEDDATASET – RECALL, PRECISION, F-1 SCORE ANALYSIS

5.7 CHAPTER SUMMARY

In this chapter, the methodology for implementingall the three models constructed with the inverse exponential linear unit functionis explained. The initial setup and software installation are elucidated. The code organisation of the filesis indicated to comprehend the confluence of dataset, algorithm and

program. The important features such as Architecture design, Parameter learning, Feature Extraction are discussed. Finally, the performance metrics are indicated and the results are analysed through graphical representation. It is inferred that the proposed deep learning based system when combined with the inverse exponential linear unit outperforms the existing classification system.In next chapter, the conclusion and prospective future work will be discussed.

CHAPTER 6
CONCLUSION AND FUTURE WORKS

This thesis presents the extensive analysis of a deep learning based document categorizer system that can be primarily deployed for the purpose of segregating various sets of documents. Literature analysis of various existing classifier algorithms indicate that, inspite of overcoming the traditional word count based classification model by introducing content based categorization, the deep learning based classifiers suffers from certain drawbacks. These include the biased feature interpretation by RNN and the window based feature comprehension by CNN. In order to overcome the limitations of the individual RNN and CNN models, we proposed to combine these modules in parallel manner. We have constructed three different architectures that combines CNN and RNN modules in parallel manner. The analysis of the performance of the proposed architectures over five different types of dataset clearly indicate that all the three variants surpasses the performance of traditional sequential architecture. And among these three variants, the RCPSCNN model is the most effective model. Our analysis also indicates that the newly introduced Inverse exponential linear unit function is better in comparison with the existing activation functions. Hence, It is concluded that the proposed deep learning based system when combined with the inverse exponential linear unit outperforms the existing classification system.The intelligent text classifier can be used in various scenarios such as document indexing, web searching, spam detection, sentimental analysis, tag suggestion and even information filtering.

Based on the results of analysis, it has been identified that the proposed algorithm works well when the convolutional and recurrent neural network are integrated modularly with one another in a parallel manner. The newly defined activation function plays a critical role in enhancing the performance of the system. The proposed system has dealt with only limited parameters. Varieties

of ideologies like ingraining a convolutional layer into a recurrent network or recurrent layers into a convolutional network are yet to be analysed.

Future work in this domain must be aimed in improving the performance of the system by identifying the techniques to implant of one type of neural network within another. The significant portion of presage work must involve designing algorithms that takes lesser processing time. Prospective future work must expand on the inconsistencies that the current system possesses and must ensure that the drawbacks of the existing system are addressed.

REFERENCES

[1] Tao Chen, Ruifeng Xu, Yulan He, Xuan Wang (2017), 'Improving sentiment analysis via sentence type classification using BiLSTM-CRF and CNN', Journal Expert Systems with Applications, Vol.72, pp.221–230.

[2] Yuan Luo(2017),'Recurrent neural networks for classifying relations in clinical notes', Journal of Bio Informatics, Vol.72, pp.85-95.

[3] Mohammadreza Ebrahimi, Ching Y.Suen, Olga Ormandjieva(2016), 'Detecting predatory conversations in social media by deep Convolutional Neural Networks', Digital Investigation, Vol.18, pp.33-49.

[4] TruyenVan Phan, Masaki Nakagawa(2016), 'Combination of global and local contexts for text/non-text classification in heterogeneous online handwritten documents', Pattern Recognition, Vol.51, pp.112-124.

[5] Zhang Pei-ying, LI Cun-he(2009),' Automatic text summarization based on sentences clustering and extraction', Computer Science and Information Technology, ICCSIT 2nd IEEE International Conference, pp.167-170.

[6] Kavita Ganesan, ChengXiang Zhai and Jiawei Han(2010), 'Opinosis Graph for Abstractive Summarization of Redundant Opinions', 23rd International Conference on Computational Linguistics, pp.340-348.

[7] Rafael Ferreira, Frederico Freitas, Luciano de Souza Cabral, Rafael Dueire Lins, Rinaldo Lima, Gabriel Franca, Steven J. Simske, and Luciano Favaro(2014), 'A Context Based Text Summarization System', IAPR International Workshop on Document Analysis Systems, pp.66-70.

[8] Shuai Wang, Xiang Zhao, Bo Li, Bin Ge, Daquan Tang(2017), 'Integrating Extractive and Abstractive Models for Long Text Summarization', IEEE 6th International Congress on Big Data, pp.133-137.

[9] Haiyi Zhang, Di Li(2007), 'Naïve Bayes Text Classifier', IEEE International Conference on Granular Computing, pp.142-147.

[10] Shahnaj Parvin, Md. Delowar, Md. Nadim, Sayed Golam, Tangina Sultana(2016), 'Enhancing Performance of Naive Bayes Classifier by

adding Extra Weights with less number of training examples', International Workshop on Computational Intelligence, pp.142-147.

[11] Srinivasan Ramaswamy(2010), 'Multiclass Text Classification - A Decision Tree based SVM Approach', arXiv, pp.112-124.

[12] Jing zhong Wang and Xia Li(2010), 'KNN algorithm for text classification, International Conference on Information', Networking and Automation (ICINA), Vol.2, pp.436-439.

[13] Yun Lin, Jie Wang(2014), 'Research on Text Classification Based on SVM-KNN, Software Engineering and Service Science (ICSESS)', 5th IEEE International Conference, pp.842-844.

[14] Siwei Lai, Liheng Xu, Kang Liu, Jun Zhao(2015), 'Recurrent Convolutional Neural Networks for Text Classification', Twenty-Ninth AAAI Conference on Artificial Intelligence, pp.2267-2273

[15] Yujun Zhou, Bo Xu, Jiaming Xu, Lei Yang, Chang liang Li, Bo Xu(2016), 'Compositional Recurrent Neural Networks for Chinese Short Text Classification', IEEE/WIC/ACM International Conference on Web Intelligence, pp.137-144.

[16] Julio Reyes, Diego Ramırez, Julio Paciello(2016), 'Automatic Classification of Source Code Archives by Programming Language: A Deep Learning Approach', International Conference on Computational Science and Computational Intelligence, pp.514-519.

[17] Ronan Collobert, Jason Weston†, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, Pavel Kuksa(2011), 'Natural Language Processing (Almost) from Scratch', Journal, pp.2498-2510.

[18] Meng Joo Er, Yong Zhang, Ning Wang, Mahardhika Pratama(2016), 'Attention pooling-based convolutional neural network for sentence modeling', Information Sciences, Vol.373, pp.388-403.

[19] Y. Bengio , R. Ducharme , P. Vincent , C. Janvin(2003) , 'A neural probabilistic language model', J. Mach. Learn. Res., Vol. 3, pp.1137–1155.

[20] Y.-L. Boureau , F. Bach , Y. LeCun, J. Ponce (2010), 'Learning mid-level features for recognition', 2010 IEEE Converences on Computer Vision and Pattern Recognition (CVPR), IEEE, pp. 2559–2566 .

[21] Y.-L. Boureau , N.L. Roux , F. Bach , J. Ponce(2011) , Y. LeCun ,'Ask the locals: multi-way local pooling for image recognition', IEEE International Conference on Computer Vision (ICCV), IEEE, pp.2651–2658 .

[22] R. Collobert , J. Weston(2008) , 'A unified architecture for natural language processing: Deep neural networks with multitask learning', Proceedings of the 25th International Conference on Machine learning (ICML), ACM, pp.160–167 .

[23] G. Frege (1948),'Sense and reference', The Philos. Rev. 57(3) pp.209–230.

[24] K.-i. Funahashi , Y. Nakamura (1993),'Approximation of dynamical systems by continuous time recurrent neural networks', Neural Netw. Vol.6, pp.801–806 .

[25] A. Graves , J. Schmidhuber (2005), 'Framewise phoneme classification with bidirectional LSTM and other neural network architectures', Neural Netw., Vol.18, pp.602–610 .

[26] S. Hochreiter , J. Schmidhuber (1997),'Long short-term memory' , Neural Comput, Vol.9, pp.1735–1780 .

[27] M. Hu , B. Liu(2004), 'Mining and summarizing customer reviews', Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), ACM, pp.168–177 .

[28] E.H. Huang , R. Socher , C.D. Manning , A.Y. Ng (2012), 'Improving word representations via global context and multiple word prototypes', Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Vol.1 (ACL), Association for Computational Linguistics, pp.873–882 .

[29] R. Johnson , T. Zhang(2015) ,'Effective use of word order for text categorization with convolutional neural networks', Proceedings of the

2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL), pp.103–112 .

[30] N. Kalchbrenner , E. Grefenstette , P. Blunsom(2014) ,'A convolutional neural network for modelling sentences', Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL), pp.655–665 .

[31] Y. Kim(2014) ,'Convolutional neural networks for sentence classification', Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp.1746–1751 .

[32] Q.V. Le , T. Mikolov(2014) ,'Distributed representations of sentences and documents, Proceedings of the 31th International Conference on Machine learning (ICML)', pp.1188–1196 .

[33] Y. LeCun , Y. Bengio , G. Hinton(2015) ,'Deep learning, Nature', Vol.521, pp.436–444.

[34] Y. LeCun , L. Bottou , Y. Bengio , P. Haffner(1998) ,'Gradient-based learning applied to document recognition', Proc. IEEE, Vol.86, pp.2278–2324.

[35] X. Li , D. Roth (2002),'Learning question classifiers', Proceedings of the 19th international conference on Computational linguistics (ACL), Association for Computational Linguistics, Vol.1, pp.1–7 .

[36] L. Van der Maaten , G. Hinton , (2008),'Visualizing data using t-SNE', J. Mach. Learn. Res., Vol.9, pp.2579–2605 .

[37] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean (2013),'Distributed representations of words and phrases and their compositionality' , Advances in Neural Information Processing Systems (NIPS), pp.3111–3119 .

[38] B. Pang , L. Lee(2004), 'A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts', Proceedings of the

42nd Annual Meeting on Association for Computational Linguistics (ACL), Association for Computational Linguistics, pp.271–278 .

[39] Baroni. M, Dinu. G, and Kruszewski. G (2014),'Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors', In ACL, pp.238–247.

[40] Bengio. Y, Courville. A and Vincent, P(2013),'Representation learning: review and new perspectives', IEEE TPAMI, Vol.35, pp.1798–1828.

[41] Dharmendra Hingu, Deep Shah, Sandeep S. Udmale(2015),' Automatic Text Summarization of Wikipedia Articles', International Conference on Communication, Information & Computing Technology (ICCICT), pp.1-4.

[42] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, Eduard Hovy(2016), 'Hierarchical Attention Networks for Document Classification', arXiv: 1702.01829.

[43] Mark J. Berger(2017), 'Large Scale Multi-label Text Classification with Semantic Word Vectors', Stanford paper pp: 1-8.

[44] Kamran Kowsari, Donald E. Brown, Mojtaba Heidarysafa, Kiana Jafari Meimandi, Matthew S. Gerber, and Laura E. Barnes(2017), 'HDLTex: Hierarchical Deep Learning for Text Classification', arXiv: 1709.08267

[45] Xin Rong(2017), 'Word2vec Parameter Learning Explained', arXiv: 1411.2738.

[46] Ilya Sutskever, Oriol Vinyals, Quoc V. Le(2016), 'Sequence to Sequence Learning with Neural Networks', arXiv:1409.3215.

[47] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean(2013), 'Efficient Estimation of Word Representations in Vector Space', arXiv:1301.3781.

[48] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean(2013), 'Distributed Representations of Words and Phrases and their Compositionality', arXiv:1310.4546.

[49] J. Ba , V. Mnih , K. Kavukcuoglu(2015),'Multiple object recognition with visual attention',Proceedings of 3rd International Conference on Learning

Representations (ICLR), arXiv:1412.7755.

[50] D. Bahdanau , K. Cho , Y. Bengio(2015), 'Neural machine translation by jointly learning to align and translate', Proceedings of 3rd International Conference on Learning Representations (ICLR), arXiv:1409.0473.

[51] A .L. Maas, A .Y. Hannun, A .Y. Ng (2013),'Rectifier nonlinearities improve neural network acoustic models', Proceedins of The 30th International Conference on Machine Learning Workshop on Deep Learning for Audio, Speech, and Language Processing (ICML), Stanford, pp.1-6.

[52] T. Mikolov , K. Chen , G. Corrado , J. Dean(2013),'Efficient estimation of word representations in vector space', Proceedings of Workshop at First International Conference on Learning Representations (ICLR), arXiv:1301.3781.

[53] Architecture of Deep Neural Network, http://neuralnetworksanddeeplearning.com/chap6.html.

[54] Layers of Convolutional Neural Network, http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

[55] Maxpooling, https://en.wikipedia.org/wiki/File:Max_pooling.png

[56] Recurrent Neural Network, http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

[57] Long Short Term Memory, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[58] Bidirectional RNN, http://blog.jacobandreas.net/monference.html

[59] Attention model, http://opennmt.net/OpenNMT/training/models/

ANNEXE I – INSTALLATION STEPS

TO INSTALL NVIDIA DRIVER:

1. Add graphics driver PPA

```
sudo add-apt-repository ppa:graphics-drivers/ppa
```

2. Update repositories

```
sudo apt update
```

3. Launch Additional Drivers and select NVIDIA driver 375.26 and then restart

4.To check NVIDIA driver version: nvidia-smi

Troubleshoot:

For some cards, there may be a black screen issue after installing NVIDIA proprietary GPU drivers. Try either set NOMODESET in grub option, or remove the proprietary driver via command:

```
sudo apt-get purge nvidia-current
```

TO INSTALL CUDA:

1. Downloaded latest stable CUDA from developer.nvidia.com/cuda-downloads

2.Installation Instructions:

```
sudo dpkg -i cuda-repo-ubuntu1604-8-0-local_8.0.44-1_amd64.deb
sudo apt-get update
sudo apt-get install cuda-8.0
```

3.CUDA installation changed NVIDIA driver from 375.26 to 367.57

4.Set the path variable and LD_LIBRARY_PATH by referring CUDA Quick Start Guide.pdf or CUDA Installation Guide Linux – Post Installation Actions

```
export PATH=/usr/local/cuda-8.0/bin${PATH:+:${PATH}}
export LD_LIBRARY_PATH=
/usr/local/cuda-8.0/lib64${LD_LIBRARY_PATH+${LD_LIBRARY_PATH}}
```

5. Successfully Installed CUDA-8.0

6. To check CUDA version: `nvcc -version`

```
cat /usr/local/cuda/version.txt
```

7. To check CUDNN version:

```
cat /usr/local/cuda/include/cudnn.h | grep CUDNN_MAJOR -A
```

TO INSTALL CUDNN:

In the following sections:

• CUDA directory path is referred to as /usr/local/cuda/

• CUDNN download path is referred to as <cudnnpath>

INSTALLING FROM A TAR FILE

1. Navigate to your <cudnnpath> directory containing the cuDNN Tar file.
2. Unzip the CUDNN package.

   ```
   $ tar -xzvf cudnn-8.0-linux-x64-v6.0.tgz
   ```

3. Copy the following files into the CUDA Toolkit directory.

   ```
   $ sudo cp cuda/include/cudnn.h /usr/local/cuda/include
   $ sudo cp cuda/lib64/libcudnn* /usr/local/cuda/lib64
   $ sudo chmod a+r /usr/local/cuda/include/cudnn.h
   /usr/local/cuda/lib64/libcudnn*
   ```

INSTALLING TENSORFLOW ON UBUNTU

1. Install TensorFlow from sources as documented in Installing TensorFlow from Sources.
2. Install or upgrade to at least the following NVIDIA versions:
3. CUDA toolkit 7.0 or greater
4. CUDNN v3 or greater
5. GPU card with CUDA Compute Capability 3.0 or higher.

Installing with Virtualenv

Take the following steps to install TensorFlow with Virtualenv:

1.Install pip and Virtualenv by issuing one of the following commands:

```
$ sudo apt-get install python3-pip python3-dev python-virtualenv # for Python 3
```

2.Create a Virtualenv environment by issuing one of the following commands:

```
$ virtualenv --system-site-packages -p python3 # for Python 3
```

3.Activate the Virtualenv environment by one of the following commands:

```
$ source ~/tensorflow/bin/activate
```

4.Ensure pip ≥8.1 is installed:

```
(tf)$ easy_install -U pip
```

5. Issue one of the following commands to install TensorFlow in the active Virtualenv environment:

```
 (tf)$ pip3 install --upgrade tensorflow-gpu==1.4
```

Keras Installation

```
$ pip3 install keras==2.1.2
```

Numpy installation

```
$pip3 install numpy
```

Matplotlib installation

```
$pip3 install matplotlib
```

Scikitlearn installation

```
$pip3 install scikit-learn
```

GraphViz installation

```
$ sudo apt install python-pydot python-pydot-ng graphviz
```