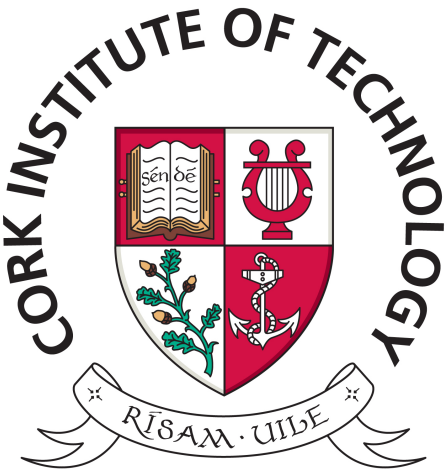


Sentiment Threshold Apprehension using CNN Auto Encoders for early detection of cyber attacks in social media

by

Sonu Prasad

This thesis has been submitted in partial fulfillment for the degree of Master of Science in Artificial Intelligence

in the
Faculty of Engineering and Science
Department of Computer Science

May 2019

Declaration of Authorship

I, Sonu Prasad , declare that this thesis titled, Sentiment Threshold Apprehension using CNN Auto Encoders for early detection of attacks in social media and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a masters degree at Cork Institute of Technology.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at Cork Institiute of Technology or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this project report is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.
- I understand that my project documentation may be stored in the library at CIT, and may be referenced by others in the future.

Signed: _____

Date: _____

CORK INSTITUTE OF TECHNOLOGY

Abstract

Faculty of Engineering and Science
Department of Computer Science

Master of Science

by Sonu Prasad

Social media, forums, blogs are important tool to convey information to large audience and has bought economic and social growth in the society, but has left many organization, government, intuitions or societies vulnerable to cyber-attacks. Businesses takes time to react to malicious activity and even with a proactive security posture and security measure in place, a breach can still happen. Lately there has been some improvements in cyber security for prevention, but it lacks early detection. We define the use of Social media, forums, blogs as a tool to gain insights, statistics and raw data. Here, we try to identify early cyber-attacks but taking into consideration the raw data from social media posts.

In this paper, we focus on sequence to sequence, CNN based neural network for sentiment analysis with transfer learning. We define CNN based auto encoders with Sigmoid activation and a loss function on the auto encoder. To reduce the bias bought by auto encoders and to avoid over-fitting of data, we will be defining posterior probability on the weights of auto encoders. The social media posts which will them be empirically compared to CNN baseline, to show that our methodology is better in solving sentiment related tasks. The proposed approach is simple, straightforward and can be used as for multiple domain. The Project is made in collaboration with `Johnson Controls International plc`, a multinational conglomerate headquartered in Cork, Ireland.

Acknowledgements

I would like to thank my thesis advisor **Dr. Haithem Afli** who gave me the chance to work on this project. He was always involved in the project with me and readily available for help whenever I ran into some trouble.

I would like to thank **Johnson Controls**, for giving me the opportunity to work with them and collaborating with this project.

I would also like to acknowledge **Cork Institute of Technology** for providing me with the opportunity for working on this project and writing the thesis.

I would like to express my profound gratitude to my Irish Granny (**Margaret**) and my parents (**Krishna** and **Mamta**) for providing me with unfailing support and continuous encouragement throughout the thesis.

Contents

List of Figures

List of Tables

Abbreviations

NN	**N**eural **N**etworks
ANN	**A**rtificial **N**eural **N**etwork
CNN	**C**onvolutional **N**eural **N**etwork
RNN	**R**ecurrent **N**eural **N**etwork
ReLU	**Re**ctified **L**inear **U**nit
MLP	**M**ulti **L**ayered **U**erceptron
MSE	**M**ean **S**quared **E**rror
GAP	**G**lobal **A**verage **P**ooling

Chapter 1

Introduction

The world is getting hyper connected due to the presence of social media where people socialize daily about lots of stuff which is one of the factor in boosting the economy and improving social coordination among people. But theres also some downside to it; i.e. it has left people and organizations open to vulnerabilities where hackers and cyber attackers can steal, misuse and manipulate the information as per their need.

Every year diversified cyber-attacks are on a continuous rise. Mobile Malwares are increasing, IP and Identity theft impacting hundreds of thousands of people worldwide and many more types of cybersecurity landscape which all leads to creation of bots. These bots, since they are substantial in number, have the power to easily bring down a powerful network where organizations, institutions keep their import data. And thats not all; most of the cyber-attacks go undetected unless its meant to keep a network down for long time; For e.g. attacks like UDP Flood, SYN Flood, NTP Amplification, HTTP Flood which are all a part of DDOS attack. Recently we have witnessed a large number of Cyber-attacks on organizations like GitHub, Ashley Madison, Sony, U.S government, Indian Aadhar card system and lots more. And not to mention the different types of ransomwares like WannaCry, Not Petya, BadRabbit, GandCrab, GoldenEye and the list goes on where some of the ransomwares are very recent and all these ransomwares were very successful in infecting organizations and people and the only way victims can regain access to their files is by paying a ransom to the cyber criminals. Many of these hacks and attacks have one thing in common and that is they lack early detection. So,

in order to alleviate the risk from cyber-attacks and add a strong preventive measure to reduce the damage, we need new approach to detect it in initial stages.

To manage an extensive cyber-attack, the hacker must 1) Identify the target and their vulnerability 2) Gain access to required tools and expertise 3) Recruit members (usually on dark web and professional forums) 4) Program and execute the attack. Other computer professionals such as software developers, security geeks, network administrators and even victims may discuss computer vulnerabilities, or even coordinate defenses against threats online. These discussions generally develop on online forums, social media like twitter, dark web forums and hence addressing upcoming potential attacks or vulnerability that will be exploited or some organization might be attacked [2]. These digital footprints provide worthwhile insights when a cyber threat is in the early stages of development and well before the malicious activity is about to happen. For example, there are lot of instances when the exploit is disclosed publicly or on dark web forums even before the public release of the vulnerability.

In order to overcome late detection of cyber attacks and have it detected even before its release, we introduce a machine learning method where we leverage the power of social media like twitter to detect sentiments and process them effectively. Traditional sentiment analysis takes into account bag of words model by mapping text into high dimensional numerical feature vector, which is not very accurate as it fails to capture complex linguistic senses. For instance, consider the two phrases which has same bag of words, cheap product market is destroying the local economy and local economy is destroying cheap product market; the former is a Negative sentence and the latter is a positive sentence. Also, a lot of previous work is based on positive / negative sentiments as the system relates to high dimensionality of extracted features vectors making the model noisy and with high variance generally because of over fitting on training data and hence does not reflect complex human emotions and sentiments. Here, we try to address the following issues: 1) Exploit hierarchical structure of the data instead of using bag of words model. 2) Use of DNN's to train the network [3].

Deep Neural Networks (DNNs) are fundamentally inspired by human brain to simulate their activity. They usually have an input layer, an output layer and one or more hidden layers in between. They have the power to ingest huge amount of data without having

prior knowledge of the domain, performs different types of learning tasks and optimize the loss and parameters using optimizers.

Many different approaches have been discovered to tackle this kind of difficult problem. The focus on our thesis is using Deep Convolutional Neural Networks which is a part of DNN as well as transfer learning on Convolutional Neural Networks (CNN) for text polarity classification, and experimentally compare variety of architectures. CNNs have already shown great result in the field of image classification [4] and they are also exceeding the field on sentiment analysis [5].

Because of the fundamental nature of CNNs to fit widely on non-linear data points, they tend to be ambitious on data and so, requires large amount of data for training purposes to function effectively. Over fitting is a huge problem in these kinds of CNNs and neural networks in general. The model tends to fit well for the training data but are not predictive for train and validation data. So, the neural network usually overlooks the hidden uncertainty in the training data if its over fitted and hence the model accuracy drops as it lacks performance of making decisions about the correct class on new data.

In order to overcome this situation, we show that CNN Auto Encoders along with transfer learning can be used to control the over fitting, learn the sentence embedding and classification ability.

The **Evaluation metric** used is F1 score along with Accuracy and Loss of Training and Validation data [6]. F1 score is the weighted average of precision and recall which is given by:

$$F1 = 2 * (precision * recall)/(precision + recall) \tag{1.1}$$

Precision: Precision is proportion of correct positive classification and is given by equation $Precision = TruePositive/(TruePositive + FalsePositive)$. **Recall**: Recall is the proportion of actual positives that was correctly predicted and is given by the equation: $Recall = TruePositive/(TruePositive + FalseNegative)$

1.1 Motivation

Recently there has been a growing number of cyber attacks and most of the attacks are discussed or mentioned on social media platforms like twitter. Neural Networks are able

to achieve state of the art results in most of the classification tasks but despite that, there can be over fitting and misleading classification on new data if its not trained on huge amount of data. Correct Neural Net (NN) architecture, weights and abundant data are required to make the model classify the small phrases related to cyber-attacks that appears on twitter.

We address these concerns by introducing CNN based auto encoders to learn the embedding vector and classification ability at the same time and then transfer the learning so that it can be trained on domain specific task.

1.2 Contribution

The main contribution of work are as follows:

1. We present how Auto Encoders can be effectively applied to CNN. We therefore, introduce the idea of Encoder and a decoder model comprised of CNN layers.
2. We show how model learns richer representation and classification ability from loss weight distribution.
3. We introduce transfer learning to be adapted for domain specific task trained on Auto Encoder based CNN.
4. We empirically show the behavior of the model with/without transfer learning for sentiment threshold apprehension.

1.3 Structure of This Document

The thesis is structured as follows: Starting from next chapter, i.e. Chapter 2, is about the Background and Literature review of Neural Networks and its architecture. Chapter 3 contains Design of our proposed Architecture. In Chapter 4, we provide our Architecture Implementation steps, Data set used and different configuration's that we aim to test. Chapter 5 is the Evaluation of different test methods that we discussed in Chapter 4. We also discuss about Threshold Apprehension in this chapter. We conclude the discussion in Chapter 6.

Chapter 2

Background

Sentiment analysis is a part of Natural Language processing also known as Opinion mining is used to study peoples opinion, reviews, sentiments, attitude towards any particular product, service, organization, stock market, individual, events [7] [8]. The rise of social media giants like Twitter, Facebook, Reddit has given rise to unstructured textual data on internet which often carries huge volume of user expressions and opinions. Internet today is user centric which means users are now co-producers for the content on the web. So, if a user wants to buy a product, then the user is not just confined to the friend inner circle but has access to massive internet which hosts user reviews, pros and cons and discussion board for the product in public forums, blogs and social media pages.

Nowadays, if one wants to buy a consumer product, one is no longer limited to asking ones friends and family for opinions because there are many user reviews and discussions about the product in public forums on the Web. Around 2.7 Billion people out of 3.2 Billion Internet Users are active on social media [9] [10] which means 86% of the users are using social media. These active social media users contribution ranges from blog posts, images, videos, tweets, articles etc of which huge volume of data goes to unstructured text.

Reviews / Opinions expressed in social media about any product or some other context constitutes an important and interesting part which is worth analysis as it has the potential to reach thousands of people and can have a wide impact on the product as well as on the society. With increase in the amount of informative resources such as movie reviews, product reviews, blog posts, tweets, the next demanding task is the need of

an algorithm than can process large volume of mined data and output the opinion of others related to the context. This information derived from the user sentiments is of immense importance to the companies as well as the users in taking informed decisions [11]. However monitoring multiple sites and extracting data from large amount of unstructured data (twitter for instance) is a gigantic task and hence automated sentiment analysis is thus needed.

Sentiment analysis is the extraction of sentiment from an unstructured data like text, images or audio and can be classified as fine grained sentiment or positive / negative. Sentiment analysis has recently taken up the development pace and received a considerable interest from the academic and developer community [12]. Initial research had various techniques for sentiment analysis in supervised and unsupervised manner. Machine Learning techniques like Nave Bayes, Support Vector Machines, Maximum Entropy are used [13] whereas for unsupervised learning focus is on text exploitation (like POS Tagger), grammatical analysis, semantic analysis [14] [15].

Recently, deep learning approaches has emerged to be very promising [16] and has derived state of the art conclusion in domains ranging from Image Processing to NLP. Deep Learning tries to exploit intermediate representations in a hierarchical manner [17] and has become very famous recently in NLP (sentiment analysis) domain.

2.1 Recurrent Neural Network

Recurrent Neural Network (RNN) is the type of network whose output depends on its previous input.

RNN allows us to use arbitrary size of data and not limited to the size of the context instead of using Feed forward network that has fixed length content and needs to be defined explicitly before training [18]. RNN takes two input at every time step. Data i.e. input layer x and hidden layer s and it outputs a result y at every time step t. Initially small values are assigned to weights with Gradient Descent as the optimizer for standard back propagation [18].

Figure 2.1 shows unfolded RNN network with 3-time steps on the right while the left is a folded RNN sequence network with multiple cycles. Number of time steps depends

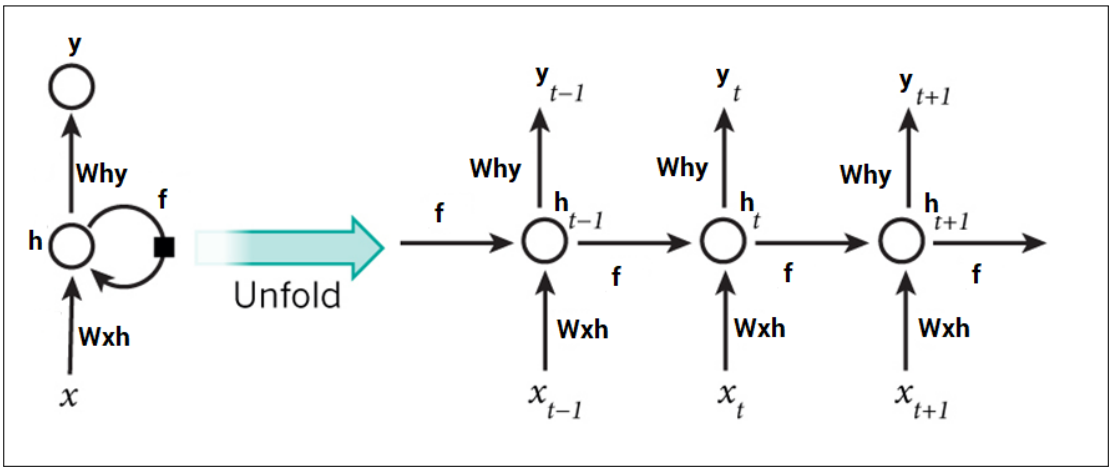


FIGURE 2.1: Recurrent Neural Network

on the length of the input. These RNN networks are quite powerful and they have the tendency to over-fit the data i.e. they might work well on training data but will give poor results on unseen data. To avoid these circumstances, Dropouts are used. Dropout is a regularization technique where random network units are dropped. This helps in increasing the robustness of the model, but it shouldn'tt be used between hidden layers as these layers are repeated [19].

RNN has gradient exploding and vanishing problem and is unable to retain much information of inputs provided few time steps before [18]. If the gradient is small, then multiplying it multiple times results in very small gradient (almost 0) whereas if the gradient is large then it increases drastically. These shortcomings of RNN are overcome by different sophisticated versions of RNN like Bidirectional RNN, Long Short-Term Memory (LSTM).

LSTM is a gradient-based learning algorithm which can learn to bridge time intervals even in case of noisy, in-compressible input sequences, without loss of short time lag capabilities. This is achieved by gradient-based algorithm which enforces a constant error flow through internal states of special LSTM units [20].

2.2 Attention Based Neural Networks

RNN and RNN versions like LSTM are generally used for sequential processing of data and hence has to deal with long range dependencies. Long range dependencies are

difficult and problematic to handle. Thus, a technique called Attention Mechanism was proposed.

Attention as a concept has given a lot of success in sequence modelling tasks, where input/output both are a sequence (like translation). Attention Mechanism can be viewed as a method for making the RNN work better by letting the network know where to look as it is performing its task by selectively focusing on parts of the source sentence during translation [21]. Attention mechanism purpose is to orient perception as well as memory access. It filters the data (perceptions) that can be stored in the memory and filters them again when they need to be retrieved from the memory.

Attention Mechanism is mainly used for language translation in NLP. It consists of encoder and a decoder where an attention mechanism is used for selecting reference words in original language (source sentence) of words in target language before translation. This makes it easier for the encoder to process and encode only few sentences instead of having to encode all information [1].

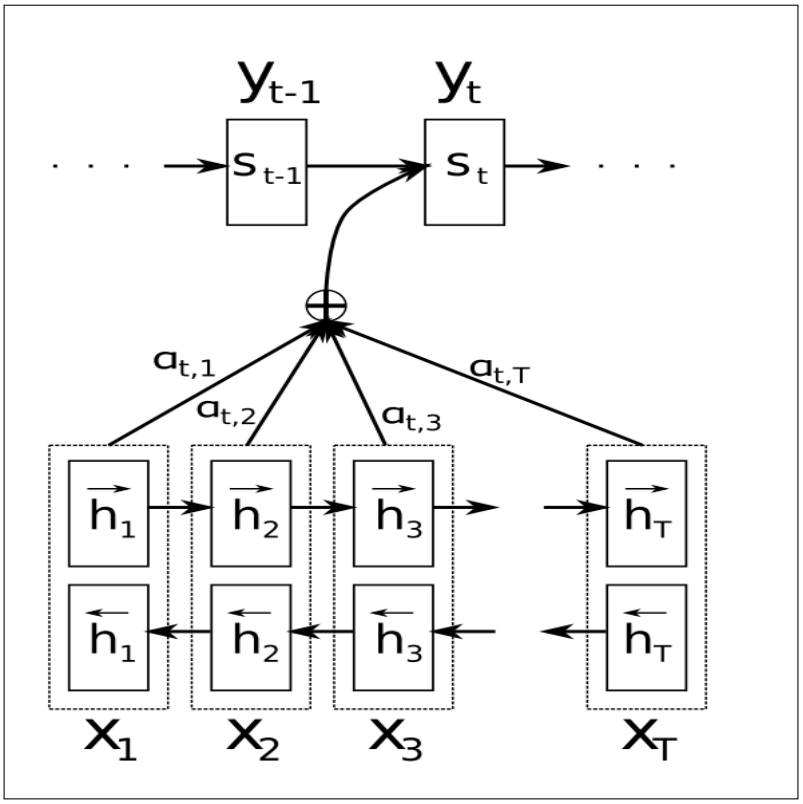


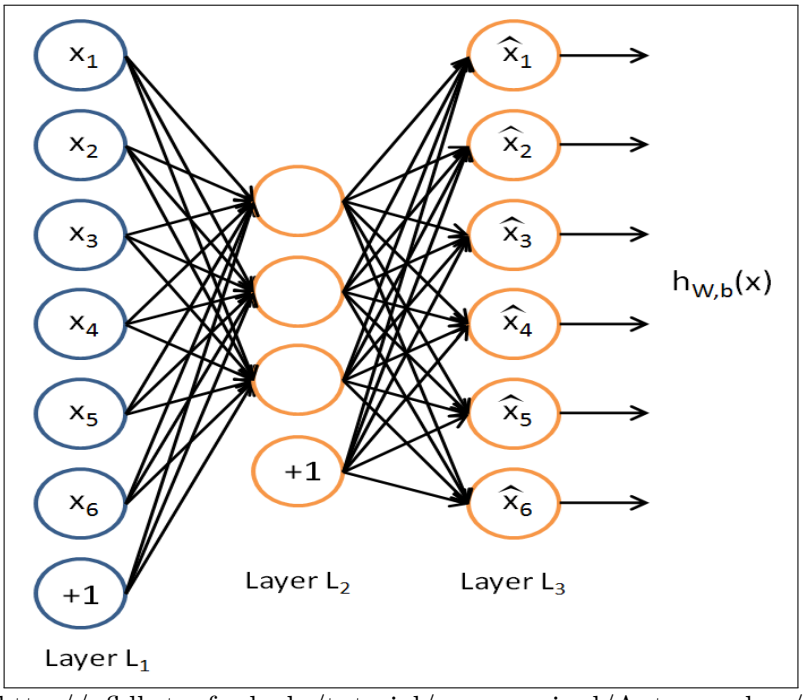
FIGURE 2.2: Attention Model to generate target y_t using input $X_1, X_2, ...X_T$ [1]

Figure 2.2 illustrates the use of attention mechanism in bidirectional RNN. The encoder generates $h_1, h_2, ...h_T$ from inputs $X_1, X_2, ...X_T$. And the decoder output word y_T not

only depends on last state but it depends on weighted combination of all input states. $(a_t, 1), (a_t, 2), (a_t, T)$ are the weights assigned to each input state that affects the output. So, if $(a_1, 1)$ has a high value, then the role played by $(a_1, 1)$ will be substantial and will attract decoder attention which will ultimately affect the second word of the target sentence.

2.3 Auto Encoders

Auto encoders are a type of neural networks which tries to map its input to its output with a hidden layer as a mediator which is responsible for processing the data. Auto encoders tries to identify approximation function so that its output $\overline{x}$ is similar to input x. They usually consist of three layers; Encoder, Decoder and a hidden layer. Figure 2.3 shows an example of Auto encoder.

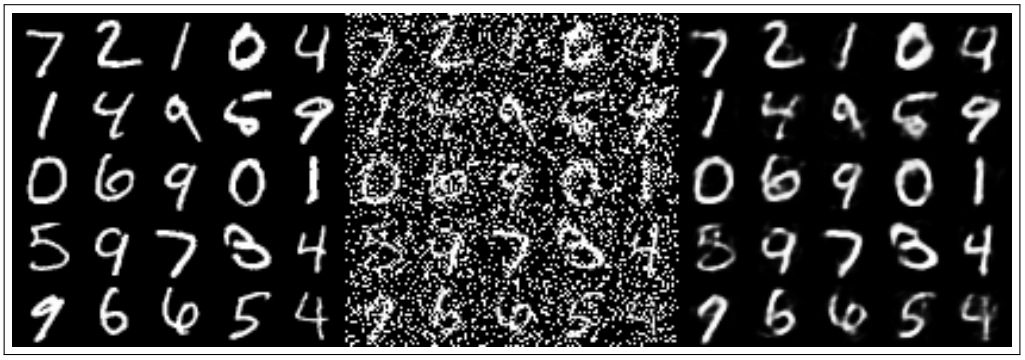


http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/

FIGURE 2.3: Auto Encoder

Given the input data which can be represented into vector space as x $\in R_d$, is first mapped to the latent representation h $\in \overline{R_d}$ using a deterministic activation function (e.g Sigmoid function) of the type h = fθ(h) = $\sigma(W_h + b)$. The decoder is used to reconstruct data from latent representation by reverse mapping of fθ: y = fθ(h) = $\sigma(W_h + b)$ [22]. Since auto encoders works by introducing a hidden layer smaller than the input layer,

it forces the data to be represented in compressed format, the general objective here is to minimize reconstruction error $h_{(W,b)}(x) \approx$ x and thus minimize MSE (Mean square difference) L(x, y) = $\Sigma(x - h_{(W,b)}(x))^2$ where x is the input, y is the output and h is the activation function.



http://www.opendeep.org/v0.0.5/docs/tutorial-your-first-model

FIGURE 2.4: Original input, corrupted data and reconstructed data

Denoising Auto encoder (DAE) is a type of auto encoder in which output is obtained from corrupted input data. The DAE with corrupted data, is forced to denoise the data and learn only the robust features instead of the complete data. The whole purpose of DAE is to make the model more robust, minimize denoising reconstruction error loss and maintain the integrity of the data by reconstructing the data from noisy input vector data. Figure 2.4 shows a DAE where source input data is corrupted by adding some noise and is feeded to DAE where the model tries to reconstruct the input data.

Stacked denoising auto encoders is further enhance of denoising auto encoders where multiple hidden layers are stacked. The training is performed layer by layer and the purpose of each layer is to minimize the reconstruction error from its previous layer (which is treated as input).

2.4 Recursive Neural Networks

Recursive Neural networks are type of recurrent neural networks that tries to exploit data to learn directed acyclic graph structure (Tree Structure).

Given an input word vector to Recursive neural network, it recursively tries to generate parent representation starting from the bottom, combining multiple tokens to produce

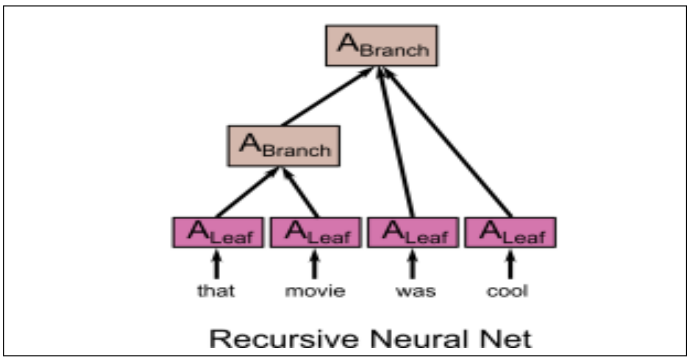


FIGURE 2.5: Recursive Neural Network

generalized representations for phrases and ultimately the complete sentence. An example of Recursive Neural network is shown in the image where different vector nodes are combined to form the parent node. Recursive Neural networks are highly useful for parsing natural scenes and language [23].

2.5 Convolutional Neural Networks

CNNs initially designed for the task of image recognition [24], has now become a very powerful architecture in the field of Sentiment Analysis. CNNs are capable to extract and learn multiple features from a multi-dimensional array such as image or sentence. For example, in the case of facial image recognition, CNN is able to recognize face and its features to accurately identify a face no matter where and how it is placed.

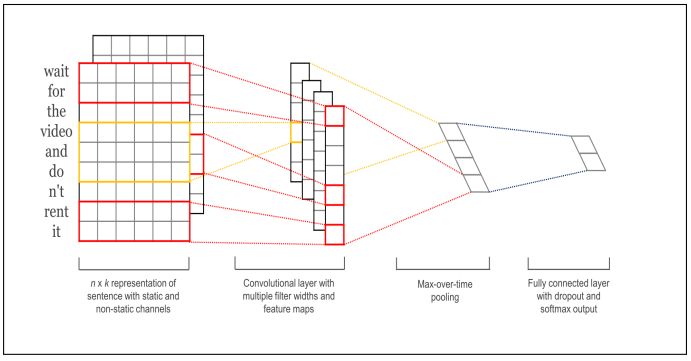


FIGURE 2.6: Recursive Neural Network [5]

The above figure represents a basic architecture of CNN. It consists of Input Layer, Convolutional Layer, Max Pooling Layer, Fully Connected Layers. The input layer is fed to the Convolutional layer which is comprised of multiple filters. These filters which

contains different operations (like strides, filter size, initializer) are applied sequentially to the data which helps the CNN to learn multiple features. The output of the feature map is usually reduced by dimension using Pooling Operation and then finally fed into a fully connected later [5]. The idea behind using CNN for sentiment analysis is on the fact that the text is structured, and CNN tends to work well when there are patterns and structures involved; otherwise it can be lost if using a feed-forward neural network.

2.6 Sentiment Analysis

This section introduces the application of the above concepts in sentiment analysis tasks.

Sentiment analysis is the process to infer users sentiment based on their language expressions [25]. Sentiment analysis has been further divided into three parts namely: Document level sentiment classification, Sentence level sentiment classification and entity aspect-based sentiment classification [26].

Document level sentiment classification aims to derive sentiment as positive or negative from single topic document. Basically, it assumes that your document is opinionated and is comprised of single entity. For example when someone post a review of a particular product, the document level sentiment classifier system tries to determine the overall sentiment and opinion of the user.

Sentence level sentiment classification aims to derive sentiment from sentences rather than documents. The sentiments can be either of positive, negative or neutral. Sentence level sentiment classification can be further broken down to subjectivity classification and sentiment classification [27]. In subjectivity classification, opinion of the sentence is extracted. The second part i.e. Sentiment classification is used to derive the sentiment of subjectivity classification output.

Aspect Entity based sentiment classification (AESC) aims to exploit multiple entities and its features/aspects from users opinion. It is more fine grained as compared to document or sentence level sentiment classification and is defined as given a document d_i and its aspect as a_j, we intend to examine the sentiment as positive, negative, neutral of each aspect in the provided document [28]. AESC can be further broken down into further subtasks like Aspect extraction, Entity Extraction and Aspect sentiment classification.

For example, love iPhone display but they should really do something about battery, the AESC system would extract iPhone as the entity and display, battery as aspect. Then further the system would classify display of iPhone as positive and negative opinion for iphone battery. In this paper we are going to explore sentence level sentiment analysis.

2.6.1 Sentence level sentiment analysis

- Recursive Auto Encoders.

  Authors: Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, Christopher D. Manning [29]

  Proposed multiple variants of Recursive Neural network to compose vector representation for longer sentence from the vector of its child words or phrases.

  Results: Their algorithm was able to accurately predict distributions over different labels compared to several other baseline models.

- Recursive Neural Tensor network

  Authors: Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng and Christopher Potts [30]

  Proposed Recursive Neural Tensor network (treebank) (RTNT) where word vectors and parse tree are used for phrase representation which is passed to the compute vectors for higher nodes in the tree using the same tensor based composition function.

  Results: Compared to multiple models like Recursive neural network [29], matrix vector socher2013parsing and other baseline models like Nave Bayes, SVM but RTNT achieves the highest score.

- TG-RNN / TE-RNN

  Authors: Qiao Qian, Bo Tian, Minlie Huang, Yang Liu*, Xuan Zhu*, Xiaoyan Zhu [31]

  Proposed multiple variants of Recursive Neural network to compose vector representation for longer sentence from the vector of its child words or phrases.

  Tag guided composition function in recursive neural network (TG-RNN) which allocates a composition function for a sentence according to POS (parts of speech) tag of the sentence.

Tag Embedding in Recursive neural network (TE-RNN) which learns on how to embed vectors for POS tags of words / phrases and combines tag embeddings and word vectors together.

Results: Faster as compared to RNN and RNTN. TE-RNN is comparable to CNN and DCNN. TE-RNTN is better than CNN and DCNN but worst than DRNN.

- Dynamic Convolutional Neural Network (DCNN)

  Authors: Nal Kalchbrenner, Edward Grefenstette, Phil Blunsom [32]

  Proposed Dynamic Convolutional Neural Network (DCNN) adopted for semantic modelling of sentences. The network uses Dynamic k-max pooling as global pooling operation over linear sequences. The network accepts varying length input sentences and induces a feature graph which can capture long-short word relations. Preformed 4 experiments namely; We test the DCNN in four experiments: small scale binary and multi-class sentiment prediction, six-way question classification and Twitter sentiment prediction by distant supervision. The system was able to achieve excellent score in all of the tasks as compared to the state of the art baselines.

Chapter 3

Design

In this section, we present our building block for CNN based auto encoders with weight decay i.e. using weighted loss functions to define the posterior probability distribution and then apply transfer learning to the domain specific task.

3.1 Word Vectorization

The input to our model is a set of word vectors on n dimensions. Word encoding is the process of converting categorical values to numerical values within the sequence. An initial size m is set which acts as the vocabulary size and each word in the data set is quantized using 1-of-m encoding. Then each word in the sequence is converted to a sequence of such m sized vectors with fixed length l0. Any sequence exceeding the length l0 is ignored; Similarly, any sequence less than size l0 are padded. Wards that are not in the vocabulary are initialized as all zero vectors. Any character exceeding length l0 is ignored and any characters that are not in the vocabulary are quantized as all zero vectors.

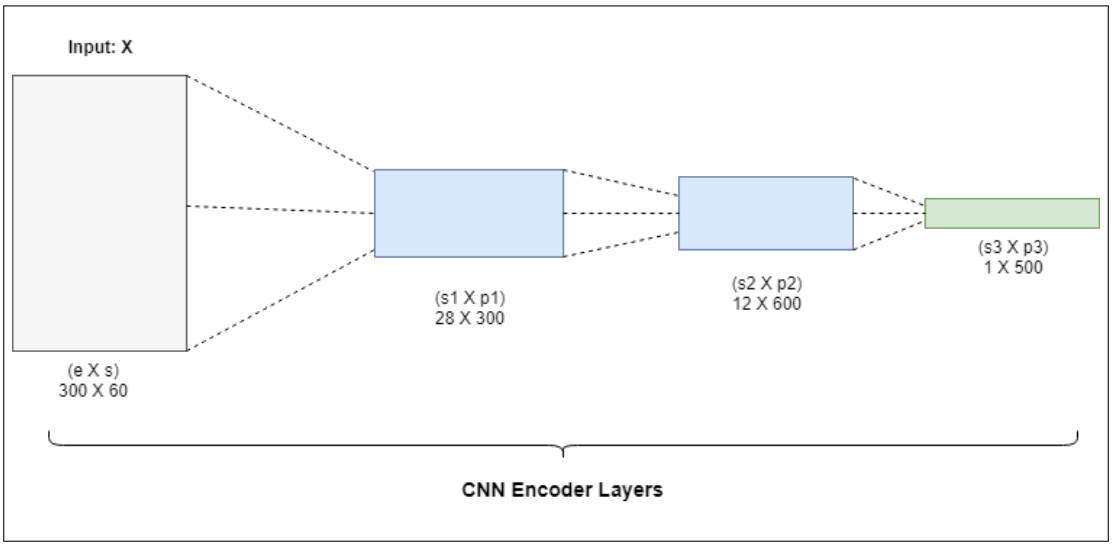


FIGURE 3.1: CNN Encoder Network

3.2 CNN based Encoder Network

In this section, we build encoder network comprising of CNN components as shown in the Figure 3.1. Learned word embedding matrix is represented by

$$W_e = \mathbb{R}^{kXm} \tag{3.1}$$

where k is the embedding dimension, m is the vocabulary size and R represents the array. The input to the network will be comprised of sentences, which is sequence of more than one word, is represented by

$$X \in \mathbb{R}^{kXS} \tag{3.2}$$

where k is the learned word embedding dimension and S is the fixed size sentence length.

The CNN network consists of **L** convolutional layers that can dynamically compress an input sentence into a fixed latent representation vector **h**. Each layer consists of **p** filters **{p1, p2, p3...pi}** which is the number of outputs learned from the input data. Stride in our network is represented by **r** which consists of **{r1, r2, r3...ri}** which when applied to Input **X** applies filter $W_c^{(1,1)} = \mathbb{R}^{kXp_i}$, where p_i is the respective filter size depending on the convolution.

This returns a latent feature map with nonlinear activation function; **Rectified Linear Unit** in our case. The latent feature map is given by equation:

$$c^{(i,1)} = \gamma(X * W_c^{(i,1)} + b^{i,1})$$

[33] (3.3)

where $\gamma(.)$ is non-linear activation function, b is the bias represented by the equation $b^{i,1} \in \mathbb{R}^{(S-h)/r^1+1}$, h is the latent vector representation from CNN layers, r is stride. The changed Embedding dimension after each convolution operation is given by:

$$c^{i,1} \in \mathbb{R}^{(S-h)/r^1+1}[33] \tag{3.4}$$

(i.e. For Layer 1 (1st convolutional layer), i = 1...p1)

The results from **p1** filters are then concatenated to yield a feature map which is **C1** as shown in the image Figure 3.1. The sentence length after every convolutional operation decreases which is given by,

$$S^{l+1} = \left\lfloor ((S^l - f_i)/r_i) + 1 \right\rfloor \tag{3.5}$$

where f is the filter size, r is the stride length and $\lfloor\rfloor$ indicates floor function.

The operations are executed for all the convolutional layers and the final layer is connected to a fully connected dense layer. For the last layer, we use a filter size equal to S^{l-1} which returns $1xp_i$ as the feature map [33].

The idea of having multiple filter is that the lower filters are responsible for capturing basic semantic information like (n-grams) whereas more sophisticated linguistic features are captured by the higher-level filters.

3.3 CNN based Decoder Network

CNN based transpose operations (i.e. Deconvolution) are applied on the CNN encoder network to decode the latent representation obtained after the last step from Encoder Network. As the network expands, the spatial dimension of the compressed input data increases as shown in the Figure 3.2.

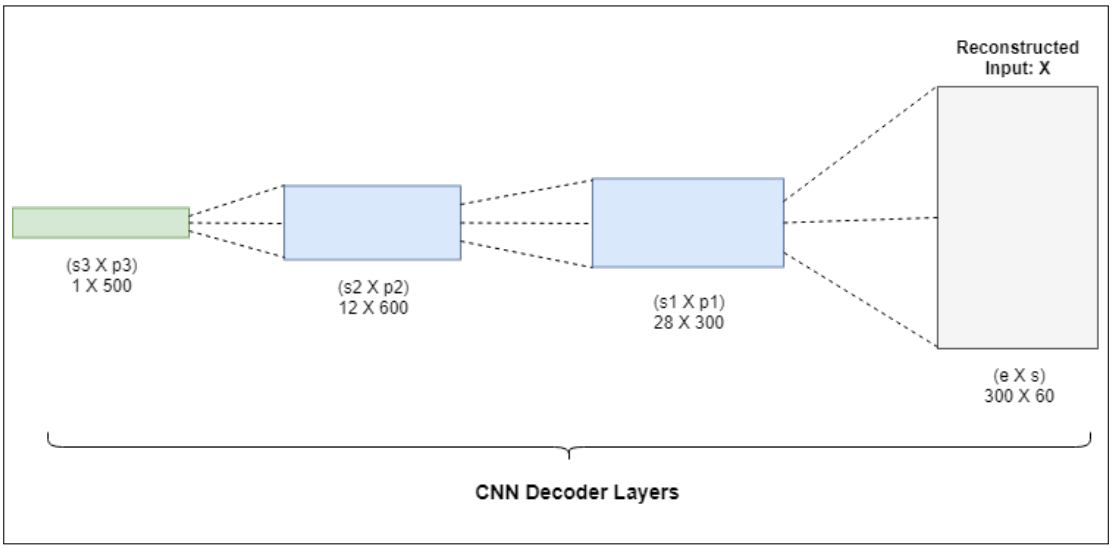


FIGURE 3.2: CNN Decoder Network

The expanded dimension is first matched with the spatial dimension of L - 1 layer of convolution and then subsequently spreads as

$$S^{l+1} = (S^l - 1) * r_i + f \tag{3.6}$$

for l = 1...n convolutional layers, S is the word length, r is the stride length in descending order where i starting from i, i-1, i-2.. i - (i + 1) and f is the filter size in decreasing order similar to strides starting from f, f-1, f - 2, f - (f + 1). The output of this decoder model aims to reconstruct the word embedding matrix. The reason for the presence of the decoder model is because we want our model to learn latent vector representation of different types of sentences.

3.4 Encoder Classifier Output

Encoder and Decoder model are useful for learning the latent representation of the sentences that capture semantic meaning irrespective of the presence of the labels. However, the complete model cannot be trained upon unlabeled data. For classification part, the model needs to perform some supervised training by ingesting some labelled data. Since, the supervised learning process takes the encoded data, the complete step can be modelled as three step process which can be described as:

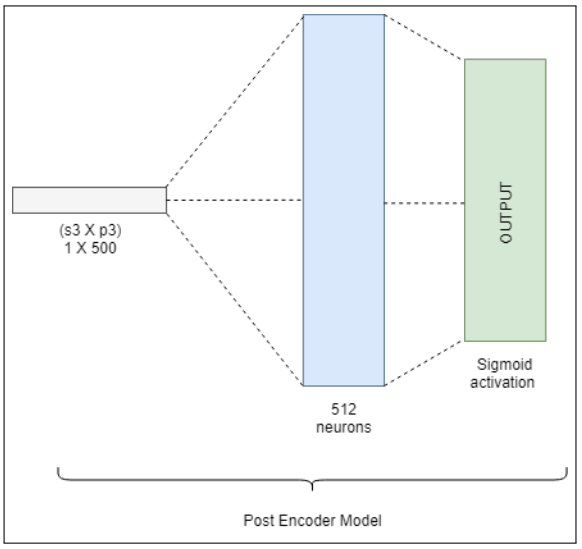


FIGURE 3.3: Encoder Classifier Output

1. Training the encoder model

2. Train the supervised classifier on the above encoder model

3. Train the decoder model

Chapter 4

Implementation Approach

4.1 Data set

Previous research on Sentiment analysis using Convolutional Neural Networks involved large data set especially when the task is related to Natural Language Processing as the model has to learn the semantic meaning of the sentences. However, most of the data set for this project is closed source or it is too small to work on for deep learning models. So, in order to circumvent that situation, we are using `Transfer Learning` which involves two different data sets and two architectures; For initial model prior to transfer learning, we will be using Yelp Polarity Data set and post initial model we will be using twitter annotated data set.

4.1.1 Yelp Data set

The yelp polarity data set is a large data set obtained from Yelp 2015 Data set Challenge comprising of 1,569,264 records. For simplicity purposes, the settings of the data set follows from [34]. The data set is comprised of reviews in terms of stars. All the data having star 3 or 4 are labelled as positive and data having stars 1 or 2 are labelled as negative. The total number of training records are 5,60,000 consisting of positive and negative labelled data while there are 38,000 test records.

4.1.2 Twitter Data set

The main motive in this project is to set the sentiment threshold level for twitter data based on cyber security tweets, i.e. In general, we have to classify, weather a tweet is dangerous or not dangerous with respect to cyber crime. Twitter Data set is a small data set consisting of annotated cyber security labels [35]. There are total of 21,000 records with labels such as [General, Vulnerability, Ransomware, Ddos, Botnet, 0Day, Leak, All]. Since, we are interested in finding only the polarity, we encode the *General* class labels as *negative* while the rest as *positive*. The data set is split into training data and test data in the ratio of 70% and 30% respectively.

4.2 Data set Pre-processing

Yelp data set doesn't need pre-processing. For Yelp data set, words and their sequential count is recorded which is similar to label encoding. The sentences in the Yelp data set are reformulated based on the label encoding and a threshold for max sentence length is set. If the sentence exceeds that specified length, then the sentence is discarded else if the sentence lacks the specified length then padding is introduced in the sentence; where we append a list of zeros to the sentence until it meets the requirement of the specified length. The current sentences are in the form of Bag of words but since we are dealing with Neural Networks, the text needs to be transformed into vector representation. We will be using `GloVe` which is learned word vector representation [36]. The label encoded sentences are then transformed to word embedding matrix which will act as the input to our model; i.e. For each word in one sentence, the words are replaced by their respective embedding matrix from Glove word Embedding and is performed for each work and every sentence. This forms our basis of input data for the first model prior to transfer learning.

Twitter Data is the raw twitter data that needs to be pre-processed before using it.

From the Figure4.1, the following are the steps involved in twitter data pre-processing.

1. **Input**: Input Is raw twitter data which consists of 21000 records. There are 2 columns in the data set; id and text. Id column is discarded as we wont be using

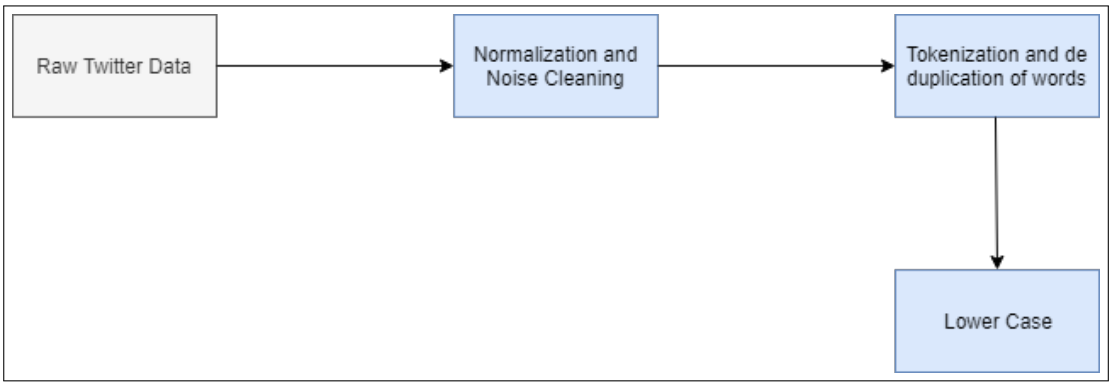


FIGURE 4.1: Twitter Data pre-processing

it. Text column goes through a series of step before being finally used for word vector formation.

2. **Normalization and Noise Cleaning**: This step is responsible for removing all sorts of characters/words that cannot be fed into a neural network model for this project which includes URL & hyperlinks, emojis, special characters and punctuation's apart from [,.!-, 0-9], @ mentions and Retweets.

3. **Tokenization**: Tokenization is an important step as splits the sentences into words. We are using *NLTK* Python package for tweet tokenization [37] which is efficient in handling user tweets. e.g. Sentence: I am sooo happy today !!! Tokenized version: [I, am, sooo, happy, today, !, !, !]

   Deduplication is the process of removing redundant words from the tokenized list. For instance, in the above example, ! mark appears more than once. Hence, only a single instance of ! mark is kept, and others are removed. The deduplication occurs if the words or punctuation's are in continuity. The sentence is deduplicated so that the neural network model should not depend on the presence of multiple entries of certain words /punctuation's to learn its classification.

4. **Lower case**: All the words in the tweet sentences are lower cased.

After the pre-processing of tweets text, a max sentence length is set which denotes the max length of the sentence which it cannot exceed. The sentences having length of more than the specified limit are discarded and the sentences below the specified limit are padded with 0 vectors. The tweet sentences are then applied label encoders where, words are treated as categorical data and converted to numerical data. Those numerical

conversion are applied on the tweet sentences, to convert the tweet sentences into a series of numbers obtained from the above step. `GloVe` Word vectors are used for learned word embedding matrix to convert the label encoded tweets to word vectors.

Due to a limited and narrow quantity of twitter data & the nature of CNNs to be data hungry, the data cannot be directly fed to the initial model for training purposes. So, we will be using this data in conjunction with transfer learning.

4.3 CNN Architecture

The architecture which we are implementing extends the basic idea of having multiple simple CNN models. In our case, since there is a shortage of cyber security data set, we are using Transfer Learning where the initial model is trained on Yelp Data set. There are two architectures which we aim to implement in this paper. The first architecture is a simple CNN based Encoder Decoder network coupled with a single Dense layer consisting of n Neurons and an output layer at the end. The second architecture is transfer learning which is ideal to the first architecture in terms of core components. We aim to take the model weights as well as the architecture for transfer learning and join it with few Multi Layered Perceptron (MLP) at the end in order to increase the model learning on the new data set. Let go through each architecture step by step. This part focuses on first part i.e. Initial CNN model architecture which is described in figure Figure4.2.

1. **Input**: Since, this is the initial model whose weights would be transferred to the later learning part, we need huge data set in order to train this. So, we will be going with Yelp data set to train this part. The pre-processing step follows from the section 4.2. The input is word embedding matrix of some batch size b.

2. **Convolutional Feature Map**: The input then goes through a series of Convolution Operations. Each convolution operation has some filters, strides and outputs associated with it. We have not used any Pooling operation as we didn't want to lose any crucial information and instead we have simply moved on with Stride operation.

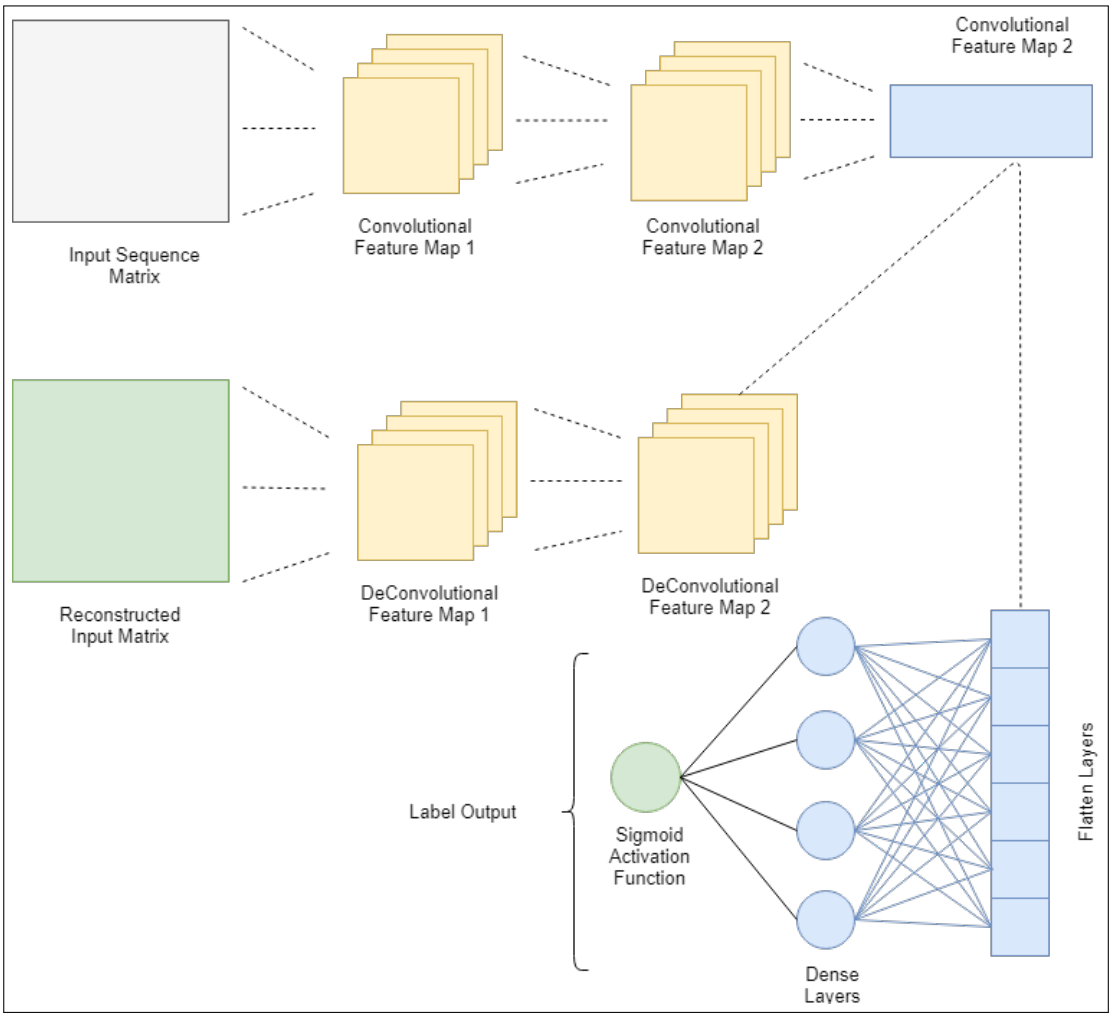


FIGURE 4.2: CNN Model Architecture - First Architecture

3. **Label Output and Sigmoid Activation Function**: This Output from the Encoder Network is consumed by 2 parts; Label Output Part and Reconstruction Output Part respectively. Label Output part is for learning the classification ability whereas Reconstruction Output Part focuses on reconstructing the output. The output from the Encoder Network is flattened out first and then mapped to a single Dense Layer consisting of n Neurons. Each item of the Encoder Network Output is mapped to every single neuron in the Dense Layer. The Dense Layer is finally connected a Sigmoid Unit. The loss function used for label output is binary cross entropy which returns a probability between 0 and 1 which is given by:

$$-1/N \sum_{i=1}^{N} y_i.log(p(y_i)) + (1 - y_i).log(1 - p(y_i))[38] \tag{4.1}$$

where, N is the total Number of data points, y_i is the truth value of i, $p(y_i)$ is predicted value of i.

4. **Deconvolution Operations**: Deconvolution Operation is the transpose of convolutional Operations. The purpose of having a deconvolution operation is in order to expand the data compressed by encoder part. Deconvolution operations are also known as decoder network. The internal architecture follows the explanation from section CNN based Decoder Network. The number of deconvolution operations follows the number of convolution operations. In the final step of the deconvolution operation, Reconstructed output is obtained whose dimension is same as the input dimension. In order to calculate the loss for reconstructed output, we will be using Mean Square Error (MSE) loss function which is given by:

$$1/N \sum_{i=1}^{N} (y_i - y_i^p)^2) \tag{4.2}$$

where, N is the total number of data points, y_i is true value of y and y_i^p is the reconstructed y.

4.4 Transfer Model Architecture

The initial model we have trained upon is Yelp data set, but thats not our sole purpose of this project. The main part relies on analyzing twitter data to set the threshold level for dangerous and not dangerous tweets. We will be utilizing transfer learning i.e. reusing the learned weights from earlier CNN model based on Yelp Polarity Data set. From the Figure4.3, the architecture remains almost the same except few changes in the classification side.

1. **Input**: The Input to this model is Twitter Word Embedding Matrix. The preprocessing steps and word embedding step follows the section 4.2. Batch input of tweets is fed to the model.

2. **Learned Weights**: The CNN Encoder Decoder Architecture remains almost the same with few layers discarded; particularly the label output part which is the main

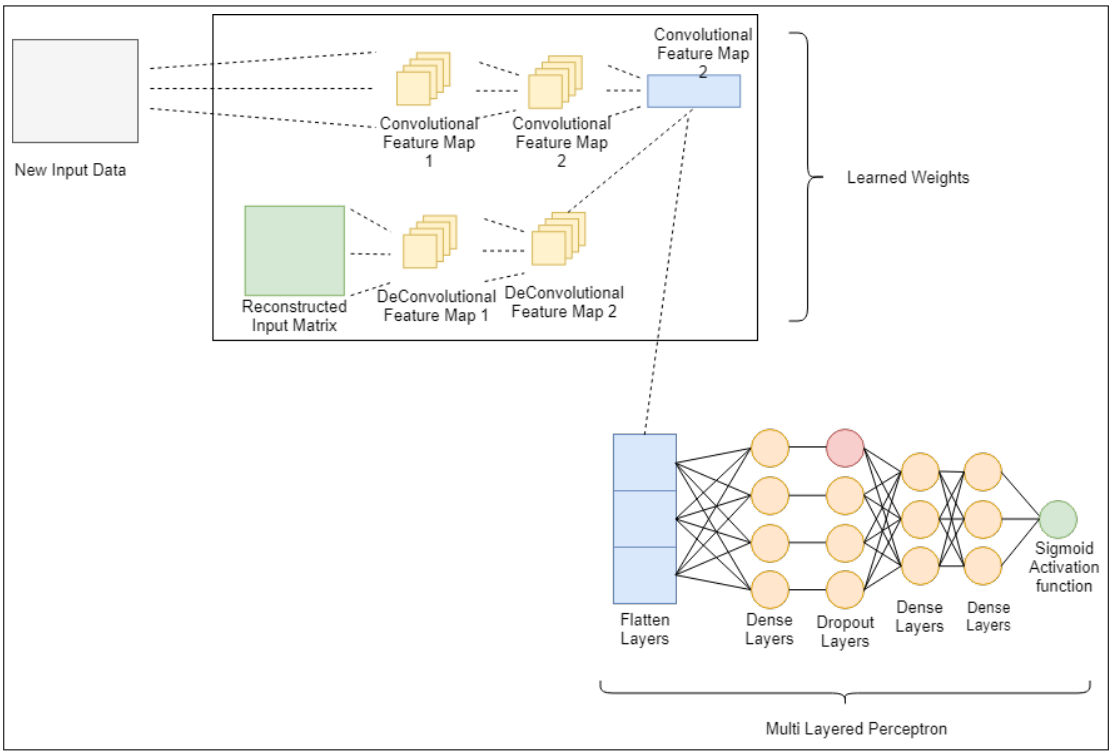


FIGURE 4.3: Transfer Model Architecture - Second Architecture

classification part is removed altogether. The input data flows through different CNN layers which is encoder network. Again, the output from encoder network is consumed by two streams, one is for reconstruction part and the other is for classification part. The only difference is that, we are making use of learned weights, which makes it easier for CNN model to learn and adapt new weights.

3. **Multi Layered Perceptron**: The output from the Encoder Model is connected to a set of Dense layers. After the first dense layer, a dropout is introduced to the model. The purpose of having a dropout is in order to reduce the overfitting on the model and make the model adapt to new features without being completely dependent on some particular features. After the dropout there are 2 Dense layers added which helps the model in learning different features of the data for classification purposes and then finally followed by a Sigmoid unit in the end. The loss used here is same as the earlier CNN i.e. binary cross entropy. In order to test the best architecture with different number of dense layers, we will be performing an extensive evaluation of different model parameters and tuning of parameters.

4.5 Defining Posterior Weights

Posterior weights are the weights defined on different output loss. As you can see from Figure4.2 and Figure4.3, there are two outputs associated; Reconstruction output and label output. So, there is 2 loss function that needs to be defined to refine the weights learned by the model. These loss functions can be assigned different weights which impacts the overall loss of the architecture. For instance, if low magnitude is set on reconstruction output, then the model focuses more on improving the classification ability rather than reconstruction output as there is more weightage on the classification part. Both the model described above, has posterior weights defined which controls the overall loss of the architecture.

4.6 Post Transfer Model

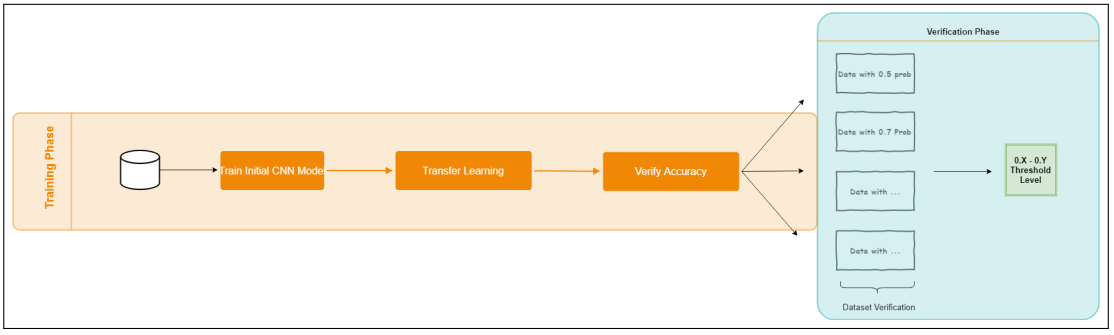


FIGURE 4.4: Threshold Apprehension

The Figure4.4 describes the process after training of all the models. The Validation and verification phase are performed after training all the models. The accuracy of the model is recorded and is further taken into consideration for threshold apprehension phase.

1. **Verification Phase**: Predicted probability along with the tweet text is taken into consideration in this phase. Different range of probability is verified for the dangerous level. For instance: We will be checking all the tweets which has prediction probability of more than 0.5 but less than or equal to 0.6. This will give us a glimpse of all the filtered tweets. These filtered tweets are compared with probability level of `0.6` `0.7` in order to check the difference between the type of tweets. Ideally, lower level probability prediction tweets are the cyber attacks

which has happened in the past while the high range of probability predictions capture more primitive information about the attacks. The mid-range prediction of tweets between `0.7` `0.9` is the prediction range which captures sophisticated linguistic information from tweet data which classifies it as dangerous or not so dangerous and basically sets the threshold apprehension level.

4.7 Base line CNN model architecture

To have a fair evaluation, we have developed a baseline system comprising of CNN components. The architecture follows similar structure from [5].

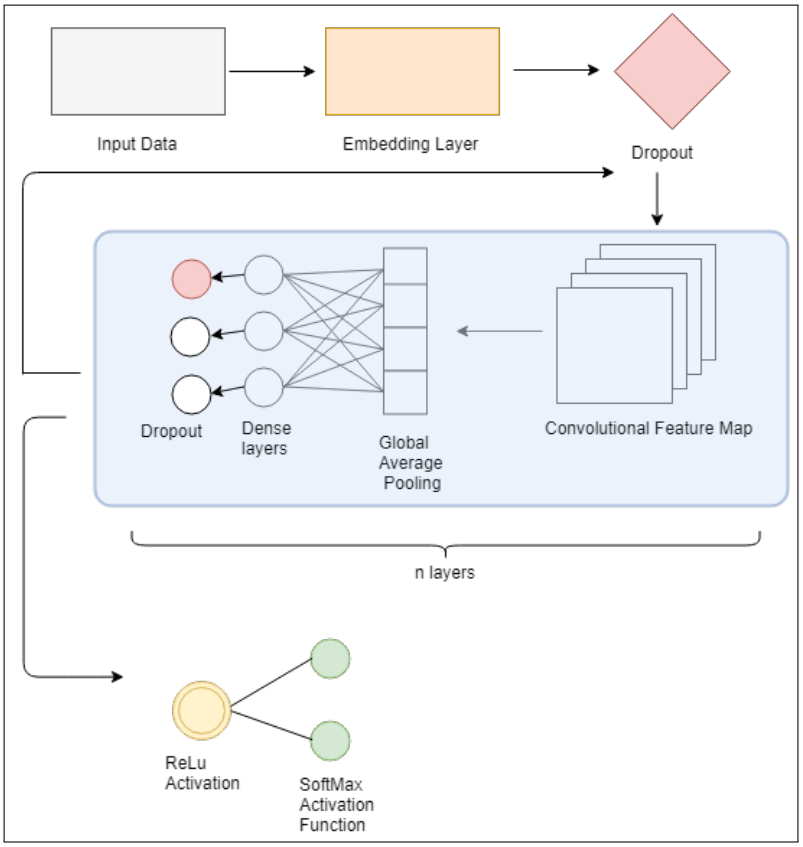


FIGURE 4.5: Base Line CNN

As from the Figure4.5, Embedding Layer is introduced after the input, i.e. We are not using a pre-trained word embedding matrix but instead we are learning the word vector weights using the Embedding Layer. A Dropout layer is introduced immediately after the Embedding Layer as the regularization technique. The Dropout helps the model to be independent of some of the features by randomly setting some values to zero. From the image, Multiple (n) blocks of the following layers are added in sequence:

1. **Convolutional Feature Map**: A convolution operation with some filter size and feature map is applied with ReLu activation function for the convolution operation.

2. **Global Average Pooling**: 2. We then apply Global Average Pooling (GAP) Operation over the feature map to reduce the over fitting by reducing the total number of parameters. GAP operation reduces the dimension of the tensor from h X w X d to dimension 1 X 1 X d by simply taking average of all hw values where h is the height, w is the width and d is the depth [39].

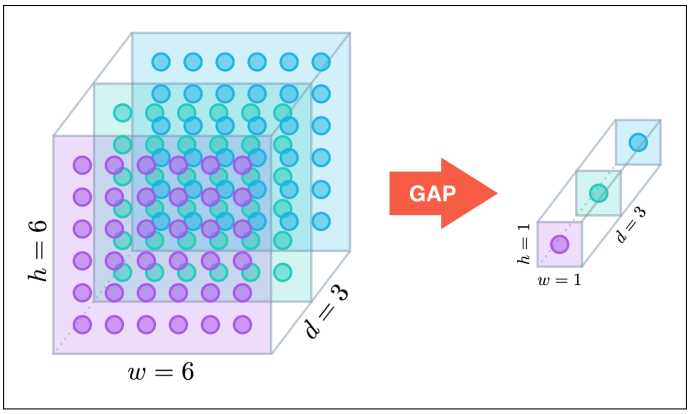


FIGURE 4.6: Global Average Pooling [39]

3. Dense Layers are connected to the GAP operation to learn the features extracted which is followed by a Dropout to prevent the over fitting during forward propagation.

After adding n number of the above blocks, the network is connected to a ReLu Activation function and then finally to a SoftMax Layer consisting of c number of neurons where c is the number of classes.

4.8 Experimental CNN Networks

In this section we focus on different variants of the model discussed above.

1. **Config 1: Baseline CNN**: In order to set the benchmark for CNN, we will be trying out the baseline architecture with filter sizes as [3, 4, 5] and number of filters = [100 ,100, 100]. Number of epochs to be trained for is 10 with Early

Stopping Criteria set to patience level 2 i.e. The model will stop training if there is no change in accuracy/ loss is observed for 2 continuous epochs.

2. **Config 2: With Transfer Learning**: In this part, we will be training our initial model for 30 epochs. The trained model weight with architecture will be reused for learning to classify twitter data. Following are all the parameters which we aim to evaluate for this configuration:

TABLE 4.1: Configuration 2 Architectures

Sr.no	**Freeze Initial Model Weights**	**Reconstruction op weight**	**Label op weight**	**Final Dense Layers**
1	False	0.4	1.0	2 layers each consisting of 300 neurons
2	False	0.8	1.0	2 layers each consisting of 300 neurons
3	False	0.4	1.0	3 layers each consisting of 300 neurons
4	True	0.4	1.0	2 layers each consisting of 300 neurons

3. **Config 3: With Partial Transfer Learning**: Partial transfer learning refers to not fully trained Initial model. The motive here is to train the initial model for only few epochs, before it reaches convergence and then transfer the learning to the new model. This way the new model will not have to learn the weights from start but at the same time it wont receive the weight which can be relied upon. The only purpose of this partial transfer learning is to give a head start to the model instead of training it from scratch.

Within each Config, for each convolution in the model, the parameters that needs to be defined are number of filters, filter size, stride size, max sentence length (max sentence length for twitter data), number of dense layers to add for classification output.

4.9 Computing Resources

The complete project is written in Python 3.6, making heavy use of Deep Learning library TensorFlow that has the power to exploit GPU acceleration. Our setup requires total hours to run all the configurations.

Experiments were performed on a single Dell Alienware 17R4 machine which is quipped with Nvidia GTX 1060 and Intel 7700HQ Chipset with 24GB DDR4 memory.

Chapter 5

Results and Evaluation

We have compared different Architectures, listed in section 4.8. Accuracy, Loss, F1 Score was used to quantitatively evaluate the output obtained. Furthermore, we also provide an in-depth analysis of test on original twitter data for company specific attack. We will focus our discussion on the following topics:

1. Study of Architecture.
2. The impact of partial training on transfer Learning.
3. The Impact of posterior weights.
4. Threshold Apprehension.

5.1 Comparing Network Architectures

5.1.1 Configuration 1

Configuration 1 comprises of the baseline CNN with parameters such as filter sizes as [3, 4, 5] and number of filters = [100 ,100, 100]. The model was trained on twitter data and the results obtained are:

Table 5.1: Configuration 1 Scores

Epochs	**Train Acc**	**Train Loss**	**Val Acc**	**Val Loss**	**F1 Score**
3	0.9575	0.1185	0.8563	0.4127	0.823

(a) Validation Accuracy (b) Validation Loss

FIGURE 5.1: Configuration 1 Scores

The model was supposed to run for 10 epochs but since no change in accuracy / loss was detected for last 2 epochs, hence the model stopped training (Early Stopping patience level set to 2). Clearly the model has scored good scores but there are huge signs of over fitting which reduces the overall performance of the model. No Learned Word Embeddings were used in this configuration and instead an Embedding Layer was used to learn the word vector weights. Despite the presence of Global Average Pooling Layer and Dropouts in the architecture, the model is highly over fitting as we can see from the Figure5.1.

Immediately after epoch 1 then loss is shooting up and accuracy is falling which is a big sign of model over fitting.

5.1.2 Configuration 2

The scores obtained for the configurations from the Table4.1 are listed in Table5.2 :

TABLE 5.2: Configuration 2 Scores

Config	**Train Acc**	**Train Loss**	**Val Acc**	**Val Loss**	**F1 Score**
1	0.975	0.346	0.867	0.561	0.850
2	0.980	0.554	0.853	0.770	0.829
3	0.984	0.285	0.866	0.963	0.847
4	0.676	1.99	0.685	1.925	0.836

Comparing the scores from the above table, the Validation Accuracy obtained is very near to the baseline CNN. However, there was a significant amount of over fitting in the baseline CNN which impacts the performance whereas using this Transfer Learning Architecture, we have tried to reduce the amount of over fitting. The reason for baseline

model over fitting is the because of less amount of data used for training purposes as CNN is data hungry and 20k records is not a substantial amount of data for CNN. In the above Table4.1, Config 1 is the best performing Configuration for this twitter data set using transfer learning. The final configuration is the worst performer as the trainable layers of initial model is frozen and the weights except for the MLP layers are not allowed to update.

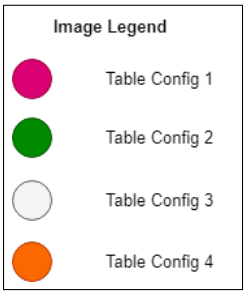
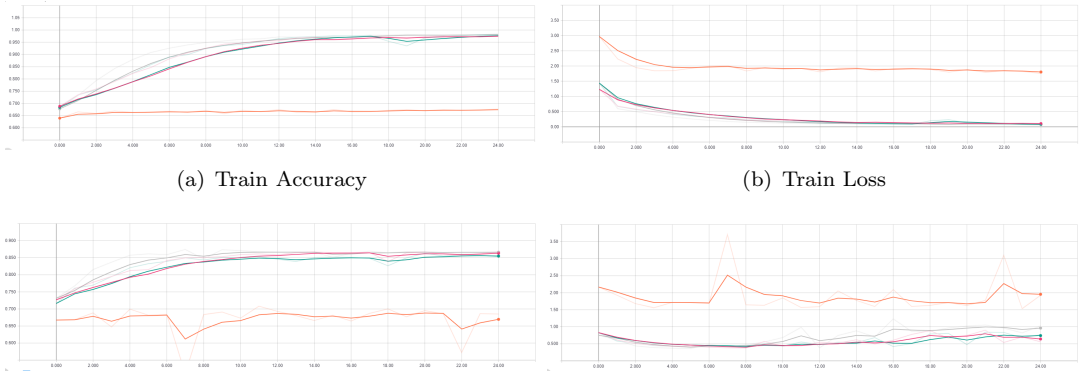


FIGURE 5.2: Config 2 - Legends



(a) Train Accuracy

(b) Train Loss

(c) Validation Accuracy

(d) Validation Loss

FIGURE 5.3: Configuration 2 Scores

The Figure5.3 is generated using TensorBoard which is powered by TensorFlow [40]. Figure5.3(a) shows training accuracy which is almost similar to all configurations except 4. In Figure5.3(b), the training loss has become almost parallel to the the x-axis which means the model is not learning any more. Figure5.3(c) represents validation accuracy. Validation accuracy is also almost constant but as compared to the baseline CNN, the model is not over fitting in this case as it is not deflecting from any point but instead it is steady. Again, the the lowest accuracy is obtained by configuration 4. Figure5.3(d) represents validation loss. The loss is constant for Configuration 1 and 2 whereas it seems to be diverging for Configuration 3 where we introduced few more layers which means more trainable parameters. Compared to Configuration 4, it is not the worst performer but Configuration 1 and 2 performs well as compared to the rest of the others.

5.1.3 Configuration 3

This Configuration trains the Initial model for only 1 epoch and then its used for transfer learning. The 1 epoch training of initial model is only to give a head start to the Transfer Learning Architecture for initializing the weights. The scores obtained are listed in the Table5.3:

TABLE 5.3: Configuration 3 Scores

Epochs	**Train Acc**	**Train Loss**	**Val Acc**	**Val Loss**	**F1 Score**
3	0.905	0.42	0.893	0.258	0.880

The Validation Accuracy obtained in this configuration is the best as compared to all the above configurations with validation accuracy reaching as high as 89.3%. Also the F1 score achieved for this configuration is highest as compared to the rest of the others.

5.2 Threshold Apprehension

The final layer of all the model returns a prediction value between 0 to 1 where tending towards 0 means negative and tending towards 1 means positive class. For all our experiments, predicted value above 0.5 is considered as class 1 whereas predicted value below 0.5 is considered as class 0.

```
true_pos_predicted[(true_pos_predicted.prob > 0.5) & (true_pos_predicted.prob < 0.6)]
```

	Unnamed: 0	text	test_y	prob	pred
146	146	showing vulnerability literally the death of a...	1.0	0.589072	True
159	159	update nasa openvsp 3.16 . 1 denial of service	1.0	0.587415	True
253	253	ddos attack from anonymouscatalonia cripples b...	1.0	0.567849	True
272	272	cracking ransomware ransomwarrior victims can ...	1.0	0.524323	True
275	275	cryptojackers exploit critical apache struts f...	1.0	0.526306	True
276	276	ransomware and other malware can be costly and...	1.0	0.594556	True
321	321	localized iterative vulnerability assessments ...	1.0	0.568523	True
329	329	babel user are quite large but there is only o...	1.0	0.593178	True
351	351	spyware company leaves terabytes of selfies te...	1.0	0.599717	True
358	358	is your manufacturing company protected agains...	1.0	0.554829	True
383	383	dear you may have heard of some of my work - g...	1.0	0.580678	True
438	438	this old ransomware is now bitcoin-stealing ma...	1.0	0.587563	True
493	493	threatlist ransomware attacks down fileless ma...	1.0	0.572008	True
545	545	cve -2018-13871 hdf 5 an issue was discovered ...	1.0	0.544605	True

FIGURE 5.4: Threshold 0.5 - 0.6

Threshold apprehension is the process of comparing text with predicted label score. We filter the positive predicted text values based on the prediction range; e.g. 0.5 0.6. We

compare the textual data in order to gain insights about the data. The information will be used for deciding the threshold apprehension level for tweets; i.e. any text prediction within the specified range makes it eligible be flagged.

Figure5.4 represents the Tweet test data in lower prediction range (0.5-0.6). Analyzing the text shows that most of the information has happened in the past or is a general computer security news.

Figure5.5 represents the tweet test data in upper prediction range (0.9 1.0). The predicted probability text data contains more primitive and general information rather than the information which we need.

```
true_pos_predicted[(true_pos_predicted.prob > 0.9) & (true_pos_predicted.prob < 1.0)]
```

	Unnamed: 0	text	test_y	prob	pred
0	0	best way to build empathy is through honesty a...	1.0	0.991308	True
1	1	cryptocurrency scams replacing ransomware as a...	1.0	0.995232	True
6	6	good slides the advanced exploitation of 64 - ...	1.0	0.999999	True
7	7	cve -2018-1000532 beep beep version 1.3 and up...	1.0	0.993755	True
10	10	you can t get to courage without walking throu...	1.0	0.999961	True
14	14	hmc says ransomware attack turned into healthc...	1.0	0.952409	True
15	15	looks like we were ddos'd over 10gbit someone ...	1.0	0.999997	True
16	16	work on ransomware trends in design of ransomw...	1.0	0.994091	True
17	17	winning the solution is fantastic for unstruct...	1.0	0.997927	True
18	18	yes george you can come ddos on release with y...	1.0	0.992089	True
19	19	add a datto backup disaster recovery solution ...	1.0	0.996188	True
20	20	cryptocurrency scams replacing ransomware as a...	1.0	0.999643	True
21	21	boi we ddos by and my unseen and you even woke...	1.0	0.981501	True
22	22	anyways thanks for being open and honest and s...	1.0	0.986776	True
23	23	vulnerability is a gift and don t let anyone t...	1.0	0.961679	True

FIGURE 5.5: Threshold 0.9 - 1.0

Figure5.6 represents Twitter test data in the range of 0.7 0.8. This range returns all the data which indicates the attack is happening or it is dangerous. Analyzing the data shows that most of the tweets belongs to a sophisticated vulnerability, CVE attack, Ddos attack or a domain specific news. Assessment of different value range shows that this range has excellent results as compared to the rest of the other brackets and has the potential to mark the tweet as suspicious or even flag it.

Hence, for this dataset our threshold apprehension range will be in between 0.7 0.8. If we need to record all types of activities that can be flagged, then simply a margin of 0.1 can be added on either side to increase the apprehension range.

```
true_pos_predicted[(true_pos_predicted.prob > 0.7) & (true_pos_predicted.prob < 0.8)]
```

	Unnamed: 0	text	test_y	prob	pred
49	49	active attacks detected using apache struts vu...	1.0	0.744522	True
105	105	there is a retropi in this dwelling that think...	1.0	0.796101	True
157	157	update ippulse 1.92 tcp port denial of service	1.0	0.771426	True
162	162	update hd tune pro 5.70 denial of service	1.0	0.704182	True
213	213	cracking ransomware ransomwarrior victims can ...	1.0	0.745759	True
221	221	don t miss the link 11 live webinar on septemb...	1.0	0.758173	True
309	309	_jynik yup hopefully vulnerability scans becom...	1.0	0.737954	True
363	363	ransomwarrior victims can now retrieve files f...	1.0	0.729732	True
368	368	what's funny is that i knew people on skype th...	1.0	0.710602	True
369	369	ddos attack from anonymous catalonia cripples ...	1.0	0.790868	True
483	483	. reports botnets serving up more multipurpose...	1.0	0.716213	True
511	511	yo zoho stop ddos attacks to know business is ...	1.0	0.758173	True
512	512	financial institutions ' consumer data exposed...	1.0	0.706178	True
526	526	cracking ransomware ransomwarrior victims can ...	1.0	0.781329	True
527	527	hack-back extensively compromises c2 infrastru...	1.0	0.732874	True
559	559	wordpress rest api 0day exploit is out freelan...	1.0	0.701590	True
609	609	just want to say i think it's really cool that...	1.0	0.710602	True

FIGURE 5.6: Threshold 0.7 - 0.8

Chapter 6

Conclusions and Future Work

6.1 Conclusion

In this thesis, we described an Encoder Decoder based CNN network for sentiment threshold apprehension or basically sentiment prediction. We experimented different models ranging from deep baseline CNN architecture to Encoder Decoder Based CNN architecture. Since, deep CNN networks are data hungry, we demonstrated the use of transfer learning when there are few datapoints. We discussed the importance of using a full-fledged trained initial model for transfer learning vs the importance of partial training of initial model for weight initialization for transfer model. We presented an in-depth analysis of different configuration with respect to transfer learning in table We showed that our architecture performs well when using partial trained transfer model. We discussed about verifying prediction confidence level and then setting the threshold apprehension mark to flag the tweets / data - which we achieved as 0.8 for our dataset.

6.2 Future Work

There are multiple ways the thesis can be improved. We present four ideas to improve the project in future. Additional resources on the use of latex is below.

1. **Tweet Verification:** We built a scraper using Twitter API to scrape real world domain specific tweets like Sony attack which happened in 2014. We tried to

run our prediction using the real data which we retrieved. However, testing the prediction for real world tweets doesn't produce as desirable results as we expected. This is because the model is strictly trained on limited amount of cyber security tweet data. So, in order to or use it for deployment purposes, the twitter data can be filtered based on the arguments passed to the twitter API to know more about the user. A verified used with security background can be assigned more weightage as compared to a non-verified user. This weighting mechanism can also be used as an extra feature when training the model. This way, the model will automatically learn to assign weights to different types of tweets. However, the downside to this will be that the complete tweet with the arguments needs to be stored and processed during training and testing phases.

2. **Focus on reconstruction ability:** The loss function for reconstruction loss in all our architectures is simple MSE. However, a more sophisticated loss function with gradual weight decrease can be adopted. By using this strategy, the loss weights will have a decay factor associated with them to adequately and efficiently train the model (e.g. Reconstruction loss weight will start at 1 with decay factor of -0.01 after every 10 epochs whereas label output loss will start at 0.3 and will have the decay factor of +0.01 after every 10 epochs). Having said that, if we introduce decay factor, which means more refining of our models, which leads to having a greater number of training cycles.

3. **Network Parameters:** Due to the nature of Deep learning models to ingest more data, it requires significant amount of time to run this architecture. So, the architecture which we tested, introduces only three convolutional layers which basically conveys n-grams. Going further deeper into the network with multiple regularization techniques can contribute a significant jump in the performance of the model.

4. **Data set:** Due to the limited amount of domain specific train data availability, there can be improvements made if we can introduce fake data points using Generative Adversial Network with reinforcement learning on twitter text. This could prove to be very powerful architecture if developed properly.

Bibliography

[1] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[2] C. Sabottke, O. Suciu, and T. Dumitras, "Vulnerability disclosure in the age of social media: Exploiting twitter for predicting real-world exploits." in *USENIX Security Symposium*, 2015, pp. 1041–1056.

[3] H. Sagha, N. Cummins, and B. Schuller, "Stacked denoising autoencoders for sentiment analysis: a review," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 7, no. 5, p. e1212, 2017.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[5] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[7] P. Mishra, R. Rajnish, and P. Kumar, "Sentiment analysis of twitter data: Case study on digital india," in *2016 International Conference on Information Technology (InCITe)-The Next Generation IT Summit on the Theme-Internet of Things: Connect your Worlds.* IEEE, 2016, pp. 148–153.

[8] V. D. Nguyen, B. Varghese, and A. Barker, "The royal birth of 2013: Analysing and visualising public sentiment in the uk using twitter," in *2013 IEEE International Conference on Big Data*. IEEE, 2013, pp. 46–54.

[9] Statista, "Number of social media users worldwide from 2010 to 2021 (in billions)," 2019, last accessed 1 March 2019. [Online]. Available: https://www.statista.com/statistics/278414/number-of-worldwide-social-network-users/

[10] Wikistat, "Global internet usage," 2018, last accessed 1 March 2019. [Online]. Available: https://en.wikipedia.org/wiki/Global_Internet_usage

[11] V. K. Singh, R. Piryani, A. Uddin, and P. Waila, "Sentiment analysis of movie reviews: A new feature-based heuristic for aspect-level sentiment classification," in *2013 International Mutli-Conference on Automation, Computing, Communication, Control and Compressed Sensing (iMac4s)*. IEEE, 2013, pp. 712–717.

[12] E. Boiy and M.-F. Moens, "A machine learning approach to sentiment analysis in multilingual web texts," *Information retrieval*, vol. 12, no. 5, pp. 526–558, 2009.

[13] G. Vinodhini and R. Chandrasekaran, "Sentiment analysis and opinion mining: a survey," *International Journal*, vol. 2, no. 6, pp. 282–292, 2012.

[14] H. Saif, Y. He, and H. Alani, "Semantic sentiment analysis of twitter," in *International semantic web conference*. Springer, 2012, pp. 508–524.

[15] A. Tripathy, A. Agrawal, and S. K. Rath, "Classification of sentimental reviews using machine learning techniques," *Procedia Computer Science*, vol. 57, pp. 821–829, 2015.

[16] Y. Bengio *et al.*, "Learning deep architectures for ai," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[17] X. Glorot, A. Bordes, and Y. Bengio, "Domain adaptation for large-scale sentiment classification: A deep learning approach," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 513–520.

[18] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh annual conference of the international speech communication association*, 2010.

[19] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Advances in neural information processing systems*, 2016, pp. 1019–1027.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[21] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[22] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *International Conference on Artificial Neural Networks*. Springer, 2011, pp. 52–59.

[23] R. Socher, C. C. Lin, C. Manning, and A. Y. Ng, "Parsing natural scenes and natural language with recursive neural networks," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 129–136.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[25] B. Liu, "Sentiment analysis and opinion mining," *Synthesis lectures on human language technologies*, vol. 5, no. 1, pp. 1–167, 2012.

[26] R. Moraes, J. F. Valiati, and W. P. G. Neto, "Document-level sentiment classification: An empirical comparison between svm and ann," *Expert Systems with Applications*, vol. 40, no. 2, pp. 621–633, 2013.

[27] V. Jagtap and K. Pawar, "Sentence-level analysis of sentiment classification."

[28] B. Liu, *Web data mining: exploring hyperlinks, contents, and usage data*. Springer Science & Business Media, 2007.

[29] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, "Semi-supervised recursive autoencoders for predicting sentiment distributions," in *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics, 2011, pp. 151–161.

[30] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1631–1642.

[31] Q. Qian, B. Tian, M. Huang, Y. Liu, X. Zhu, and X. Zhu, "Learning tag embeddings and tag-specific composition functions in recursive neural network," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, vol. 1, 2015, pp. 1365–1374.

[32] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," *arXiv preprint arXiv:1404.2188*, 2014.

[33] Y. Zhang, D. Shen, G. Wang, Z. Gan, R. Henao, and L. Carin, "Deconvolutional paragraph representation learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4169–4179.

[34] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in neural information processing systems*, 2015, pp. 649–657.

[35] V. Behzadan, C. Aguirre, A. Bose, and W. Hsu, "Corpus and deep learning classifier for collection of cyber threat indicators in twitter stream," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 5002–5007.

[36] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[37] J. Perkins, *Python text processing with NLTK 2.0 cookbook*. Packt Publishing Ltd, 2010.

[38] R. Gmez, "Understanding categorical cross-entropy loss, binary cross-entropy loss, softmax loss, logistic loss, focal loss and all those confusing names," 2018, last accessed 1 March 2019. [Online]. Available: https://gombru.github.io/2018/05/23/cross_entropy_loss/

[39] A. Cook, "Global average pooling layers for object localization," 2017, last accessed 20 March 2019. [Online]. Available: https://alexisbcook.github.io/2017/global-average-pooling-layers-for-object-localization/

[40] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

Appendix A

How to replicate results

The main requirements for this project are Tensorflow GPU 1.12.0 and Python 3.6.(All the other requirements with their versions are clearly specified in requirements.txt file provided in the Github link)

The complete project is hosted on `Github` repository:

`https://github.com/hippie-dev/sentiment-threshold-apprehension`

Due to the size of the data used in this project, the models and Tensor board graph generated by this project, are huge. So the heavy data is uploaded to Google Drive. The link to the Google drive is in Github repository with required description of where to place it in folders.

Note: There is no need to download contents (Tensorboard images, models) from Google drive except the data if you intend to run the model on your computer.

Steps required to run the project:

1. Make sure you have Python 3.6 install or if not then you can download it from the official Python sources.
2. Clone the Github repository.
3. Run the following command to install all the requirements: pip install -r requirements.txt

4. Tune the parameters in `keras_config.py` as per your needs. (Like setting training of transfer model to True or False, etc)

5. Run main.py to run the main code: python main.py

Tensorboard is initialized in this project, so make sure you have some extra space in your drive as the run may generate large image files depending on the type of data.

To run tensorboard:

1. Open up the folder

2. Use the following command: tensorboard - -logdir=graph