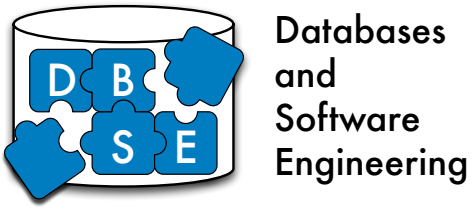
University of Magdeburg

School of Computer Science



Master's Thesis

Deep Sentiment Representation Through Char-level CNN and LSTM

Author:

Prabhu Kumar Reddy Appalapuri

March 14, 2019

Advisors:

Prof. Gunter Saake

Institute of Technical and Business Information Systems

Dr. Jim Sellmeijer

Search talent GmbH

Yang Li, M.Sc.

AG Datenbanken & Software Engineering

Appalapuri, Prabhu Kumar Reddy:
Deep Sentiment Representation Through Char-level CNN and LSTM
Master's Thesis, University of Magdeburg, 2019.

Abstract

Sentiment analysis is not the analysis of emotions, but it is the analysis of people's opinions on a specific topic (e.g. product reviews and social media comments). In recent years, deep learning has gained more popularity for the use in sentiment analysis as different types of network architecture designs. They have significantly improved analysis accuracy over shallow machine learning methods. The deep neural network employs multiple layers for processing, and it has excellent capabilities for data modelling in order to solve challenges (e.g. polarity detection, cyber security, marketing and fraud detection). They are applied in large volumes of domain-specific data such as image recognition, sentiment analysis systems and internet search engines. Some of the more common techniques are *Convolutional Neural Network (CNN)* in which essential features can be extracted moreover, *Long Short Term Memory network (LSTM)* which maintains the sequential order of the data.

In my thesis work, a novel deep learning method was developed using *CNN* and *LSTM* at the lowest atomic representation. This method overcomes the weakness of CNN's sequential order and LSTM's important features extraction of the text for sentiment analysis. This method is adaptable to any alphabetic languages and uses one hot encoding, benefits of the integration of capturing important features and ability to learn long term dependencies. Based on the new method, there are two different variations of the method namely *CNN_LSTM* and *VDCNN_LSTM* were developed in the thesis at different layer depth of ten and thirty. These proposed models have outperformed existing state-of-the-art methods for a single language. Furthermore, the model for multiple languages performed better than individual techniques.

Acknowledgements

First, I would like to thank Prof. Gunter Saake, who has given me an opportunity to write this thesis under his supervision. Also, I would like to mention my deepest gratitude to my advisers Dr. Jim Sellmeijer and Yang Li, who has given me a chance on this topic of interest, too. Their constant guidance and valuable discussions are only the main factors for the progress of this topic. Once again, I thank from the bottom of my heart for their patience and availability during the whole duration of this thesis.

I want to thank Mr. Benjamin Visser, the CEO of the Searchtalent GmbH, who has believed in me and given me an opportunity to write this thesis. I extend my gratitude to my colleagues for their motivation during my thesis work. Last but not least, I would like to thank Dr. Lining Yu, who has initially given shape to this thesis topic. Further, I want to thank Bauhaus University Weimar, for their resource supply for this thesis.

Further, I would like to thank my uncles: Prabhakar Reddy, Sudharshanam Reddy, Dr. Krupakar Reddy, Mohan Reddy and their families, who have supported me in all regards to the completion of my Master's thesis, my brother Prasad Kumar, my cousins and of course the whole-hearted support of my parents, without whose contribution this may not be possible. I would like to thank all my friends, namely Harinath Reddy and Nikhila Ponugoti and fraternity, who were part of my studies directly and indirectly. I would like to special mention Pfarrer. Kowol Siegfried, friends from Vaters Haus, Kirity Rapuru, Yeshwanth, Sudhakar, Spandana and Mounika Sri. Finally, I would like to give my regards to all the people who helped me to achieve my Master's degree.

Declaration of Academic Integrity

I hereby declare that this thesis is solely my own work and I have cited all external sources, which were used.

Magdeburg, March 14th 2019

Prabhu Appalapuri

Contents

List of Figures

List of Tables

1. Introduction

The purpose of sentiment analysis is to identify and extract subjective impressions from the data. In general, it is a process to determine the attitude of a speaker or a writer on a specific topic[1]. Also, the subjective impression is assumed as a binary position in opinions, e.g. good/bad, like/dislike etc. The extracted subjective information from a massive amount of data is used to make better decisions in business. Nevertheless, the manual analysis of subjective information from a huge amount of data is a difficult task. Hence, sentiment analysis is an automatic process to capture the sentiment of the people. One of the questions that can be answered through sentiment analysis is: "Is the product review positive or negative?". Such kind of question can be answered through the subjectivity of the text. Sentiment analysis (SA) is also known as opinion mining of the text, which can be extracted by using machine learning methods, especially deep learning.

Modern deep learning, a typical machine learning technique, is inspired by how the brain works and learns. It employs more layers and units, than other machine learning techniques, within what is called a "deep network". The essential concepts, building blocks and their working principles will be explained in the Chapter 2 including sentiment analysis and its importance. Deep learning is a subset of artificial intelligence and also known as a type of representation learning.

Since deep learning is part of AI [2], DL used to solve challenges from the different domain use-cases[3] in the field of natural language processing (NLP) such as sentiment analysis(SA). Multiple deep learning methods have been developed, which by itself are based on neuronal processes for measuring sentiment analysis such as *Convolutional Neural Network (CNN)*, *Long Short Term Memory Network (LSTM)*[4]. LSTM is often used in speech recognition, whereas CNN is famous for its use in computer vision. In the early days of computer vision, handcrafted features were used to classify images. Over the years, CNN has been improved to extract hierarchical features.

In most NLP applications, words are considered as basic units with continuous representations (i.e. vector representations of each word), and these words became

as state-of-the-art embeddings. However, representing a whole sentence with specific word embeddings is difficult because each sentence is composed of complex syntactic and semantic relations. The existing trend is to consider the entire sentence as a sequence of characters or tokens. The dependencies are learned between every two points in a sequence of characters and also able to remember for a long time by LSTM[5].

These deep learning methods can learn hierarchical representations from the text. Usually, the text has similar properties(e.g. sentences, phrases, words, n-grams are formed from characters) as images pixels[6]. In this thesis, we will have a closer look at combining the CNN and LSTM for sentiment analysis. However, the Searchtalent GmbH use-case of sentiment analysis is to detect the polarity of the candidate responses whether the candidate has responded to the job offer positively or negatively. Since the high-quality and labelled email conversation data is not available, so the product reviews from the *Amazon website* used as an alternative for the experimentation. The description of the data explained more clearly in Section 5.2.1 which is in three different languages namely English, German and French. Furthermore, my thesis work is also focused on providing multi-language support which can significantly benefit the business of the company in the real world.

1.1 Motivation

CNN is a type of deep neural network and similar to a multilayer perceptron network. These are popular for selecting local patterns (i.e. properties or correlation between characters in a text) without knowing a language and its syntactic or semantic structures[7]. Recurrent Neural Networks (RNNs) are also a type of deep neural networks and similar to a "Feed Forward Neural Networks". These are useful for following prediction where the order of the data matters. However, RNN has a vanishing gradient problem that is the weight values may explode (i.e. higher change in weight values) or may vanish (i.e. become closer to zero) while updating gradients during backpropagation, which makes model prediction ability worse. Long Short-Term Memory Network (LSTM) is a special type of RNN can handle vanishing gradient problem. These networks are useful for remembering information for longer periods[5]. Despite having CNN's features extraction ability and LSTM's sequential order ability, these above methods do have disadvantages. CNN lack the ability of sequential order. LSTM lack the ability of features extractions.

CNN was proven efficient in sentiment analysis for character level[7], and this approach shifted from the use of "Bag-Of-Words (BOW)" method that contains knowledge of the used words in the text. Whereas, "Term Frequency-Inverse Document Frequency (TFIDF)" which contains the relative occurrence of each word in the text. As opposed to bag-of-words, bag of n-grams can also be used in which it ensures that the partial information of the structure of text[8]. LSTM is different from CNN. LSTM can remember the relation between features over the time when it gets new input also known as long term dependencies between words. However, LSTM needed a list of unique words which are used in the text.[9].

For language specific model, more work needed to handcraft features for a particular language. While usage of such handcrafted features can only benefit that language

but not for other language and readjusting such features for other language require more effort. Usually, the text (i.e. raw data) is a series of characters and contains more knowledge about the structure of data. Moreover, the model can be feed with unprocessed data such as with spelling mistakes. Also, the model can handle a mix of multiple languages as well. However, most of the existing methods failed to learn directly from raw data and cannot handle a mix of languages.

Therefore, a combination of CNN and LSTM could extract features through CNN, and LSTM can handle long term dependencies. However, the literature fails to describe this proposed approach that could increase sentiment analysis accuracy. In order to research this, I will be using CNN, LSTM and the atomic level representation of the text (i.e. alphabets of the language). In this thesis, a *the combination of CNN and LSTM for sentiment analysis* will be implemented. In addition, I plan to investigate whether the combined model CNN-LSTM outperforms the individual CNN model and LSTM model. Moreover, I would like to identify the limitations of CNN-LSTM in terms of multiple languages, for instance, English, German and French. This CNN-LSTM will be implemented based on a new technology Pytorch (developed by Facebook), which manages computations similar to numpy, but with a more efficient GPU support in Python.

1.2 Goal of the Thesis

The goal of the thesis work is to develop a hybrid model by combining Convolutional Neural Network (CNN) and Long Short Term Memory Network (LSTM), thereby improving model efficiency by overcoming the weaknesses of the individual deep learning models and documenting its limitations. Additionally, I aim to develop a single model for multiple languages. Since human language is complex, informal and often used on social media, it is difficult for a sentiment analysis system to determine the polarity of the text. Also, each language has its own grammar rules. Therefore the atomic level (i.e. alphabets) representation of a language, excellent feature extraction ability of CNN and sequential order ability of LSTM may strengthen the combined model for sentiment analysis even though text is in multilanguage. Hence, to achieve this goal, I have to breakdown total work into the following logical steps

- Conduct a literature review to analyze existing approaches which support multilanguage sentiment analysis.
- Develop a combined character-level CNN-LSTM model which can be used for multiple languages and also used for topic classification, sentiment analysis.
- Document the limitations of CNN-LSTM.

1.3 Structure of the Thesis

This work is divided into six chapters and organised as follows:

Chapter 2, provides introduction of the necessary topics to understand this work. Therefore, we will go through sentiment analysis, its challenges, approaches and

techniques. Also, deep learning frameworks. This chapter aims to provide sufficient background knowledge of sentiment analysis and deep learning techniques.

Chapter 3, provides literature research related to the thesis topic. This section describes various methods, most closely related journals to this thesis and some relevant books. This chapter aims to present the latest up to date literature related to this thesis.

Chapter 4, describes the multilanguage sentiment analysis system. The aim of this chapter is to propose and explain the problem definition and approach in detail.

In **Chapter 5**, I evaluate the performance of the new method with three important types: *comparing with individual techniques*, *by increasing network depth* and *multilanguage model Vs single language model*.

Chapter 6, is a summary of the results and findings.

Chapter 7, provides an outlook for the future work.

2. Background

2.1 Sentiment Analysis

Sentiment analysis(SA) is used to detect subjectivity in a text[10]. In Sentiments analysis, the primary task is to recognise emotions. Moreover, polarity detection categorized as an advanced analysis. Processing these emotions is an essential task for polarity detection. "The public sentiments capturing through automation" has given opportunities such as to gain an overview of the public opinion about a specific topic. Also, It has given challenges for research such as analysing product preferences, political movements and marketing campaigns etc. This led to a new field of sentiment analysis and can be used to create recommender and customer relationship management systems. For example, these systems can identify the opinions of people about products whether they feel happy with new features or not and avoid products which are having negative sentiment.

Sentiment analysis has become a core factor for business intelligence in order to predict the public attitudes towards their products and to plot business strategies. There are a variety of sentiment tools available for example Brandwatch[1], Lexalytics[2], Sentdex[3], open source tools, e.g. EMOTICONS, SenticNet, Sentiment140 and many more[11].

2.1.1 The Existing Problems in Sentiment Analysis

Opinions are mainly studied through sentiment analysis of negative and positive statements. For example, read the following reviews of a Fossil watch.

Review(1)
Posted by Sharath on 18 May 2018
Highly recommended!

[1]https://www.brandwatch.com
[2]https://www.lexalytics.com
[3]http://sentdex.com

This is one of the classiest watches in the market. It is a sure shot head turner yet very subtle. Blends well with formals & casuals alike. Highly recommended!

Review(2)
Posted by Shivkumar on 25 July 2018
All the watch was good but straps not same each other.
Received the watch just now and the watch where defective. All the watch was good but straps not same each other.

By observing the above messages[4], it becomes clear that it is not simple to estimate whether an opinion is positive, neutral or negative. It is due to the format of the review as follows.

- **Structured sentiments:** These can be seen in formal sentiment reviews that were written by professionals, e.g. books or articles.
- **Semi-structured sentiments:** This type of sentiments can be found in a range between structure and unstructured sentiments. It depends on the advantages and disadvantages of the review and its content which are clearly stated by the writer. However, these contents are short phrases.
- **Unstructured sentiments:** These are found in free text format where the writer does not follow any constraints, the content might be having several sentences and no formal separation between pros and cons of the review.

Hence, each review structure requires more work for sentiment detection at various levels of text granularity's, e.g. paragraphs, sentences, terms etc.

2.1.2 Challenges in Sentiment Annotation

Most data found on the internet is unstructured and not categorised. To prepare structured data with labels (i.e. annotated data), it may vary depending upon the human annotator linguistic knowledge. However, it increases ambiguity in the data. At least two human annotators are required to reduce ambiguity. They must follow the same specifications that is based on the SA task specification, and rules will be defined for annotation throughout the data. Also, it will increase the quality of the data annotation because of more than one-time data annotation. However, having more number of human annotators means it is expensive. Hence, getting quality annotation data is difficult with simple and clear instructions, e.g. a simple task annotation whether it is positive, negative or neutral.

In the simple sentiment questionnaires approach, it will check whether the word is positive or negative. Also, it checks the word association with the polarity moreover which word is more positive[12] or is to ask respondents for the target of opinion identification and its sentiment[13]. These kinds of annotations referred to as "the semantic-role based sentiment questionnaires". The semantic-role based sentiment questionnaires are too detailed even though it is not sufficient for many scenarios. For example, some challenging sentence types are listed below[14]:

[4]https://www.amazon.co.in

- **Speaker's emotional state:** The speaker's emotional state could have the same polarity as the opinion expressed by the speaker. For example, a politician's tweet can mean both a negative opinion about a competitor's past foolishness, and a happy mental state as the news will impact the competitor negatively.
- **Success or failure of one side with respect to another:** Often the sentence opinion may be positive on one perspective of thinking. Never the less, the same sentence can be unfavourable to the prior opinion, e.g. Polar bear shot dead after attacking cruise ship guard.[5]
- **Neutral reporting of valenced information:** When a speaker does not show any emotional state but expresses situations or valenced information then it is difficult to categorise them as a neutral or negative, e.g. The war has created millions of refugees.
- **Quoting somebody else or re-tweeting:** These sentences are unclear and explicitly do not prove whether the opinions are same as the one who quotes to that of quotee.

2.1.3 Challenges in Applying Sentiment Analysis

The sentiment analysis systems may not be precise at defining each sentence polarity. However, they can recognise the number of changes in the sentences that are positive and negative. The most common applications of the sentiment analysis are sentiment concerning products, politicians, and companies, improving customer relation models and improving automatic dialogue systems. Moreover, they can be modified based on the application. For instance, an application requires identification of a threshold between positive to neutral and neutral to negative. Likewise, some applications require instances which are strong positives and negatives. Due to the massive availability of product reviews have led to more research in sentiment analysis with more diverse and compelling applications.

2.2 Approaches for Sentiment Analysis

Sentiment analysis investigates people sentiments, opinions, attitudes, evaluations, appraisals, emotions towards services, products, individuals, organisations, issues, topics events and their attributes[1]. The sentiment analysis can classify a piece of text based on the following criteria[15].

- The polarity of the sentiment expressed as positive, negative or neutral.
- The polarity of outcome[16].
- Whether the news is good or bad[17].
- The people's opinions whether they agreed or not about the topic[18].

In order to implement such systems, following Section 2.2.1 and Section 2.3 will explain different types of sentiment classification approaches and techniques.

[5] @BBCWorld posted in Twitter on 29 July 2018

2.2.1 Sentiment Classification Approaches

In a classification task, the target unit in a document classified at three different levels[19]:

- **Document level:** The entire document considers one single topic and expresses a positive or negative opinion.
- **Sentence level:** Each sentence considers an opinion and categorised as positive or negative.
- **Character level:** In this approach, the text considered at the lowest possible atomic representation or text viewed as a stream of characters without language semantics structures.

The above classification levels are classified through different approaches for sentiment analysis, namely: the knowledge-based, language-model driven and machine learning driven approaches[20].

1. **A knowledge-based approach** is the building of word lexicons with positive and negative classes. The sentiment weight values assigned before the application of sentiment-analysis[21].
2. **Language models approach** is the building of n-gram models based on the continuous sequence of n-items from a given sequence of text that is the sentence will be split up into a consecutive pair of words by moving one word forward at a time[22]. Below simple example gives you a clear understanding of n-gram:
   The sentiment analysis systems can be developed based on supervised or unsupervised methods
   Unigram: (The),(sentiment), (analysis), ...
   Bigram: (The, sentiment), (sentiment, analysis), ...
   Trigram: (The, sentiment, analysis),(sentiment, analysis, systems), ...
3. **Machine learning approach** is done based on supervised or unsupervised learning through the extraction of the features from the text and learns the model.

2.3 Sentiment Analysis through Machine Learning

The machine learning-driven methods have shown a good result in most of the sentiment analysis task for more than two decades. The sentiment analysis systems can be developed based on supervised or unsupervised methods. Among many machine learning-driven methods, deep learning has performed better than the shallow machine learning models.

2.3.1 Supervised Techniques

In supervised learning, the classifier learns to predict the data label (Y) from input (X). It uses manually labelled data. The most widely used supervised classifiers were built by using support vector machine (SVM), naïve bayes or decision tree (DT) methods. These supervised learning methods follow a specific workflow that features are extracted manually before training the model.

Moreover, the model references those features when analysing and classifying new data never the less, and these models can not learn pattern in the data by itself. Hence, these models are often categorised as shallow machine learning models. The shallow machine learning models performance depend on extracted features based on term frequency and inverse document frequency (TF-IDF). TF-IDF is a weight value for defining a word in the corpus and based on a word's definition in the context whereas Parts of speech (POS) tagging was used to select the features by tagging a word with its POS. In POS, nouns hold features of the product, adjectives and adverbs together hold opinion words.

2.3.2 Deep Learning Techniques

Recent technological advancement has sparked an increase in the creation of deep learning techniques. In this approach, the non-linear neural networks learn multiple levels of representation from the text data. These representations are transformed into higher representation and with a more abstract level. These used as features for classification task in natural language (NLP) field. The most classic example of deep learning is feed forward neural network and multilayer perceptron (MLP). The goal of the neural networks is to learn real-valued text representations automatically from the data. The most common techniques are described in Section 2.8 and Section 2.9.

2.4 Deep Learning Frameworks

Most of the big companies such as Amazon, Facebook, Microsoft and Google have implemented their own deep learning frameworks. These technologies have become more reliable for using deep learning methods. The most widely used deep learning frameworks are shown in Figure 2.1 such as Pytorch, Tensorflow, Keras, Theano and Caffe2 and MxNet etc.

1. ***Keras:***

   Keras is an interface for quickly building neural networks. Keras uses TenserFlow and Theano as a backend for neural networks and is written in Python. Keras is now included within the Tensorflow framework.

2. **Caffe:**

   Caffe was designed to deliver speed and modularity. Berkeley Artificial Intelligence Research was developed Caffe framework. The primary goal of Caffe is for building CNN's. Based on the success of Caffe, Facebook has developed a Caffe 2 Framework, which is included with pre-trained models.

Figure 2.1: Deep learning frameworks

3. **Theano:**

   Theano was first implemented in 2007 by Yoshua Bengio and the research team at the University of Montreal. It was the first widely used deep learning framework. It is a Python library and low-level deep learning framework. In 2017, the latest version of Theano was released.

4. **Tensorflow:**

   Tensorflow is an open source software library and was developed by Google Brain Team for numerical computation using data flow graphs. It is the most widely used deep learning framework and has a big community. In January 2018, Google has released TensorFlow1.5 with the new features and enhancements for faster computing. It was written in C++ and Python.

5. **Pytroch:**

   In 2017, Facebook released a pythonic version of Torch known as Pytorch. It analyses unstructured data using dynamic computational graphs. Pytorch has a customised GPU allocator, which offers deep learning models memory efficient and with a high speed. Pytorch framework has a significant market share, which is used by big companies, such as Facebook, Nvidia and Twitter. Pytorch is a developer-friendly and efficient framework.

Pytorch has been chosen for this thesis work in order to implement the model in a simple, intuitive way with easy access to GPU functionality in the computing machine.

2.5 Data Preprocessing

Data preprocessing is an initial step and an essential part in the sentiment analysis, which helps to interpret data. To understand data, preprocessing comprise of different techniques to convert unstructured data into structured data. Commonly used preprocessing techniques are,

- *Tokenisation*
- *Tagging*
- *Stemming*
- *Lemmatisation*

Further, preprocessing is extended to treating misspelled text and removable of stop words depending upon the task at hand. The advantages of preprocessing are

- Increases the classifier accuracy.
- more information regarding text.

2.5.1 Tokenisation

Tokenisation is a process of splitting up a document into various components such as words, phrases, symbols. These components are called tokens, which are separated by whitespace, punctuation marks or line breaks. The most common tokenisation techniques are:

1. **Sentence tokenisation:** It is know as sentence segmentation. The process of splitting a text document(corpus) into sentences based on period(.), or a semicolon(;), or a newline character called sentence tokenisation. For example,

   Text = """Tokenisation is a process of splitting up a document into various components, such as words, phrases, symbols. They are called tokens."""

   After sentence tokenisation,

   ["Tokenisation is a process of splitting up a document into various components such as words, phrases, symbols", "They are called tokens"]

2. **Word tokenization:** It is a process of splitting sentences into a list of constituent words. The word tokenization is done based on the following rules:

   - Split and separates periods that appear at the end of a sentence.

   - Splits and separates commas quotes when followed by white spaces.

- Most punctuation characters are split and separated into independent

After applying word tokenization,

[['Tokenisation', 'is', 'a', 'process', 'of', 'splitting', 'up', 'a', 'document', 'into', 'various', 'components', ',', 'such', 'as', 'words', ',', 'phrases', ',', 'symbols'], ['They', 'are', 'called', 'tokens']]

2.5.2 Stemming

In any language, stem, suffixes, prefixes are known as morphemes, and they are minimal grammatical units, i.e. a word cannot further divide into smaller parts. New words were created by adding suffix or prefixes to the word stem. The process of reducing a word into its root form is called Stemming. The root form is called a stem. Different stemming algorithms are available to get a stem of a word, e.g. porters stemmer[6], N-gram stemmer, krovetz stemmer and more[23].

For example, the words JUMPING, JUMPED and JUMPS were derived from its root JUMP. The word JUMP is the stem. But not always connecting to root form is not a stem. For example, study, studying and studies, when stemming into studi, which is an incorrect English word.

2.5.3 Lemmatisation

Lemmatisation is the process of deriving a root word from a word. The root word is known as a lemma. Usually, it is a dictionary entry or lemmatisation lexicon of a word. Sometimes lemmatisation may not work well on the words which are not in a lemmatisation lexicon or misspelt words.

For example,
cars ⇒ car
ate ⇒ eat
fancier ⇒ fancy

2.5.4 Tagging

Tagging is a process of categorising a word based on its role within a sentence, i.e. assigning to one of the lexical categories. The main parts of speech are noun, verb, adverb and adjectives. POS tagging is used to perform a specific analysis of NLP tasks such as grammar analysis and word sense disambiguation. Below Section 2.5.4[7] shows POS tagging in Spacy module.

2.6 Optimisation Algorithms

Optimisation algorithms are used to update a neural network model parameters such as weight and bias values. In simple terms, an error function will be updated or minimised by an optimisation algorithm depending upon the difference between predicted and actual value of the input used in the model. The error function

[6] http://snowball.tartarus.org/algorithms/porter/stemmer.html
[7] https://spacy.io/api/annotation#section-pos-tagging

POS	Description	Examples
ADJ	adjective	big, old, green, incomprehensible, first
ADP	adposition	in, to, during
ADV	adverb	very, tomorrow, down, where, there
AUX	auxiliary	is, has (done), will (do), should (do)
CONJ	conjunction	and, or, but
CCONJ	coordinating conjunction	and, or, but
DET	determiner	a, an, the
INTJ	interjection	psst, ouch, bravo, hello
NOUN	noun	girl, cat, tree, air, beauty
NUM	numeral	1, 2017, one, seventy-seven, IV, MMXIV
PART	particle	's, not,
PRON	pronoun	I, you, he, she, myself, themselves, somebody
PROPN	proper noun	Mary, John, London, NATO, HBO
PUNCT	punctuation	., (,), ?
SCONJ	subordinating conjunction	if, while, that
SYM	symbol	$, %, ©, +, ?, ×, =, :), ?
VERB	verb	run, runs, running, eat, ate, eating
X	other	sfpksdpsxmsa
SPACE	space	

Table 2.1: Part-of-speech tagging notations

depends on the weight(w) and bias(b) values. The error function is known as a loss, which is optimised during the training process of the neural network. Nevertheless, loss function depends on its gradient values which are evaluated by using first-order partial derivatives of one variable or multi-variable, and it is represents by a jacobian matrix. These algorithms use gradient values with respect to parameters to compute loss function $E(x)$. Hence, these are considered as first-order optimisation algorithms, e.g. gradient descent.

Gradient descent optimisation algorithms show whether the loss function is increasing or decreasing. Often, they were used as black box optimizers for neural networks and used widely for tuning model parameters, especially to minimise a loss function. Optimisation techniques are less time consuming, easy to compute and these converge fast on the large data sets.

2.6.1 Gradient Decent Variants

Depending upon data usage, three types of gradient descents are available to compute gradients of a loss function[24].

1. **Batch gradient descent:** It is also known as vanilla gradient descent and used to update parameters of the models.

$$\theta = \theta - \eta . \bigtriangledown J(\theta) \begin{cases} \eta \text{ is a learning rate,} \\ \bigtriangledown J(\theta) \text{ is loss function,} \\ J(\theta) \text{ gradient with respect to parameter } \theta \end{cases} \tag{2.1}$$

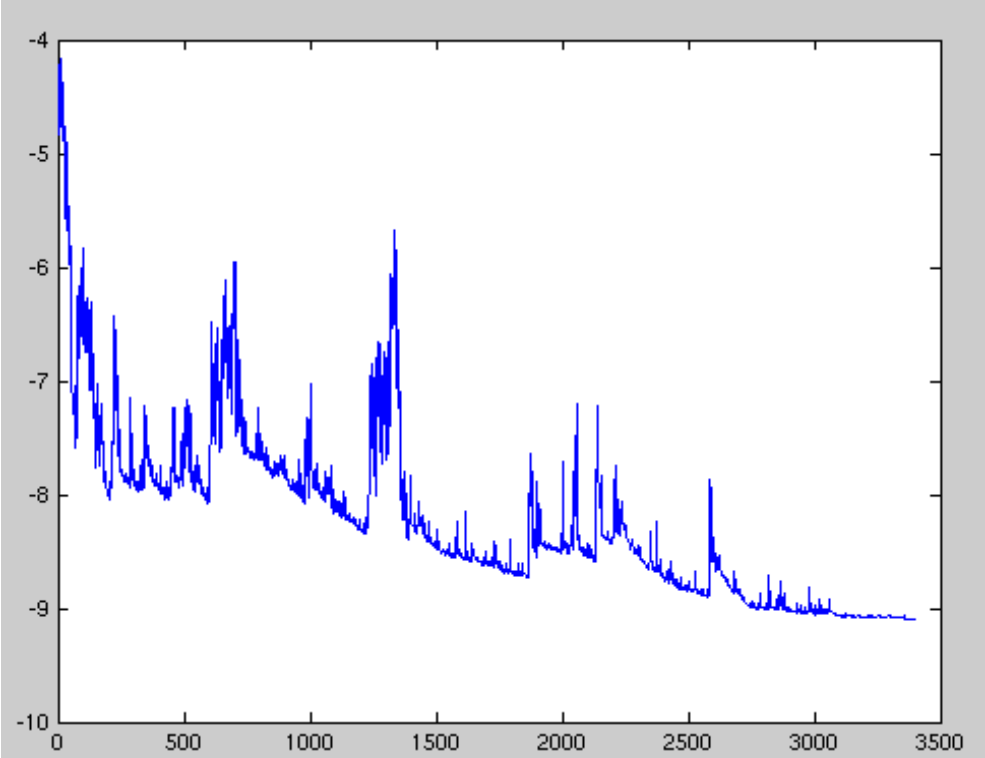


Figure 2.2: Fluctuations in loss function

For a single update, it requires to evaluate whole data set gradients. Therefore, it is slow and not useful for a data set which does not fit in memory.

2. **Stochastic gradient descent:** For each training sample $x^{(i)}$ and its label $y^{(i)}$, stochastic gradient descent (SGD) performs a parameter update. Therefore, it is faster than batch gradient descent. Where $x^{(i)}, y^{(i)}$ are training sample.

$$\theta = \theta - \eta . \bigtriangledown \theta J(\theta; x^{(i)}; y^{(i)}) \tag{2.2}$$

The frequent update of parameters by stochastic gradient descent causes high variance in parameters update and fluctuations in loss function, which was shown in Figure 2.2. Loss function fluctuations enable SGD to jump better local minima. However, it complicates convergence to an exact local minimum. When the learning rating decreases slowly, the optimisation result in same convergence as batch gradient descent behaviour such as converging to the global minima for convex and a local minimum for non-convex.

3. **Mini-batch gradient descent:** Mini-batch gradient descent composes with the advantages of batch gradient descent and stochastic gradient descent. Mini-batch gradient descent makes performance update of parameters for every mini-batch batch of n training samples. Thus proving with the advantages as reduced variance in parameters update to achieve a stable convergence and efficient computations of the gradient with respective to a mini-batch. Where $x^{(i:i+n)}, y^{(i:i+n)}$ are mini batch training sample

$$\theta = \theta - \eta . \bigtriangledown \theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)}) \tag{2.3}$$

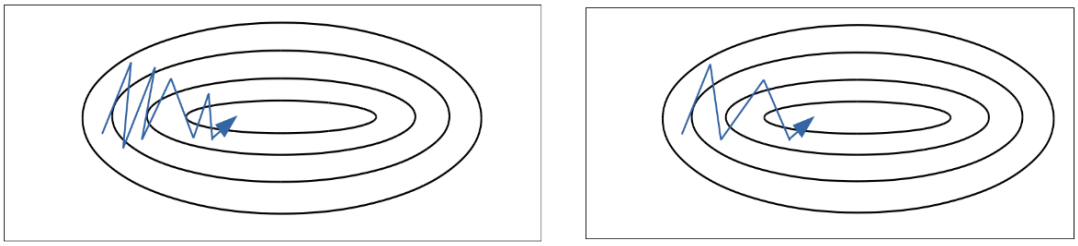


Figure 2.3: Momentum

Usually, the range of minibatch size is between 50 to 256 depending upon the application. Mini-batch gradient descent is the most common choice for training a neural network.

2.6.2 Gradient Descent Optimization Algorithms

Some of the deep learning optimisation algorithms will be explained in the section[24].

1. **Momentum:** SGD faces trouble while navigating to local optima where the surface curves much more steeply exist in dimension than in another. In these cases, SGD oscillates across the slope of the ravine with uncertain progress towards local optimum.

   Figure 2.3[8] **Momentum:** This method helps to accelerate stochastic gradient descent in the direction of optimal minima and reduces the ossifications. By adding a fraction γ of the update vector of the previous time step to the current update vector.

   $$v_t = \gamma v_t - 1 + \eta\theta \bigtriangledown J(\theta) \tag{2.4}$$

   $$\theta = \theta - v_t \tag{2.5}$$

   The value of the momentum γ, usually set to 0.9

2. **Adaptive momentum estimation:** Adaptive momentum estimation (Adam) is a method to compute adaptive learning rates for each parameter. Adam stores an exponentially decaying average of past gradients m_t, just like momentum.

   $$m_t = \beta_1 m_{t-1} + (1 - \beta_1).g_t \tag{2.6}$$

   $$v_t = \beta_2 v_{t-1} + (1 - \beta_2).g_t^2 \tag{2.7}$$

   where m_t, v_t are estimates the first moment(mean) and second moment(the uncentered variance) of the gradients respectively.

   From the scientists observations, when the Adam optimizer's m_t and v_t are initialized to zero valued vectors then adam is biased towards zero especially during initial time step and the decay rates are very small(i.e. β_1 and β_2 are

[8]https://www.willamette.edu/ gorr/classes/cs449/momrate.html

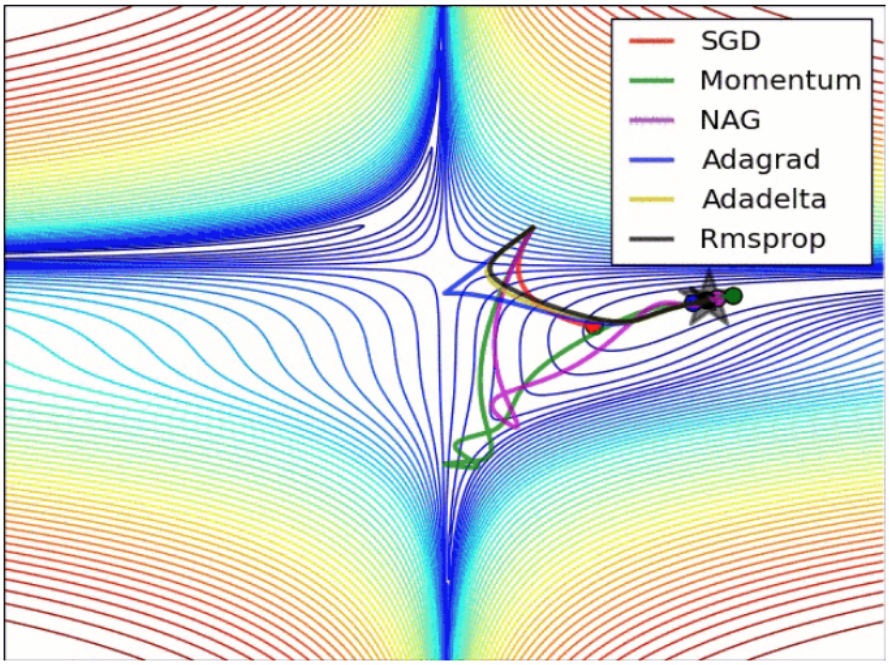


Figure 2.4: SGD optimisation on loss surfaces contours

close to 1). The biases of m_t and v_t are corrected by computing as Equation 2.8 and Equation 2.9 respectively.

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.8}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.9}$$

then the computed m_t and v_t are used to update the parameters also known as *Adam update rule* Equation 2.10:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t \tag{2.10}$$

The proposed default values of β_1 is 0.9 , β_2 is 0.999 and for ϵ is 10^{-8}. The researchers have proven that *Adam* practically works well.

2.6.2.1 Visualisation of Algorithms

The optimisation behaviour of the optimisation algorithms were shown in Figure 2.4. It shows that Adam converges very fast and that too in the right direction of parameters update whereas moment and NAG, SGD is very slow.

2.7 Activation Functions

The activation function of a neuron or a node in the neural network defines neuron output for given input data. The output of a neural network maps to 0 , 1, -1 , 1 or 0, max, which were shown in Figure 2.5. Based on the input data, the respective non-linear activation function will be used in the model.

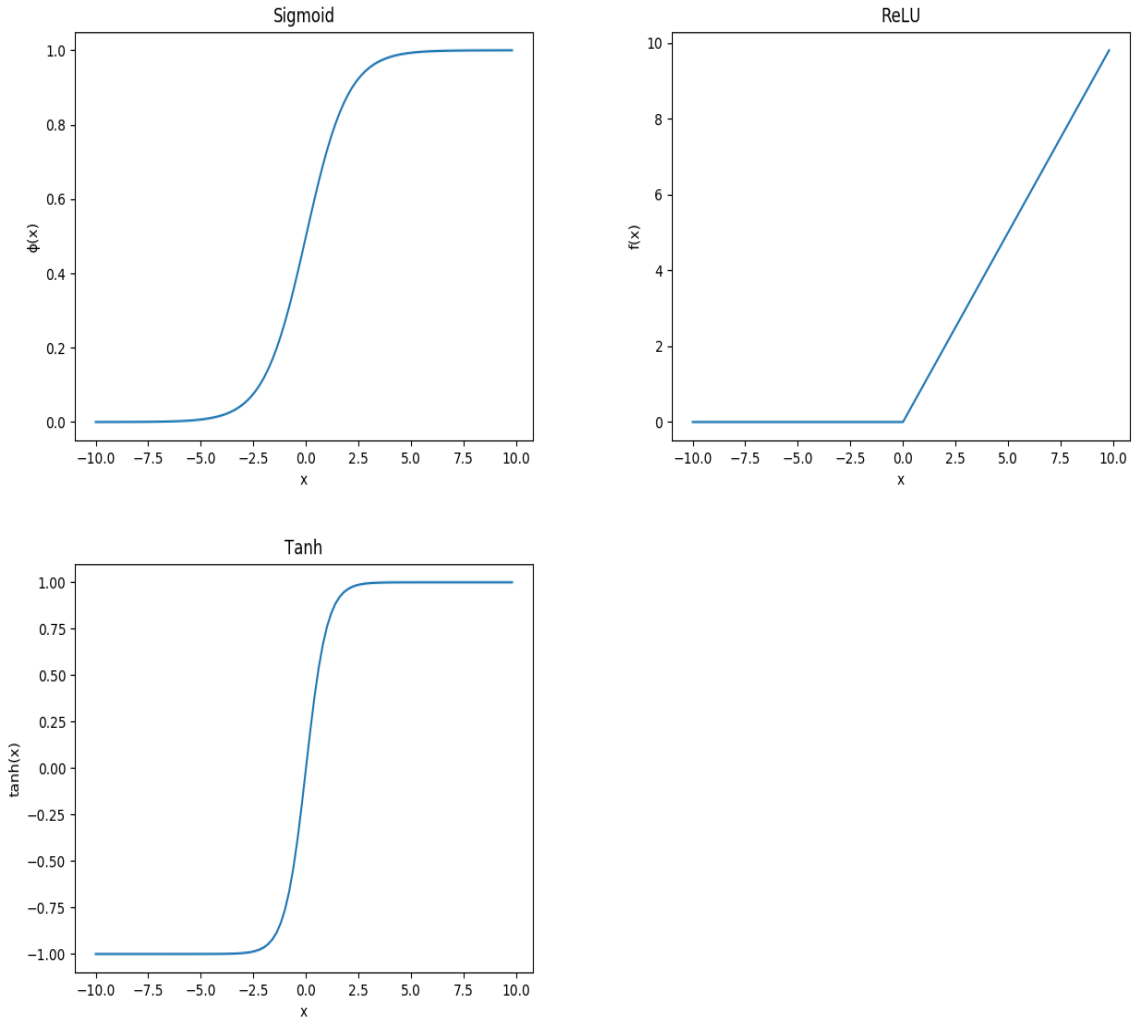


Figure 2.5: Different types of activation functions

1. **Sigmoid activation function:** It is known as logistic activation function Equation 2.11. This activation function squashes real-valued input data between 0 to 1

$$\phi(x) = \frac{1}{1 + e^{-x}} \tag{2.11}$$

2. **ReLU:** Rectified linear unit(ReLU) takes real values input data and replaces negative values with zero from the Equation 2.12.

$$f(x) = \max(0, x) \begin{cases} 0 \text{ for } x < 0 \\ x \text{ for } x \geq 0 \end{cases} \tag{2.12}$$

3. **Tanh activation function:** Also known as hyperbolic tangent activation function refer Equation 2.13. The real-valued input data squashed by the activation into a range of -1 to 1

$$\tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{2.13}$$

Figure 2.6: A living room with a couch a table and a television

2.8 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a family of neural networks, which have demonstrated promising result in the field of image recognition, classification, identification of objects, self-driving cars and in natural language processing tasks.

Figure 2.6[9] **Caption recommendation for a scene:** A convolutional neural network can recognise objects in the image and is able to summarise its content to recommend a caption for the scene.

The current section will cover CNN working principle on images and text related tasks with four main operations. These are fundamental building blocks in a CNN.

1. Convolution operation

[9]https://cs.stanford.edu/people/karpathy/neuraltalk2/demo.html

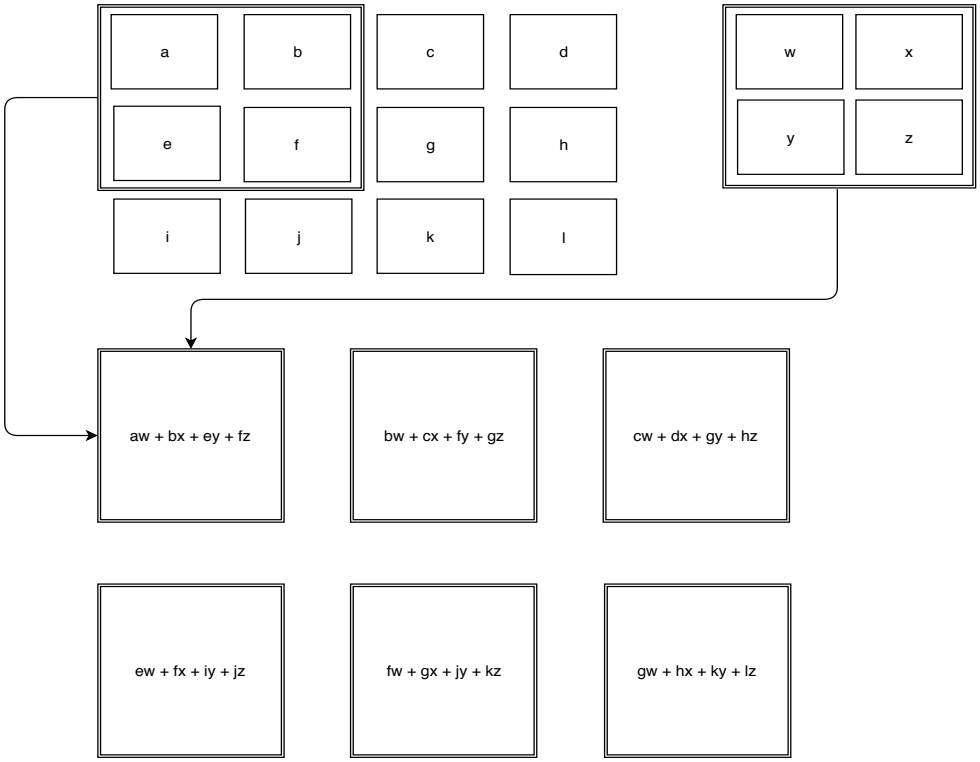


Figure 2.7: Feature extraction with kernel from the input

2. Rectified Linear Units(ReLU)

3. Sub-sampling or pooling

4. Fully connected layer

1. **Convolution operation:** The convolution operator is used to extract essential features from the input image. Convolution is also known to be a specific type of linear operation. It preserves the spatial information of the pixels by learning image features. For instance, every image is considered as a matrix with pixel values. It can be represented as a greyscale image, i.e. single channel or three channel image with red, blue or green, i.e. each colour for one channel. Hence, each channel is a 2d matrix with pixel values. In the convolution step, feature detector, i.e kernel will be applied on input data to get feature map (or convolved feature). A feature map also called an activation map, and both terms can be used interchangeably. The kernel is initialised randomly with a 0's and 1's to extract essential features from the input. The network will not learn any features if the kernel was initialised with all zero's. Most common size of the kernel is 3x3, 5x5 or 7x7. For example, input data with a 3x4 matrix and a kernel with a 2x2 matrix, which was shown in Figure 2.7[10] By sliding kernel by 1 pixel (also known as stride) on the input from left to right, it will compute feature map that is an sum of element-wise matrix multiplication of input and kernel results 2x3 feature map. Also, Convolution operation

[10]https://www.deeplearningbook.org/contents/convnets.html

stores the local dependencies. More complex features can be extracted depending upon the number of filters and their size. Essentially, feature map size depends on depth, stride and padding.

- **Depth:** It is the number of filters used in the convolution operation.
- **Stride:** The number of pixels, by which filter slides over the input data.
- **Zero-padding:** In order to control the feature map size, zero will be added around the input matrix border. It is a hyperparameter of CNN. The performance can be improved by applying "padding" in the network.

The significance of a convolutional layer can be understood with an example such as identifying a person through his/her face at any given moment. While recognising a face, our brain tends to extract visual features namely eyes, ears, nose and skin colour etc. Also, it leaves unnecessary information. Hence, our brain can conclude with essential features. If the brain has features including useless information, then it has process each piece of information to make a decision. That results in the mental break down. In a similar way, a convolutional operation can be performed in CNN with multiple kernels to develop several activation maps. These are known as convolutional layers.

2. **Rectified Linear Units(ReLU):** ReLU is a non-linear activation function. Since most of the real-world data is composed of non-linear patterns, ReLU is applied after convolution operation. Therefore, non-linearity will be introduced in CNN. Besides ReLU, there exist other activation functions such as tanh, sigmoid and softmax. However, ReLU has shown better performance than the other activation functions. ReLU output is defined as the following Equation 2.14. where x is element from the feature map.

$$f(x) = \max(0, x) \tag{2.14}$$

3. **Sub-sampling or pooling:** Downsampling is applied after the application of ReLU on the feature map to reduce feature map dimensionality and to control overfitting. Commonly, pooling is a size of a 2x2 matrix along with the stride length same as pooling. It yields in a maximum number of every sub-region the filter moves around shown in Figure 2.8[11]. Other than max-pool, average pooling and L2 norm poling are also available.

4. **Fully connected layer:** It is also known as multilayer perceptron[12], and it uses a softmax activation function to categorise the final output.

2.9 Recurrent Neural Networks

In natural language processing, the order of the words is essential for better classification of the data. Usually, traditional neural networks would not follow the

[11] https://www.learnopencv.com/image-classification-using-convolutional-neural-networks-in-keras/

[12] https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/

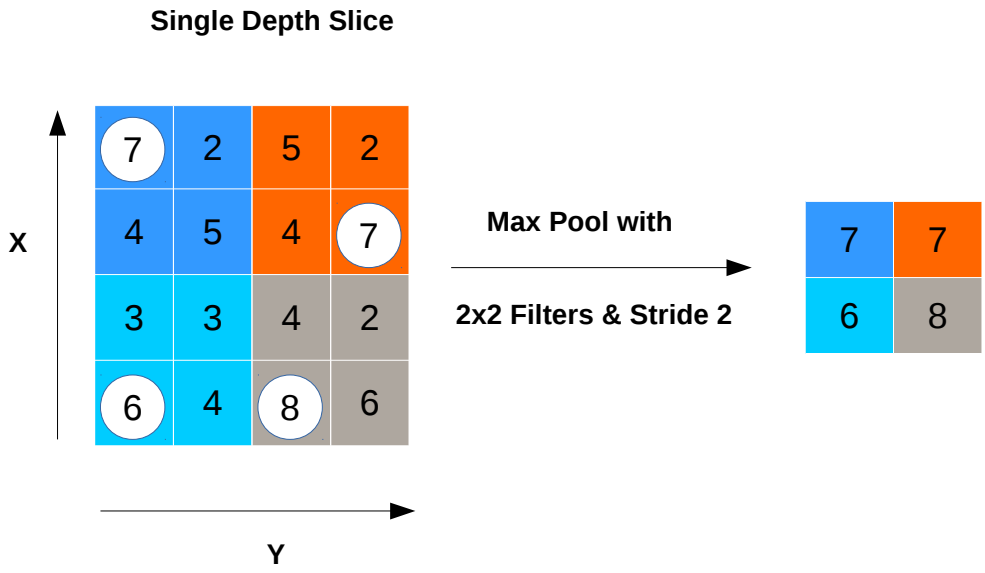


Figure 2.8: Feature extraction after pooling

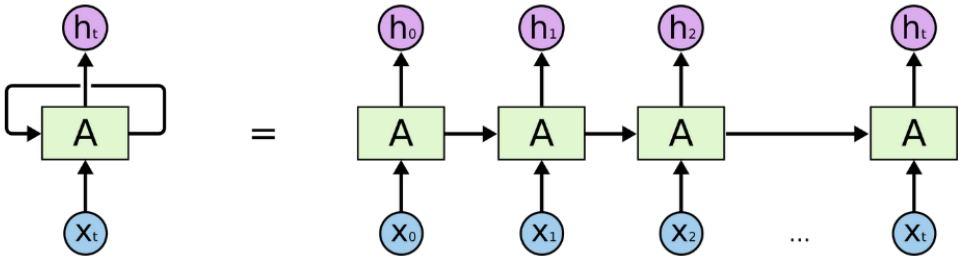


Figure 2.9: An unrolled recurrent neural networks

order of the words. For example, consider a movie for classifying the events which were happened in it. By using traditional neural networks, it is hard to estimate events, which have happened in the previous time-step to predict later events in the film. Recurrent neural networks can handle these kinds of tasks in everyday life. RNN was applied successfully to language translations, speech recognition for image captioning and more.

From the Figure 2.9, a neural network A(in the image from the left side) takes an input x_t and gives an output h_t. The network A allows a piece of information for each time-step through its network and produces a corresponding hidden state output. This process will take place until the complete information passes through the network. It is like a chain process and represents as a sequence and list.

However, recurrent neural networks are limited by the vanishing and exploding gradient problem. It happens because of neural networks use back-propagation to update the weights of the network for each time-step with gradient descent, i.e. flowing backwards in time. A standard network does not have this issue, but in

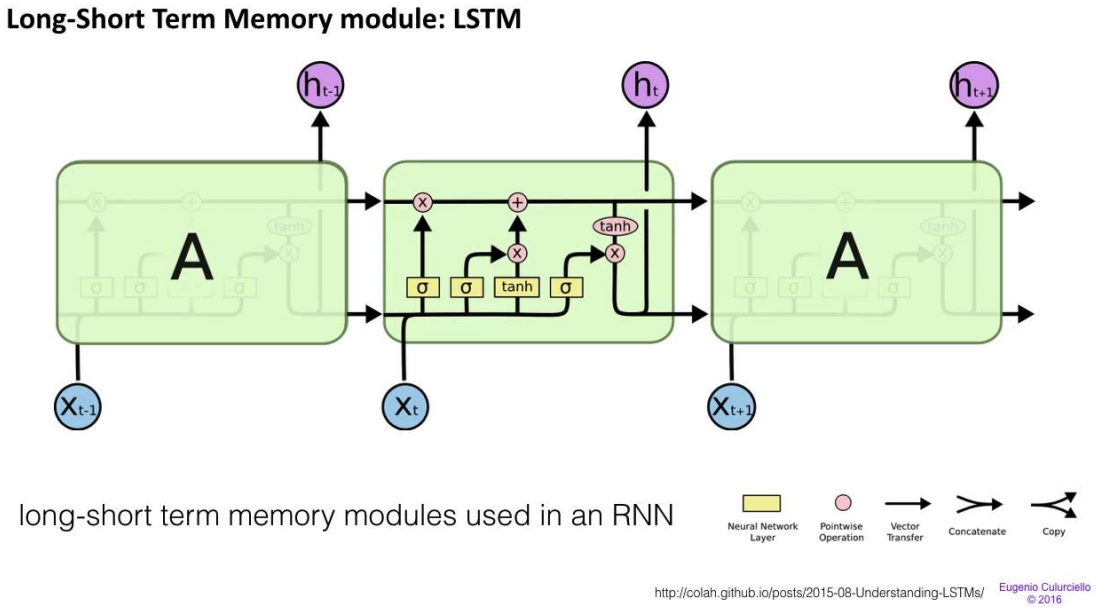


Figure 2.10: The memory module in LSTM with four interacting gates

deep learning, networks have the exploding and vanishing gradient problem. Especially in RNN, the hidden state activation at each time-step depends on its previous time-step. Hence, RNN has to update weights(W) to all its time-steps to minimise error value. They are known as long term dependencies.

$$If\ |W| \begin{cases} < 1\ \textit{Vanishing gradients} \\ > 1\ \textit{Exploding gradients} \end{cases} \tag{2.15}$$

In order to prevent the vanishing gradient problem in neural networks, long short term memory neural networks were invented[25].

2.9.1 Long-Short Term Memory Network

LSTM is an enhanced version of RNN and capable of learning long term dependencies. For every time step, LSTM has a cell which is referred to as the ”Memory” of the cell. LSTM can remember information for long periods. The standard RNN has a simple structure, like a chain of repeating modules with an activation function layer, i.e tanh activation function. Also, LSTM has the same chain-like structure, but repeating module has a different structure with four gates. These gates interact uniquely.

In the above Figure 2.10, the cell state is like a conveyer belt, the horizontal line, at the top of the cell runs straight down the entire chain with linear interaction. The information without a change in it will flow through the line. LSTM cannot manipulate the information in the cell state, but gates can control and can add information to the cell. These gates are with a sigmoid activation function and a pointwise multiplication operation. In the sigmoid activation function, the flow of

information is controlled through the gate. 0: means let nothing through, 1: let everything through.

LSTM has three gates to protect and control the cell states. A Step-by-step LSTM walkthrough:

- ***Step 1:*** LSTM decides which information needs to avoid the cell state. The ”Forget gate” with sigmoid activation function will regulate the information, 1: completely keep this, 0: means completely get rid of this.
- ***Step 2:*** is to decide about new information will store in the cell state. At these two sub decisions will be made by the gates. First, the input gate, i.e. with sigmoid function decides which values to update. Second, a vector of new candidate values will be created by the tanh layer; these will be added to the state. In the following step, is a combination of these two for an update to the state.
- ***Step 3:*** is an update of an old cell state into a new cell state. The previous steps will decide what kind of information to update. By multiplying old cell state with a forget gate, yields what to forget earlier then, by adding a new input to update ”vector values” results in value to update cell state.
- ***Step 4:*** is for getting an output. The output is based on cell state after an update to the cell state. With a sigmoid layer, outputs the cell state parts. Then the output of the cell state flow through tanh, i.e. the output between values -1 to 1 and multiply the output with the sigmoid gate output.

There exist different types of LSTM. One popular LSTM variant introduced Gers[13] is adding ”peephole connection” which means let gates look at the cell state.

2.10 Pre-trained Glove Embeddings

Using word embeddings is an alternative approach for text processing. These word embeddings can be created by using word2vec algorithms. For any given NLP task, creating own word embeddings may take a long time. It requires more resources such as a computer with high ram and more disk space. Instead of the creation of own embeddings, one can use pre-trained word embeddings, which are published by the Standford University and Facebook etc.

The global vectors for the word representation (GloVe) are trained on Wikipedia data, which contain 6 Billion tokens and 400,000 vocabularies in the model. These models are of 50, 100, 200 and 300- dimensional vectors[26].

The Facebook has also developed different kind of word2vec models[14] for 294 languages with 300-dimensional vectors. These embeddings are trained on Wikiedia data using FastText[15] library[27].

In this thesis, I will focus on three important languages namely, English, German, French and mix of the three languages.

[13] ftp://ftp.idsia.ch/pub/juergen/TimeCount-IJCNN2000.pdf
[14] https://fasttext.cc/docs/en/crawl-vectors.html
[15] https://fasttext.cc/

2.11 Evaluation of Classifier

The following section describes several key concepts and metrics, which are used to evaluate the quality of the neural net classifiers. The quality concepts are the properties of the neural network. These properties will be evaluated by set metrics, which define the extent of properties whether they present or absent. Most common quality concepts are precision, accuracy, recall and F1-Score. These will define how much closer the sample is to the real value, by knowing what is wrong can help to improve the classifier performance.

The quality of metrics is evaluated based on the contingency table, i.e. famously known as confusion matrix. These metrics define the extent of a specific property contained by the classifier. Let us take a confusion matrix an example,

Table 2.2: Confusion matrix

Total number of samples= 120	Actual pos	Actual non-pos
Predicted pos	52	8
Predicted non-pos	12	48

- **True Positives:** The predicted cases which are pos and actual cases were also pos
- **True Negatives:** The predicted cases, which are non-pos and actual cases were also non-pos
- **False Positives:** The predicted cases, which are pos but actual cases were non-pos
- **False Negative:** The predicted cases, which are non-pos but actual cases were pos

Accuracy: It is a performance measure for a classifier based on correctly predicted samples to the total samples. From the Equation 2.16

$$Accuracy = \frac{True\ Positives + True\ Negatives}{True\ Positives + True\ Negatives + False\ Positives + False\ Negatives} \tag{2.16}$$

Precision: If the classifier prediction is same as the sample actual label, then how often it is. From the Equation 2.17

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{2.17}$$

Recall: With respect to actual label of a sample, how often classifier predicting correctly. From the Equation 2.18

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{2.18}$$

F1_Score: It is a measurement for test's accuracy and a harmonic mean between recall and precision. F1 score lies between 0 to 1. Based on F1-Score model performance will be defined precisely. A higher value of F1 defines the better performance of the model. From the Equation 2.19

$$F1\_Score = \frac{2 * Recall * Precision}{Recall + Precision} \tag{2.19}$$

2.12 Summary

This chapter describes the sentiment analysis and its importance. Specifically, the challenges associated with it which are solved by using deep neural networks. The fundamental concepts of deep learning methods namely CNN, LSTM were explained. Furthermore, the necessary data preprocessing steps for sentiment analysis were explained. During training of a deep learning model, an optimisation algorithm plays an essential role in achieving a desirable result. Hence, its importance and different types have been given in the chapter.

Moreover, activation layers were used to introduce non-linearity in the model. Therefore different types of activation functions were also mentioned. Finally, the model evaluation metrics were provided along with their formulas.

3. Related Work

For the past decade, more research was done on the subject of sentiment analysis through machine learning. Along with it, numerous technologies have emerged in the context of deep learning on the basis of different variations in the CNN, RNN and their combinations. Originally, intended propose of CNN is to use for computer vision. However, CNN proven effective for sentiment analysis in the field of natural language processing (NLP). NLP is composed of different applications such as information retrieval, ranking and document classification and sentiment analysis etc.[8]. The Section 3.1 will give you an idea of deep learning uses in different fields. And, Section 3.2 provides differences between my work to the prior work.

3.1 Recent Advances in Deep Learning

Deep learning has become the newest trend in the field of artificial intelligence(AI). Although, there was a variety of research available on deep learning applications (DLAs). One of DLAs is an automated speech recognition system, which has become important in the field of information and communication technology. Anuroop et al.[28] improved speech recognition through LSTM and GRU methods by inducing invariance to noise. Another application of DL is image recognition also known as computer vision. Through image recognition, the goal is to have computers detect objects just like a human does. However, computer vision is far away from a human's ability to recognize objects. To improve this Zheng et al.[29] have introduced a method based on CNN and conditional random fields (CRF). It has the combined strength of individual techniques. They have applied their method on semantic image segmentation. Since deep learning can also be used for NLP, Yang et al.[30] have proposed a hierarchical attention network for document classification with two levels of attention mechanism at the word and sentence level. This model selected informative words and sentences. However, from the work of Zhang et al.[7], character level CNN was implemented for text classification. Also, provided a comparison against TFIDF variants and word level deep learning models using publicly available data. Furthermore, Zhang et al. provided a way to handle non-English languages.

Vosoughi et al.[31] extended the work of Zhang et al. through a combination of CNN-LSTM encoder-decoder for tweets classification. This *Tweet2Vec* model learns tweet embeddings at the characters level. Moreover, this model is adaptable to different languages. Also, we can understand that combined CNN-LSTM can handle short text. However, the above mentioned NLP methods still needed improvements. They depend on high balanced quality labelled data. It is difficult, expensive and time-consuming to prepare such kind of data. To reduce annotation cost, Li et al.[32] proposed an active learning method for imbalanced class distribution. Sandeep et al.[33] applied active learning on the Telugu language for sentiment analysis. They have used word embeddings from Fasttext. Additionally, they investigated the combination of different query selection strategies in order to increase model efficiency. Most of the methods were developed for single use cases such as for single language SA method or only one NLP task-based system. To compensate more than one NLP task, Jeremy Howard and Sebastian Ruder[34] invented a new model universal language model fine-tuning (ULMFiT) for text classification using transfer learning. Furthermore, they have compared their model performance with 100 labelled samples to 100x more data.

In NLP, sometimes the selection of DNN method becomes essential for particular NLP task. From [35], Yin et al. have provided a systematic comparative study of CNN and RNN on seven NLP tasks. Some of the NLP tasks are sentiment classification (SentiC), relation classification (RC), textual entailment (TE), and answer selection (AS) etc. In the case of SentiC task, CNN, GRU and LSTM models have trained on stanford sentiment treebank (SST) dataset. They have provided performances of CNN, LSTM and GRU for different NLP tasks including optimal hyperparameters and their sensitivity. That is a change in model performance when there is a change in hyperparameters such as batch size, learning rate and sentence length etc. Hence, Yin et al. have concluded that the global/long range semantics decides the performance of DNN method for text classification. These guidelines were helpful while selecting a DNN method. Similar to NLP tasks in [35] , Young et al.[3] also covered various methods based on CNN, RNN, recursive neural networks their combinations. Further, Young et al. have provided information regarding reinforcement learning (RL) application in language generation task. For more recent developments in deep learning provided in [36].

3.2 Deep Learning based Sentiment Analysis

Sentence modelling is one of the core steps in NLP application. Zhou et al. developed a unified model using convolutional neural network and long short term memory network (C-LSTM)[37] for sentence level for "sentiment analysis and question classification tasks". In their research, traditional sentence modelling models failed to provide a good result compared to their new method with individual CNN and LSTM models. Even though Zhou et al. were able to propose a better technique, still it can't be used for new data. Each time, it is essential to create sentence level word-embeddings for new data which is a difficult task, costly in terms of time and resources. However, Van et al.[38] have addressed sentence-level sentiment analysis by employing their model with transfer learning to improve word embeddings. Van et al. implemented their method based on CNN and tree LSTM. Also, they

have evaluated with other combined CNN-LSTM architectures. Whereas, Huang et al.[39] have investigated a model for the Chinese language that can use word embeddings by combining CNN and two-layer LSTM for sentiment analysis. Also, Huang et al. have shown that number of LSTM layers could benefit the combination of the model. However, pre-trained word embeddings were challenging to prepare for every language which is a time consuming and unnecessary additional task.

Cieliebak et al.[40] developed a CNN based model using word embeddings for German tweets. Moreover, they aimed to design industry-ready software libraries for the German language. However, CNN models performances can be improved by adding more LSTMs to it[41]. To enhance system ability for mixed language, Deriu et al.[42] paper investigated a novel approach using weakly supervised data trained only on CNN method up to 3 layers. Also, Deriu et al. have evaluated their model performance with different dataset size, including SemEval-2016[1] sentiment prediction benchmark (Task 4). However, model performance can be improved with more number of convolutional layers[6]. Alayba et al.[43] implemented CNN-LSTM model for Arabic tweets. They have used stride values up to 5-gram characters in convolutional layer, i.e. s = 5. But, the described method was used very less data for training.

For European languages, Narr et al.[44] developed sentiment analysis classifiers based on shallow machine learning models only. This model can handle tweets in multiple languages. Further, they analysed the characteristics and usefulness of semi-supervised sentiment classification. However, recent improvements in deep learning especially the combination of CNN and LSTM techniques had a higher accuracy than shallow models.

[1] http://alt.qcri.org/semeval2016/

4. Implementation

In Chapter 2, I have discussed the importance of sentiment analysis (SA), challenges of SA, and techniques to approach SA. The SA challenges are stated as: data cleaning, multilanguage support and limited resources. To address the above problems, Pytorch was used, which is a deep learning framework. The traditional machine learning models were required data preprocessing steps and cannot handle multilanguage input. In this thesis, English, German and French languages were used including a method to overcome the multilanguage sentiment analysis problem. In Section 4.2, a step by step approach of the method was explained such as data manipulation and implementation of the proposed method, i.e. model workflow. Also, some other sentiment analysis systems were described.

4.1 Problem Definition

For the past few years, sentiment classification has drawn much attention both in industry and academia. However, accurate multilingual sentiment analysis models have become difficult to implement due to the lack of high quality labelled data. Most of the existing approaches were used transfer learning[1] from English. Also, most of the research focuses on one language (most often English). For English language, labelled data and the resources are sufficient, but for other languages, resources are much more limited. To use an English classifier, sometimes a target language, i.e. a foreign language was translated into the English language. Moreover, state-of-the-art methods, i.e. individual techniques were failed to provide excellent results in this specific use-case.

Hence, a new sentiment analysis method is needed to overcome the problems mentioned above by combining individual deep learning techniques such as CNN and LSTM. "The SA through CNN and LSTM" methodology will be explained in Section 4.2. However, individual techniques have their weaknesses, and they drop their performance on single language and on multilanguage. Hence, the CNN-LSTM SA system is required to evaluate on the three crucial aspects such as comparing against

[1] a method that uses a model which was trained on one task and used it for another task.

other individual state-of-the-art methods, by increasing layer depth and performance evaluation of the multilanguage model. They will be answered based on the metrics precision, recall, F1_scores and accuracies on English, German, French and on the multilanguage datasets in Section 5.2.3

4.2 Methodology

In this work, a combined model will be implemented for sentiment analysis based on CNN, LSTM and character-level approach.

As an initial step, input data is loaded into pandas dataframe. It is also an important step to maintain an equal distribution of the input data and hence the input data was manipulated. Each partially preprocessed input (i.e. review) was converted into a 2-dimensional array. Since the machine can understand data in numbers, the 2D array was composed of one and zero (also known as one-hot-encoding).

Furthermore, CNN techniques cannot handle data in variable length(i.e review length). Therefore the review length was fixed to 1024 characters including space and special characters. Any review which has a length lesser than 1024, will be padded with all zero's until it is equal to a fixed review length. In one-hot-encoding of a 2D, all available alphabets were considered as features and each character in review was placed as an index (i.e row). Hence, each character from the review was represented with 85-dimensional character embeddings (i.e equal to a total number of alphabets including special characters). The structure of the model and the data flow were described in the Section 4.3. The previous studies[6][7] stated that a sentence length with 1024 characters could represent semantic meaning of the whole sentence.

Once the data has been prepared, then it has to be sent to the model. In the combined model, convolutional layers have stacked over the LSTM layers. Therefore, the first convolutional layer receives the 2D array input data. CNN can be composed of variable layer depth. In my research, the 8 convolutional layers based on the description of the [6]. Having more number of convolutional layers could be useful for extracting all relevant features without missing them. After that, those features have been increased to 512 dimensions before giving them as an input to the LSTM layer. The LSTM layer can remember all dependencies among features and it can process all those features in a sequential order. Hence, it can learn about alphabets and recognise them one after the another based on probability values. Also, from the gating mechanism of the LSTM technique, it can leave out unnecessary information which does not contribute to the sentence classification.

Nevertheless, real world data always have nonlinear patterns. Therefore, the activation layers (i.e ReLU) are used in the model to induce nonlinearity into the network so that the model can learn nonlinear patterns existed in the input data. Before applying an activation layer in the network, a batch normalisation layer will be introduced to speed up the training. Otherwise, the network has to adapt immediately to new changes which are made by the each input batch. Hence, batch normalisation transforms inputs from every batch into a mean (zero) and unit variance. It helps to keep input distribution same for every batch and reduces covariance shift for each

activation layer. It is thereby ensuring that each layer with more stable distribution of inputs.

The feature dimensions were reduced to 128 by the LSTM layer. The high number of dimensions cause to generate more parameters. Thus an increase in more neurons in a fully connected layer can cause a memory error. The fully connected layers in the method was used for classification of the text sentiment. However, dropout layer with a value 0.5 can also be used between two fully connected layers. It is a regularisation technique to avoid overfitting of the neural network. Dropout randomly selects neurons in fully connected layer and then ignores them during the training of the model.

Therefore, the combination of the two techniques could overcome potential drawbacks of individual techniques for sentiment analysis. For this specific case, English, German, French and a mixed dataset of these three languages will be used. The model parameters and layer depth were selected based on the [6]. Further, hyperparameters were selected after careful experimentation with the different combinations of each hyperparameter. I will be using Pytorch as a deep learning framework which allows to do similar computations as with numpy[2], but more efficiently in respect to GPU usage. Further, *a performance comparison* between the CNN-LSTM model and the individual CNN and LSTM models will be made. Additionally, the CNN-LSTM model will be compared with the baseline models such as support vector machines (SVM) and decision tree (DT) methods. Also, multilanguage model will be compared with single language models and a performance study will be made based on increasing the model layer depth.

4.3 Model Workflow

The proposed model architecture is shown in Figure 4.1. In the model, the first layer contains a lookup table with s character embeddings that can generate a 2D tensor of size(f_0, s). For input,s has a fixed length of 1024 characters and f_0 is an "RGB" channel. However, the input text is a single dimension, namely the Grey channel. Therefore input to the CNN first layer, i.e. lookup table was prepared as 16 vectorized dimensions for 85 features, which were one hot encoding (i.e 1's and 0's).

The temporal convolutional block is shown in Figure 4.1, in which each convolutional layer has X feature maps, kernel of size 3, padding of length 1. The second convolutional layer has 64 dimensions and a kernel size of 3 which is then followed by a stack of temporal convolutional blocks. From Conneau et al.[6], each convolutional block has the same number of features and also the same temporal resolutions. Hence, the proposed method has three levels of feature maps produced by three pooling operations. Between each convolutional block, the temporal max-pooling layer has kernel size 3 (i.e k_3) and stride 2 (i.e s_2). The final layer of the CNN generates 512 vectorised dimensions for 85 features. Then, they were fed to a single layer of LSTM which has one hidden state. The LSTM outputs with 128 dimensions after learning sequential order from the input embeddings.

[2]http://www.scipy-lectures.org/

Hence, the entire network can produce tensors at higher level representation for the fixed length input text in 3D. The 3D tensor output then converted into 2D tensor to feed fully connected layer with 65536 neurons. It is represented as *fc_layer(I, O)*, which is a *matrix of size(I X O)*. The final layer of the method produces a sentiment of the input text whether it is a positive (1) or negative (0) in binary digits.

4.4 System Details

The thesis source code written in Python using deep learning framework Pytorch[45] exploiting GPU acceleration with the CuDNN library[46]. The learning procedure takes approximately 2- 48 hours for deep learning model and 24 hours to 1 week for traditional machine learning model with 48k reviews. Experiments were conducted on a computer with GEFORCE GTX 1070 GPU having 1920 CUDA cores and 8 GB of RAM and instances of *Amazon Web Services(AWS)* with a1.xlarge 4 vCPU 8GB of RAM.

4.5 Sentiment Analysis Systems

In the thesis, following sentiment analysis systems were implemented:

1. **Bag-of-words and its TFIDF**

   The bag-of-words TFIDF model was developed for each dataset by considering 50,000 most frequent words from each training dataset. For the term-frequency inverse-document-frequency (TFIDF), uses the counts as the term-frequency. The inverse document frequency is the logarithm of the division between the total number of samples and number of samples with the word in the training dataset. The features are normalized by dividing the largest feature value[47].

2. **Decision Tree with n-gram**

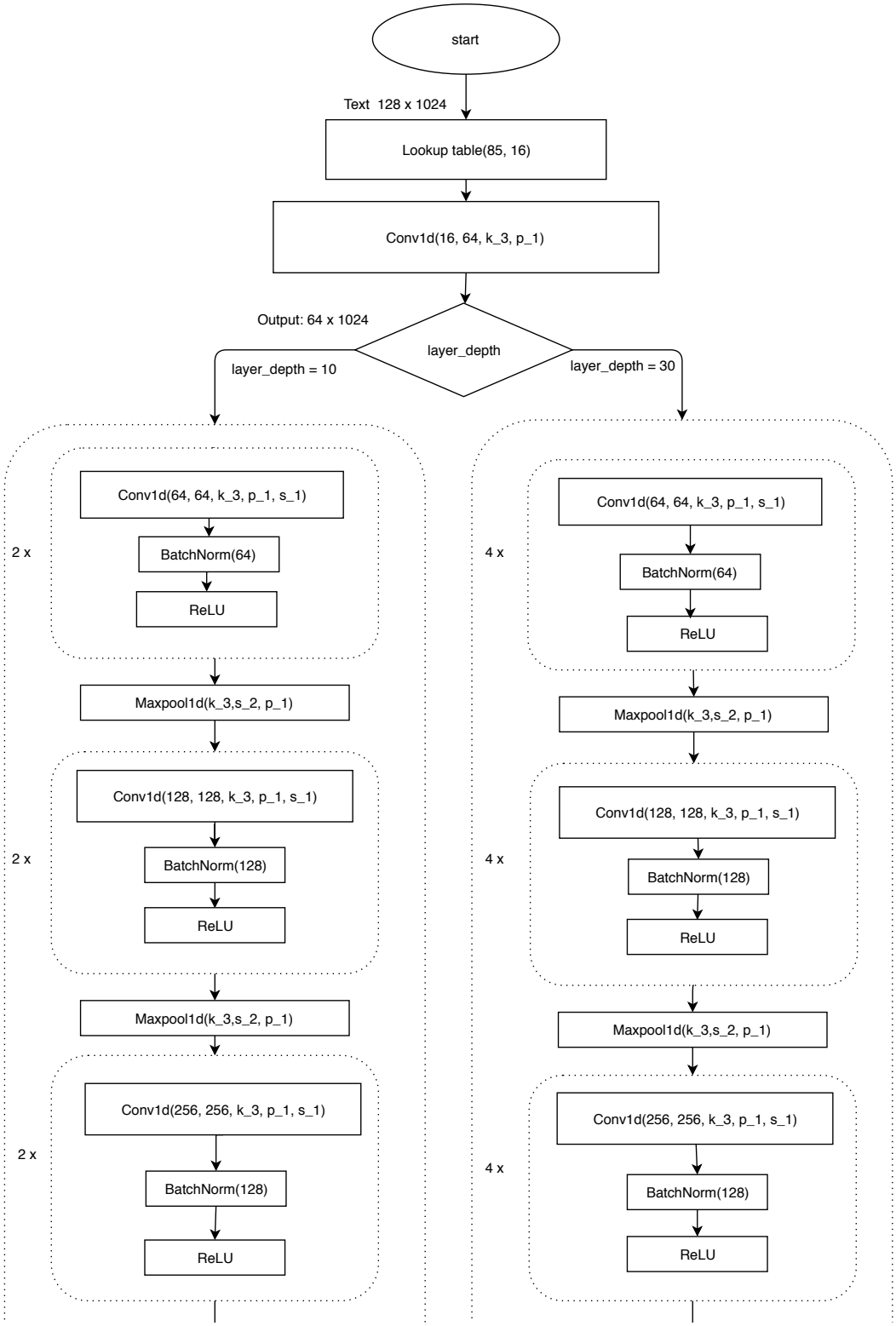
   The decision tree (DT) n-gram model is constructed by selecting possible unigram and bi-gram features for each training dataset. The DT model can predict a target with the simple decision rules which are learned by understating features.

3. **6 conv char-level CNN**

   The ConvNets designed with the six convolutional layers, two fully connected layers and with a one dropout layer having 0.5 probability. As per the description of Zhang et al.[7], the input length is fixed to 1014 characters.

4. **char-level LSTM**

   Despite knowing char level LSTM model slow at updating their weight values than word level LSTM, a unidirectional long short term memory(LSTM) model was designed to check its performance[5]. It gets input features as *one-hot encoading* embeddings. The output is a mean of all LSTM cells with a dimension of 32.


start
Text 128 x 1024
Lookup table(85, 16)
Conv1d(16, 64, k_3, p_1)
Output: 64 x 1024
layer_depth
layer_depth = 10
layer_depth = 30
2 x
Conv1d(64, 64, k_3, p_1, s_1)
BatchNorm(64)
ReLU
Maxpool1d(k_3,s_2, p_1)
2 x
Conv1d(128, 128, k_3, p_1, s_1)
BatchNorm(128)
ReLU
Maxpool1d(k_3,s_2, p_1)
2 x
Conv1d(256, 256, k_3, p_1, s_1)
BatchNorm(128)
ReLU
4 x
Conv1d(64, 64, k_3, p_1, s_1)
BatchNorm(64)
ReLU
Maxpool1d(k_3,s_2, p_1)
4 x
Conv1d(128, 128, k_3, p_1, s_1)
BatchNorm(128)
ReLU
Maxpool1d(k_3,s_2, p_1)
4 x
Conv1d(256, 256, k_3, p_1, s_1)
BatchNorm(128)
ReLU

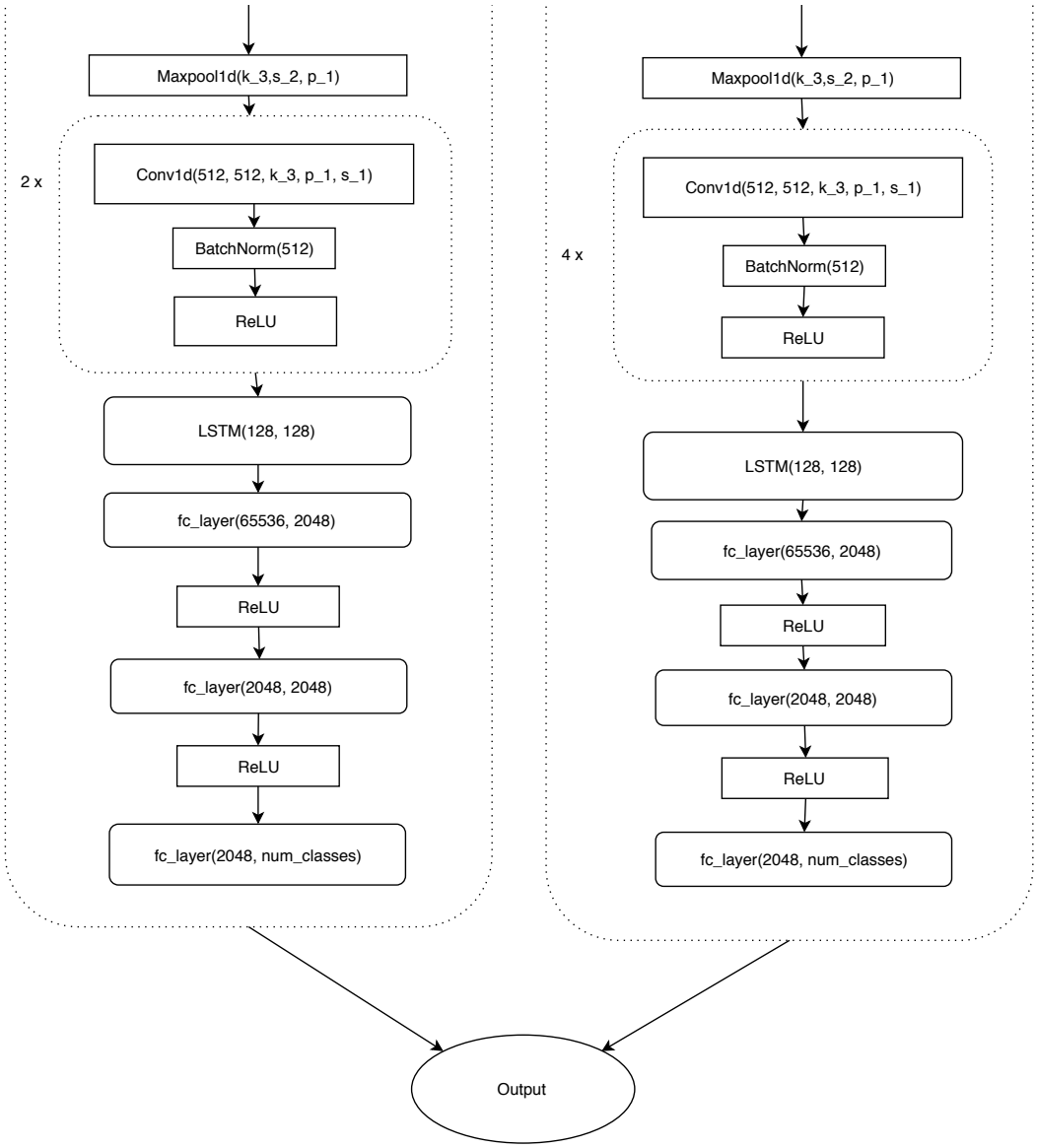


Figure 4.1: Flowchart

5. **char-level VDCNN**

   The very deep ConvNets was implemented based on the description of Conneau et al.[6] from the Facebook research group. It has a fixed input sequence with a length s of 1024 with a depth of 9 and 29 convolutional layers, with an initial character embeddings size 16. This network is having three pooling operations, which are resulting three levels of 128, 256 and 512 feature maps for a stack of temporal convolutional blocks respectively. The final temporal convolutional block output is with a tensor of size $512 \times s_d$ where s_d is $s_d = \frac{s}{2^p}$ with $p = 3$, i.e. a number of downsampling.

6. **char-level Stacked-LSTM**

   The multi-layer structure of LSTM can process new information at each layer. One layer may not capture all the features. If it has more than one layer, the layers can extract all features from the text. Each layer receives an input from the output of a hidden state in previous layer. For final output is processed through a softmax layer for prediction of the target. This type of architecture is a combination of deep neural networks and recurrent neural networks, which in turn creates stacked LSTM[48]. In char-level stacked LSTM, it has 2 LSTM layers; each layer has one hidden state, two fully connected layers and without a dropout layer.

7. **Proposed method**

   Based on the proposed method, I have implemented two models in which one model with ten layer depth and another model with 30 layer depth excluding fully connected layers based on Conneau et al.[6] experimental results. They showed that an increase in layer depth could benefit model performance. The 10-layer model settings are as follows:

   - Intial character embedding size is 16 for first convolutional layer.
   - Adam optimiser with a learning rate of 0.001.
   - The maximum length of the input stream of characters fixed to 1024 with a minibatch size of 128.
   - The number of epochs (known as the number of re-training) set to 20. If the model doesn't improve for each epoch then it will stop automatically for six epochs.
   - The first convolutional block is composed of 2 ConvNets with input dimensions of 64, output dimensions also with 64; kernel size is 3; padding is 1.
   - The second, third and fourth convolutional blocks are also consisting of 2 ConvNets with input dimensions of 128,256 and 512; output dimensions are same as corresponding convolutional block input dimensions; kernel size is 3 and padding is 1.
   - Each convolutional block uses an activation layer(ReLU) and a batch-normalisation of size same as output dimensions in ConvNets.
   - Each max-pooling layer with kernel size is 3; stride is 2 and padding is 1.

- Single LSTM layer is used.
- Hidden-size dimension of the LSTM layer is 128.
- The number of neurons in the first fully connected layer is 2048.

In the case of 30 layer model, it uses 4 times each convolutional block in order to increase its neural network depth.

4.6 Summary

From the implementation chapter, we have seen the problem definition for sentiment analysis and its importance in real world application especially for multilanguage. Also, method implementation details were shown step by step including the importance of each layer. Further, additional models for comparison including their parameters and hyperparameters which were selected based on previous studies. However, the baseline model (i.e SVM, DT) parameters were selected after parameter tuning. The hyperparameters of the proposed method were selected after several experiments which are mentioned in Section 5.2.3. Furthermore, the used resources for training were mentioned in this chapter as well.

5. Evaluation

In this chapter, I shall establish with the three important research questions that they will be tackled in the study. Subsequently, I shall be describing an analysis of the performance of the proposed models per language and per model. Also, the increasing depth of network architecture will show whether it affected the proposed models performance and on individual models at different configurations. Unfortunately, no scientific research uses the same datasets (Section 5.2.1) to compare the performance of the proposed models. Hence, in this thesis, *svm_tfidf* and *dt_ngram* models were implemented and were considered as baseline methods for performance evaluation of the deep learning models. Also, I have implemented and evaluated state-of-the-art methods namely: very deep CNN (i.e char-level VDCNN)[6], 6 conv char-level CNN[7] and stacked-LSTM[48]. However, stacked-LSTM was limited to 2 LSTM layers because it was computationally intensive and time-consuming.

5.1 Research Questions

Artificial neural networks, i.e. deep learning technique performances depend on the hyperparameters, random initial weights and shuffling of the data during each training epoch. Further, deep learning is known for its performance which is highly dependent on the data used. Additionally, evidence was established from my literature study that it is possible to build a reliable method through a combination of CNN and LSTM. Also, a methodology (Section 4.2) was provided for building a combined model from CNN and LSTM. Due to the lack of accurate SA system and limited resources in the foreign language, optimization of the method is required to achieve the best performance. In order to identify its best performance , I want to answer the following research questions in this study through the result of all sentiment analysis systems (Section 4.5), which I shall achieve on the test set.

1. Does the proposed model outperform the individual CNN model, LSTM model and the baseline models such as support vector machine (SVM) and decision tree (DT)? (Section 5.2.3.1)

2. How does the model behave with increase in layer depth? (Section 5.2.3.2)

3. What will happen if the proposed method fed with multiple languages at the same time? (Section 5.2.3.3)

5.2 Experiment setup

5.2.1 Datasets

Language	Label	Sample
English	Positive	Like crackin open a cold beer. This is hip hop ya'll. This what needs to be on the airwaves. This whole album is listenable. No poppy fluff here. Just a dope MC paintin pictures with words and not videos. How dudes like JOC D4L and franchize boys sell records is still beyond me. Support real hip hop and buy this. You wont regret it
German	Negative	wenn man sich die anderen Popstars-Gruppen anguckt... kein Vergleich... wartet auf die Alben der TvTotal-Castingshow da bekommt ihr Qualität. das hier ist total öde!
French	Negative	Ces trois là s'improvisent donneurs de leçon dans le Jazz Magazine du mois de mai, or leur prestation sur cette album m'a laissé sur ma faim

Table 5.1: Text samples per language and their labels

The large dataset is composed of 800.000 *Amazon product reviews* in four different languages: *English, German, French and Japanese* for the three product categories such as books, dvds and music. They were crawled in the month of November 2009 in the XML format[49]. Each product review was labelled among 1 to 5. A subset from a large dataset, in which ratings one and two considered as a negative review, four, five considered as a positive review and rating three was ignored. The subset from a large dataset is around 48k *Amazon product reviews* for the three languages: *English, German, and French.* An example of each language sample was shown in Table 5.1. Each deep learning model takes an input of a sequence of encoded characters. The input language was encoded by defining an alphabet with a size of m (i.e 85) and then quantise each character using *1-of-m* encoding or *one-hot* encoding. After that, the sequence of characters will transform into a sequence of such m sized vectors with fixed length s. If the length of characters exceeds s (i.e 1024) and also, characters which do appear in the alphabet, were considered as all-zero vectors. The backward quantisation order was considered for the characters, i.e. fully connected layers always fed by the updated latest weights.

The alphabets used for the deep learning models are composed of 85 characters, including 26 English letters, twelve special French letters, four German special charac-

Language	Dataset	Total	Pos.	Neg.
English	Training	47820	23910	23910
	Validation	6000	3000	3000
	Test	6000	3000	3000
German	Training	47804	23896	23908
	Validation	6000	3000	3000
	Test	6000	3000	3000
French	Training	41236	20618	20618
	Validation	6000	3000	3000
	Test	6000	3000	3000
M_type_1	Training	109488	54682	54806
	Validation	-	-	-
	Test	-	-	-
M_type_2	Training	48000	24000	24000
	Validation	6000	3000	3000
	Test	6000	3000	3000

Table 5.2: Data used for training models

ters, ten digits, 32 other characters and the new line character. The list of non-space characters are:

éàèùâêîôûçëïabcdefghijklmnopqrstuvwxyzäüöß0123456789

,;.!?:'/" |_\@#$%ˆ€¿+'-=<>()[]{}*

An overview of the labelled datasets, which were used in the experiment, is shown in Table 5.2.

Data Preparation:

Each product category review file was processed as following steps:

- Each *XML* file has been converted into *.csv* file.
- The text was changed into lowercase.
- Following datapreprocessing steps were applied to the traditional machine learning model
    1. *Word tokenisation:* The text was tokenised into a list of array of words.
    2. *Stopword eliminations:* Stopwords are frequently occurring words and they doesn't contain any meaningful information in the text. Hence, they were removed from the text.
    3. *Lemmatization:* Lemmatization was applied on a word to reduce its ambiguity in the context (i.e mapping a word to their common lemma). Also, lemma is dictionary entry or lemmatisation lexicon.
    4. *Stemming:* Stemming was applied to reduce different inflections on the words.

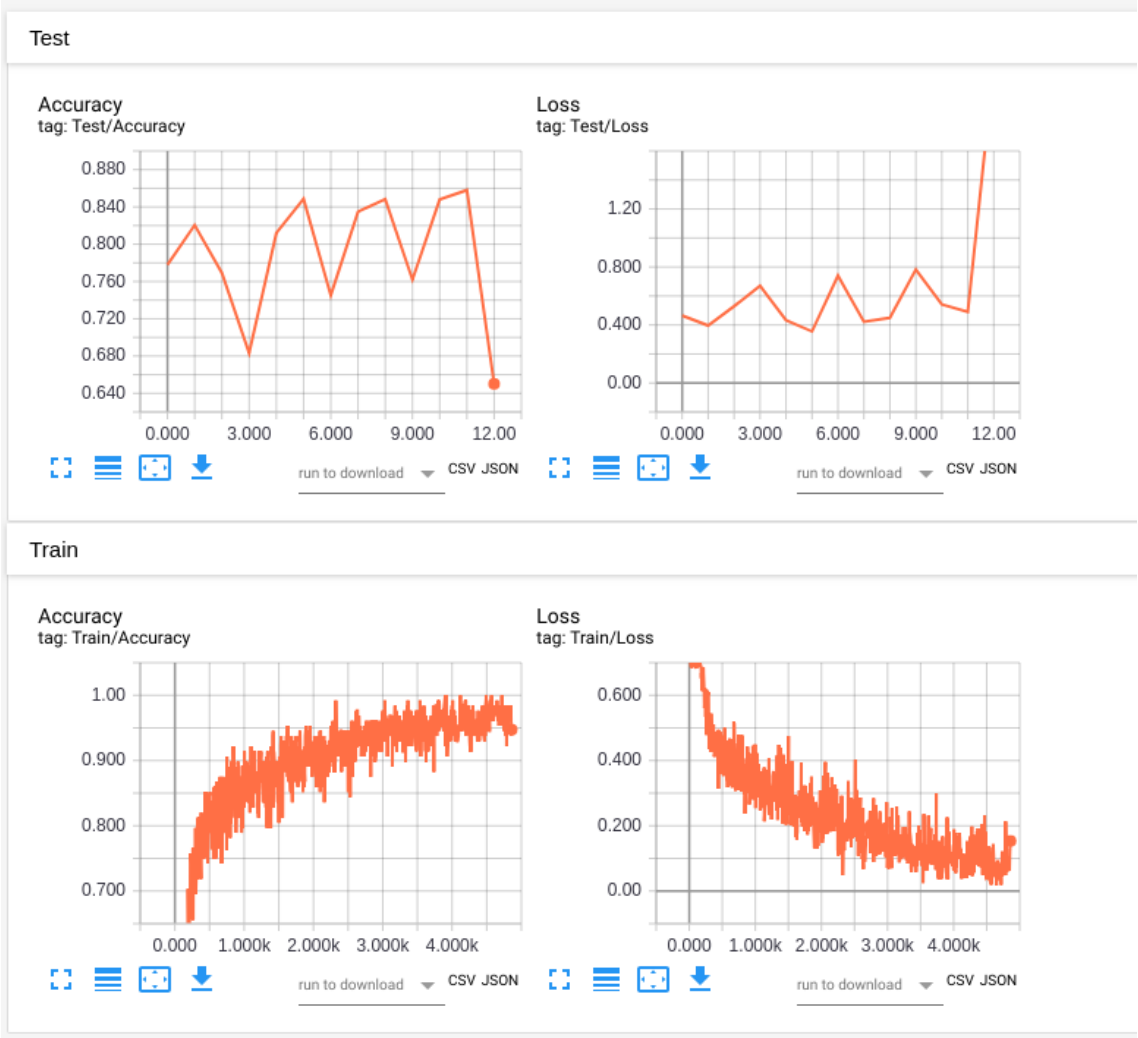


Figure 5.1: Cnn_lstm_english train vs test accuracy and loss plots

5.2.2 Training Process

5.2.2.1 Cnn-Lstm Model

In this section, the training progressed per language was show for ten layer depth. From the Figure 5.1, Figure 5.2, Figure 5.3, Figure 5.4and Figure 5.5, the selected model with best parameters is given by the lowest loss achieved on the test set rather than the highest accuracy on the test set.

5.2.2.2 Vdcnn-Lstm Model

In this section, the training progress of the model with 30 layers was shown per language. From the Figure 5.6, Figure 5.7, Figure 5.8, Figure 5.9 and Figure 5.10, the model with best parameters was selected by the lowest loss achieved on the test set. This model follows the same settings from the model in Section 5.2.2.1. However, the depth of the network was increased to four times of each convolutional block, instead of 2 times.

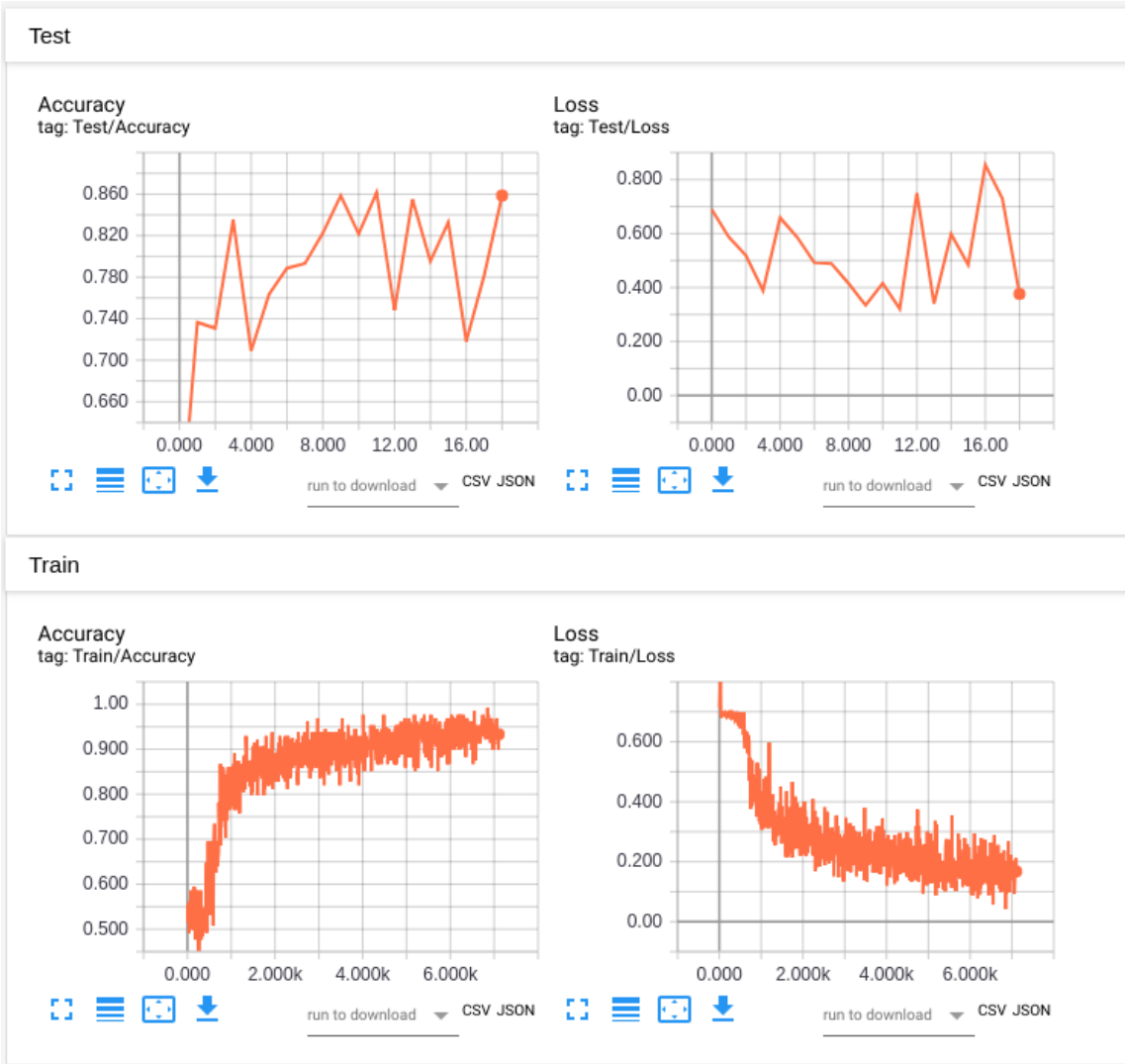


Figure 5.2: Cnn_lstm_german train vs test accuracy and loss plots

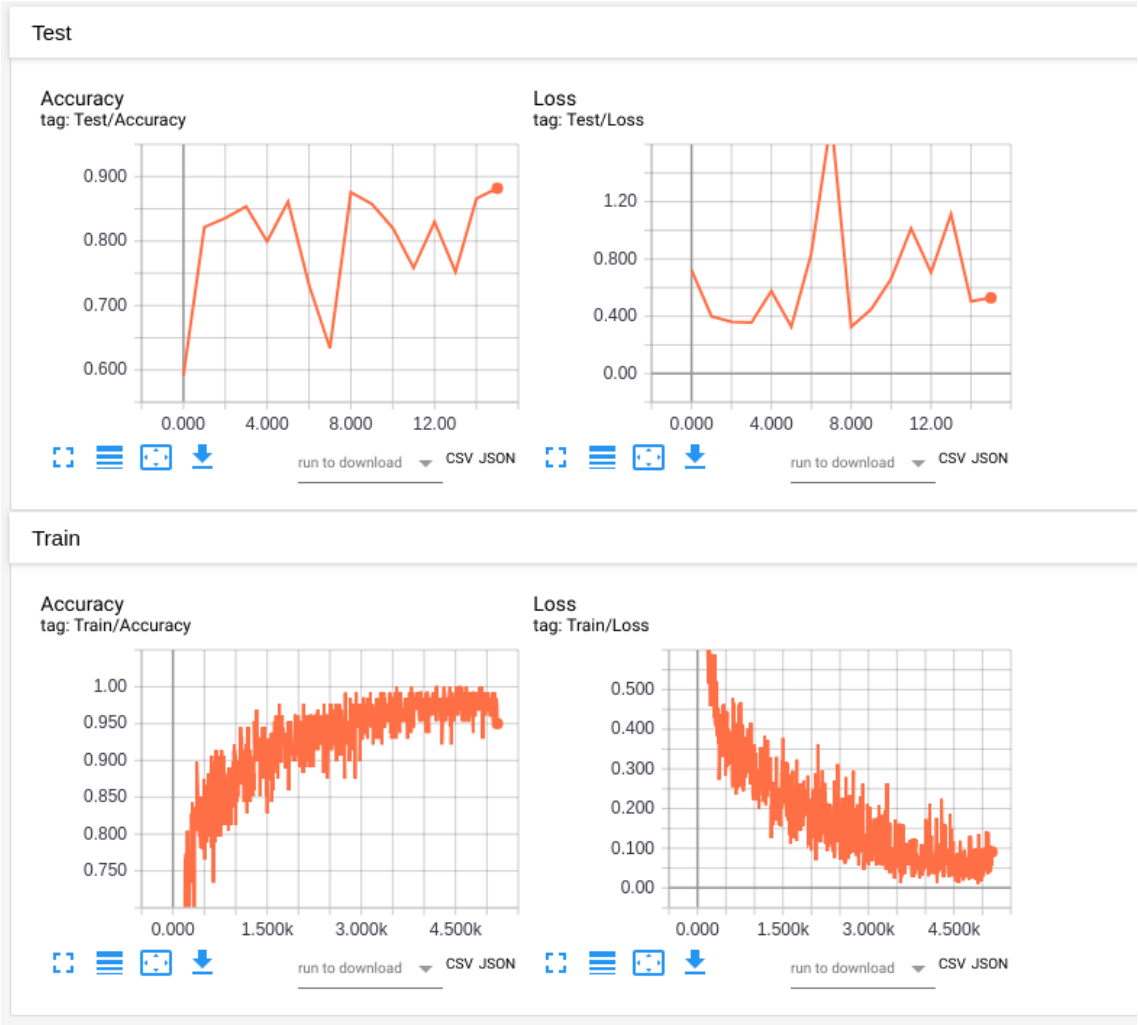


Figure 5.3: Cnn_lstm_french train vs test accuracy and loss plots

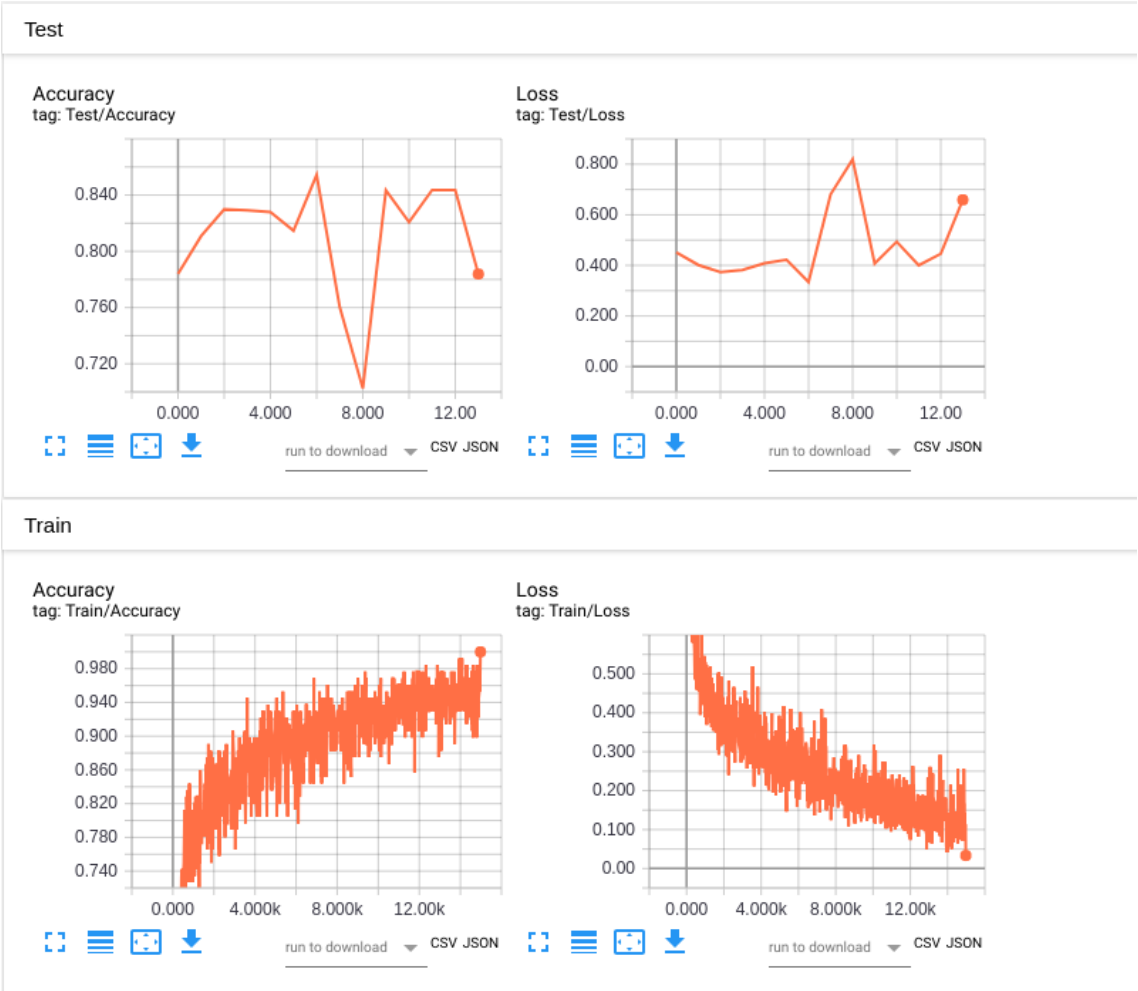


Figure 5.4: Cnn_lstm_m_type_1 train vs test accuracy and loss plots

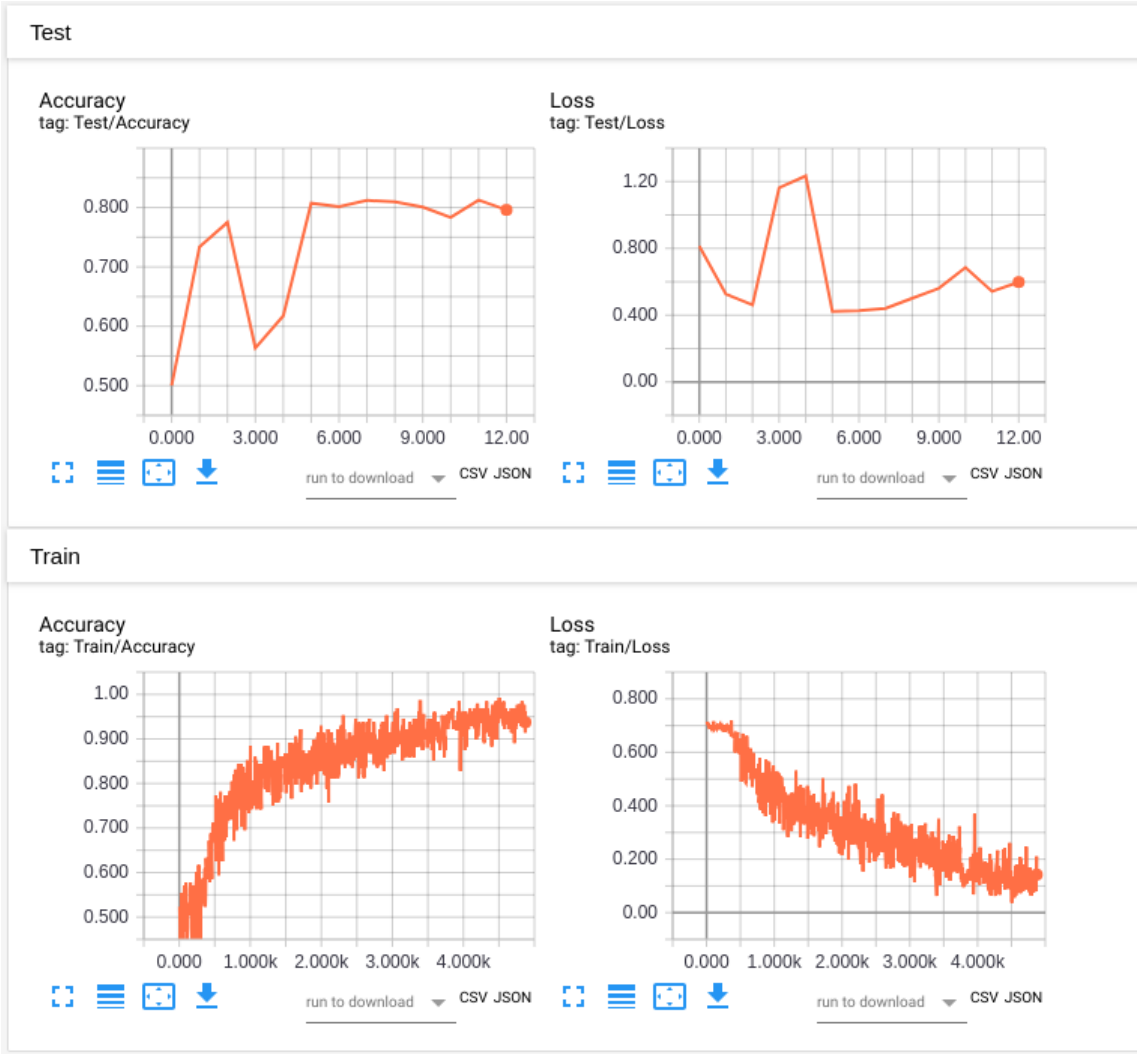


Figure 5.5: Cnn_lstm_m_type_2 train vs test accuracy and loss plots

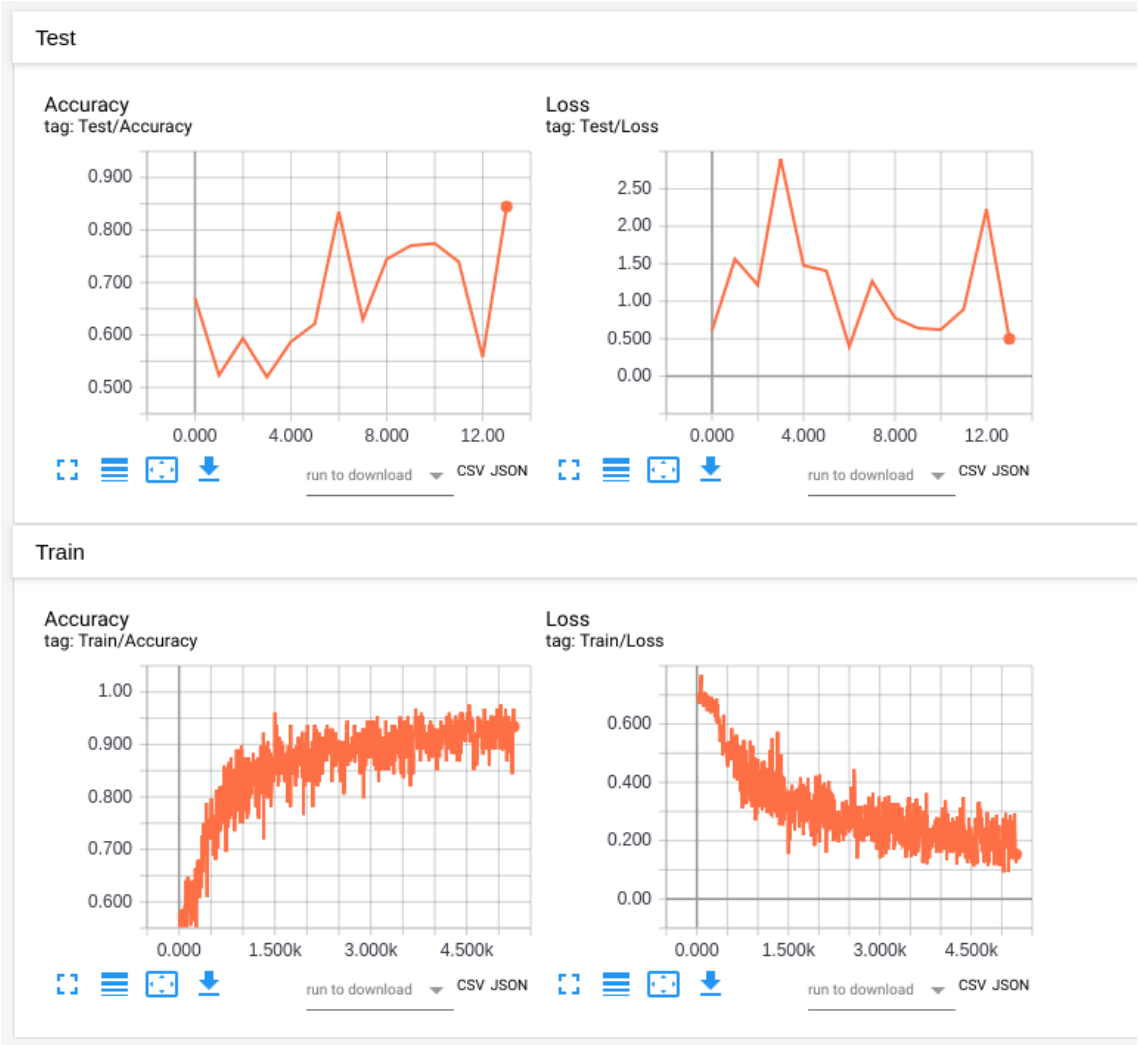


Figure 5.6: Vdcnn_lstm_english train vs test accuracy and loss plots

Figure 5.7: Vdcnn_lstm_german train vs test accuracy and loss plots

Figure 5.8: Vdcnn_lstm_french train vs test accuracy and loss plots

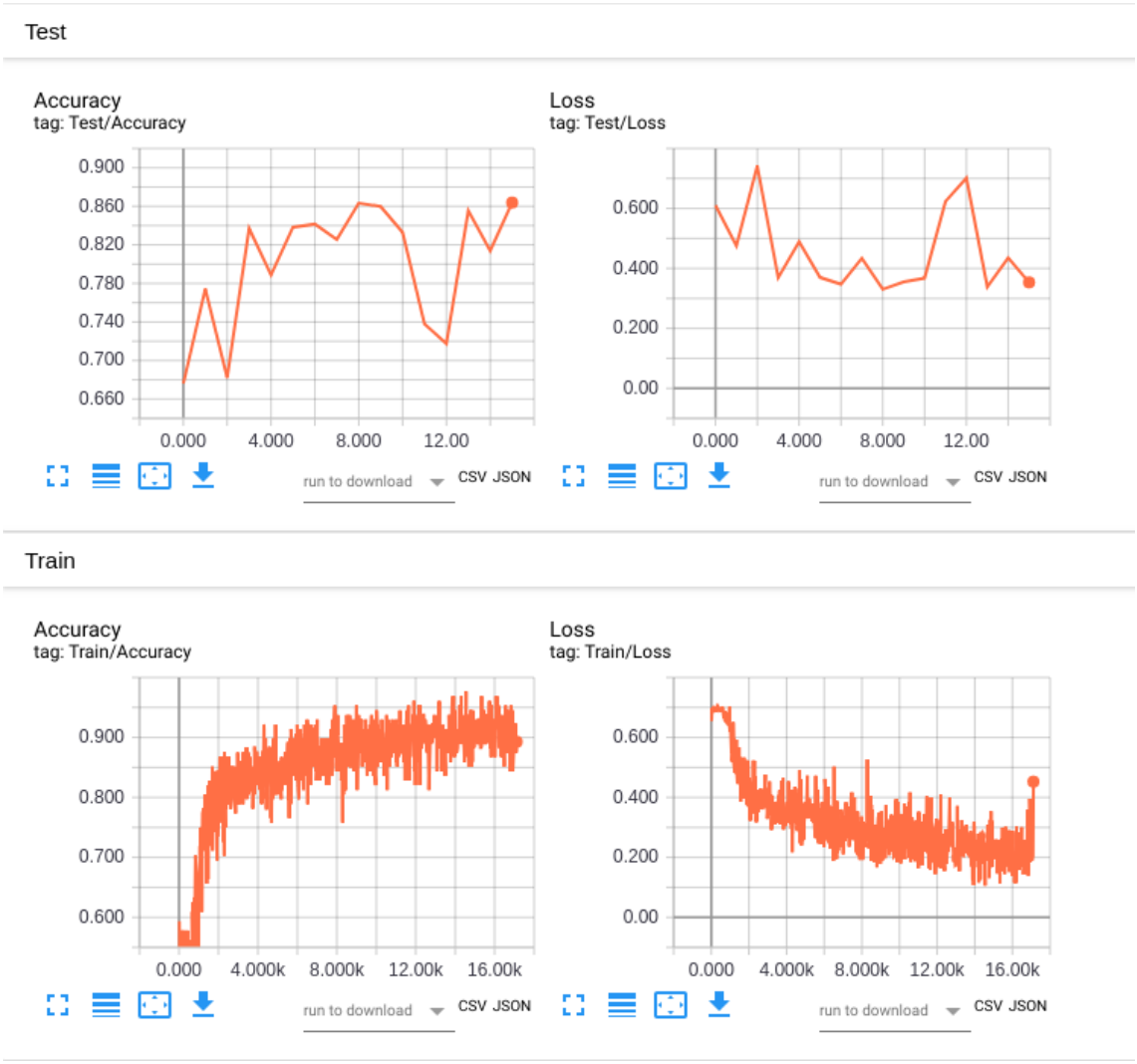


Figure 5.9: Vdcnn_lstm_m_type_1 train vs test accuracy and loss plots

Figure 5.10: Vdcnn_lstm_m_type_2 train vs test accuracy and loss plots

5.2.3 Experimental Results

5.2.3.1 Combination of the Techniques

From the Table 5.3 shows accuracies on the new test data. The proposed models outperformed individual models. The combination of the CNN and LSTM helped to overcome the weaknesses of the both models, i.e CNN and LSTM. The precision, recall and F1_scores of all sentiment analysis models from Section 4.5 were shown in Figure 5.11, Figure 5.12, Figure 5.13, Figure 5.14 and Figure 5.15. From the figures, we can see that getting the desired result alone from CNN or LSTM not possible because they lack the ability to learn long term dependencies and not able to capture important features from the text. Hence, in the unified method, each convolutional layer captures the different representation (i.e invariances) of the features when an input text is passing through them. However, CNN cannot recognise the existed long range semantics in the text. But all these semantics will be recognised by the lstm layer. Therefore, the trained model has not only the knowledge about seen data but also able to learn the patterns within the data and to memorize them whenever the model encounters the new data. Additionally, the optimal hyperparameters of the model were selected based on several experiments in order to improve the model performance.

Phase I with 20 epochs; batch size 64:

Experiment 1: Using Adam with a learning rate of 0.001 for all datasets

Experiment 2: Using Adam with a learning rate of 0.01 for all datasets

Experiment 3: Using SGD with a learning rate of 0.01 for all datasets

Experiment 4: Using SGD with a learning rate of 0.001 for all datasets

Experiment 5: Using Adam with a learning rate of 0.001 for all datasets

Experiment 6: Using SGD with a learning rate of 0.01 for all datasets

Phase II:

Experiment from 1 to 6 with 20 epochs with batch size 32

Phase III:

Experiment from 1 to 6 with 20 epochs with batch size 128

From the above experimental observations, I have found that the changes in batch size, number of epochs and learning rate did not show much difference in performance for model with Sgd optimiser. Therefore, the model performance was improved gradually for each epoch with the following parameters: batch size: 128; Adam optimiser; learning rate: 0.001. These parameters were considered as optimal hyperparameters.

Since the accuracy of the model was not enough to determine model predictions capability, the F1_scores, precision and recall were evaluated to gain more information about each model performance. Usually, F1_score maintains a balance between precision and recall. Sometimes the char level VDCNN model performed relatively better than the combined technique due to its higher rate of true positives predictions than the true negatives. Overall, *the combination of the methods is quite powerful than using a single technique* for sentiment analysis.

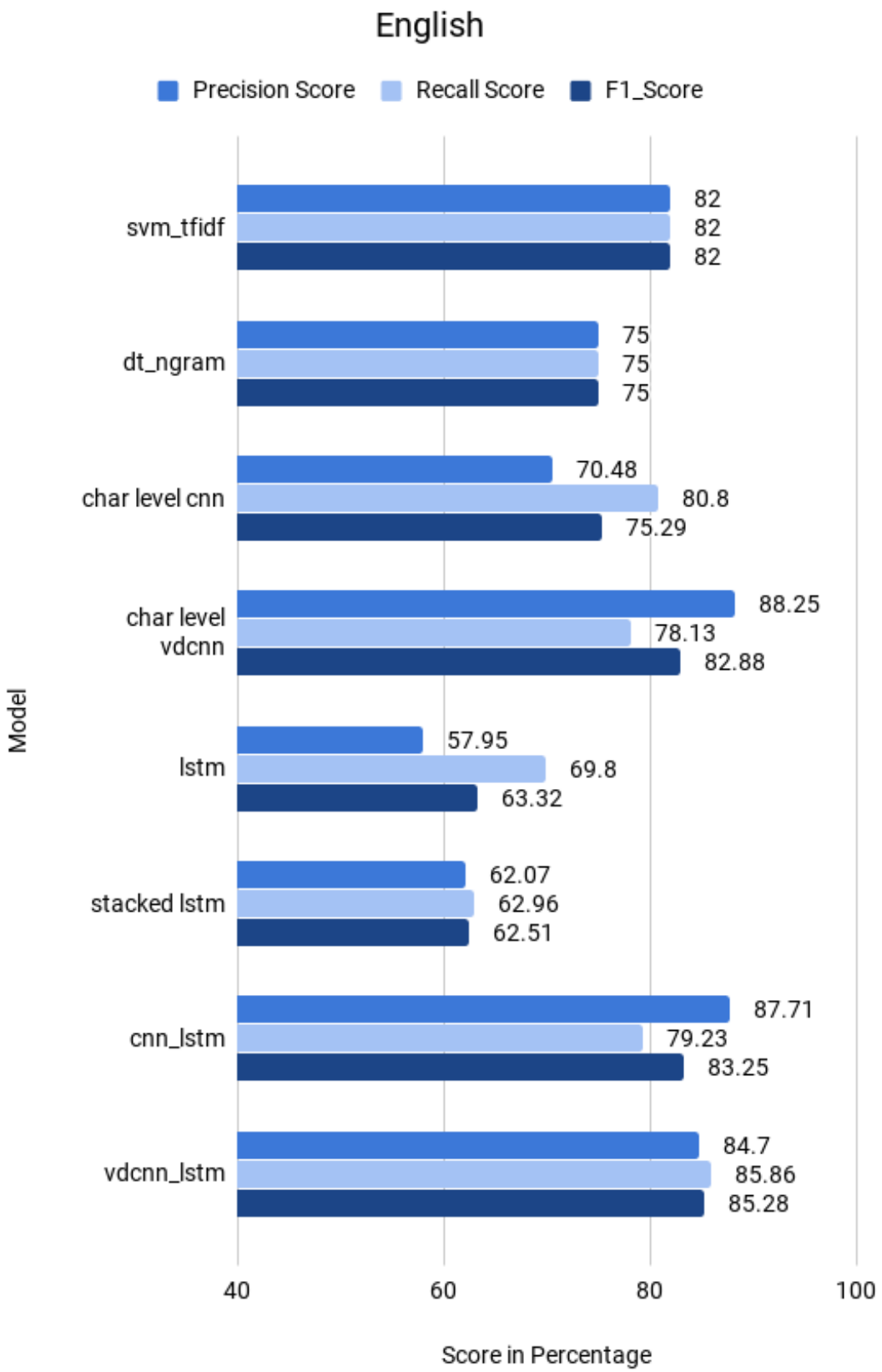


Figure 5.11: English test set

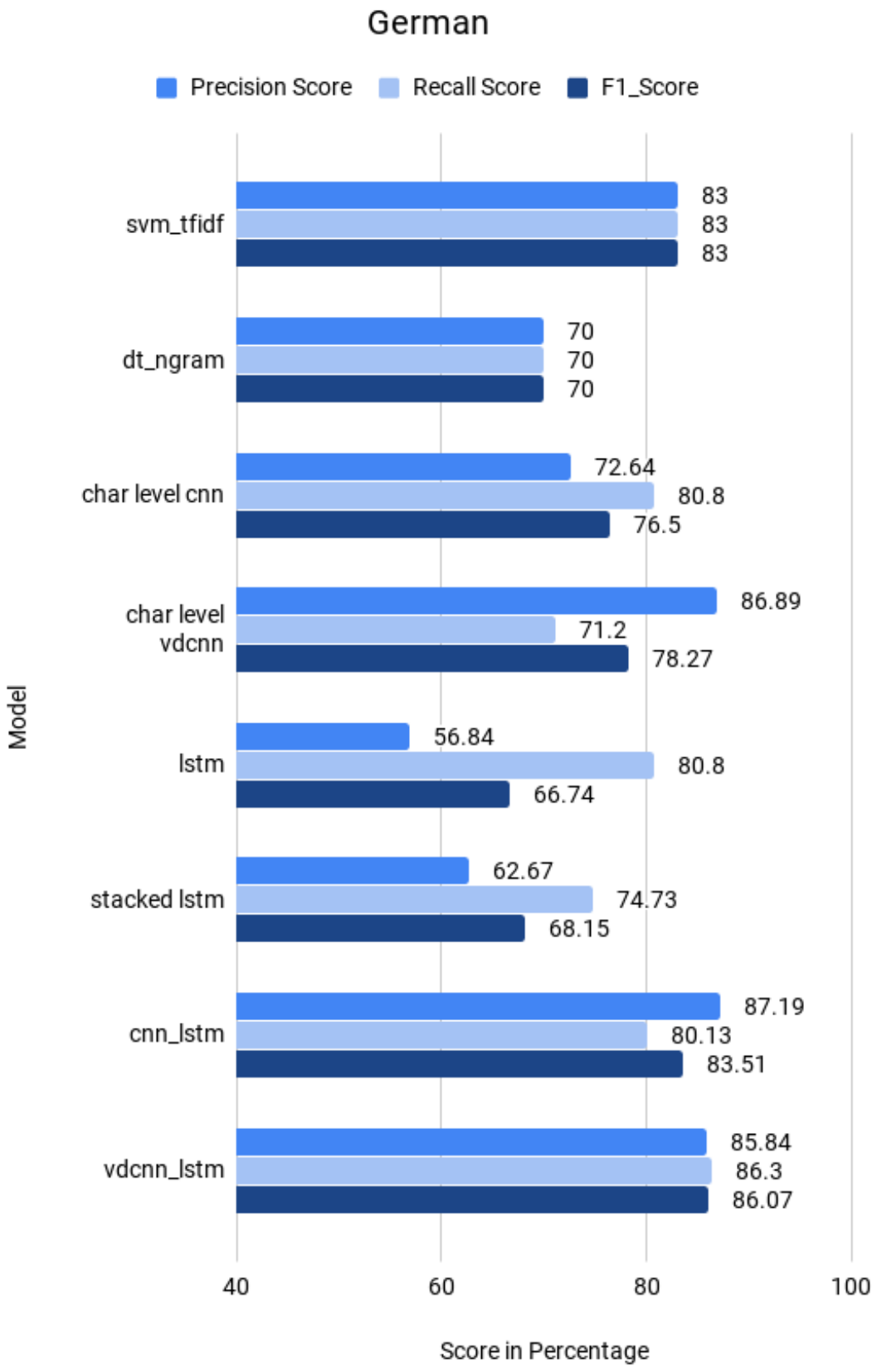


Figure 5.12: German test set

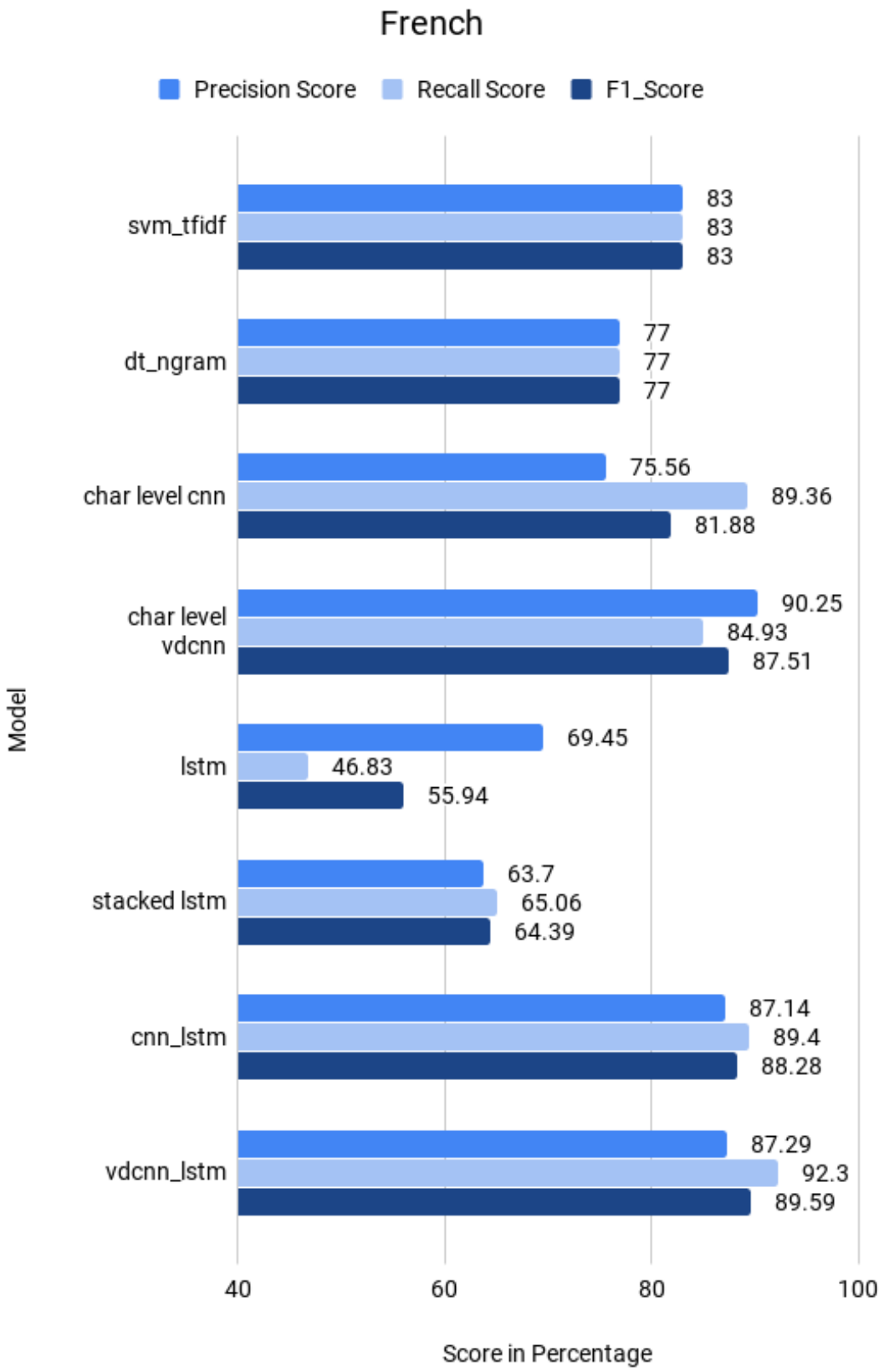


Figure 5.13: French test set

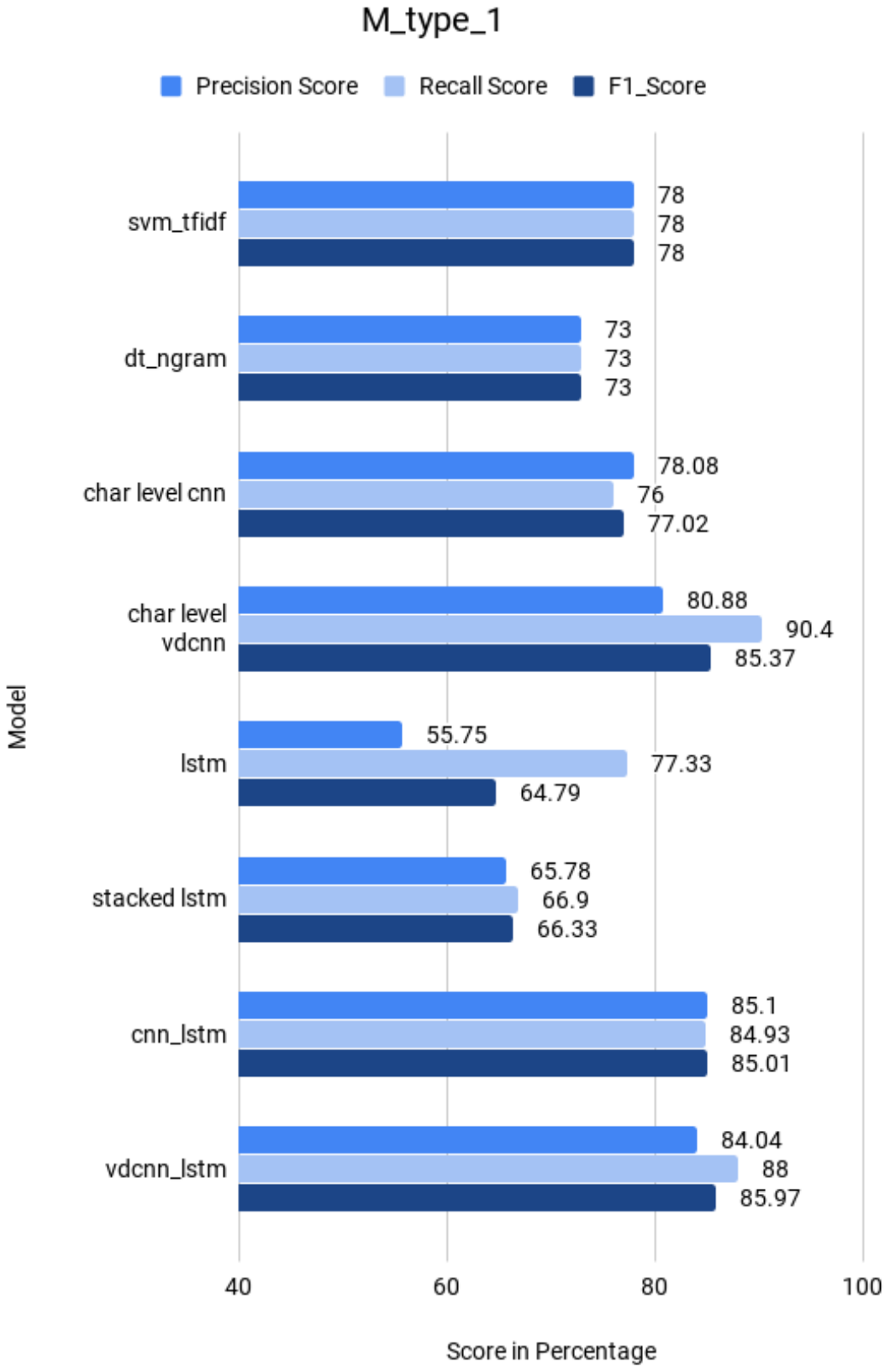


Figure 5.14: M_type_1 test set

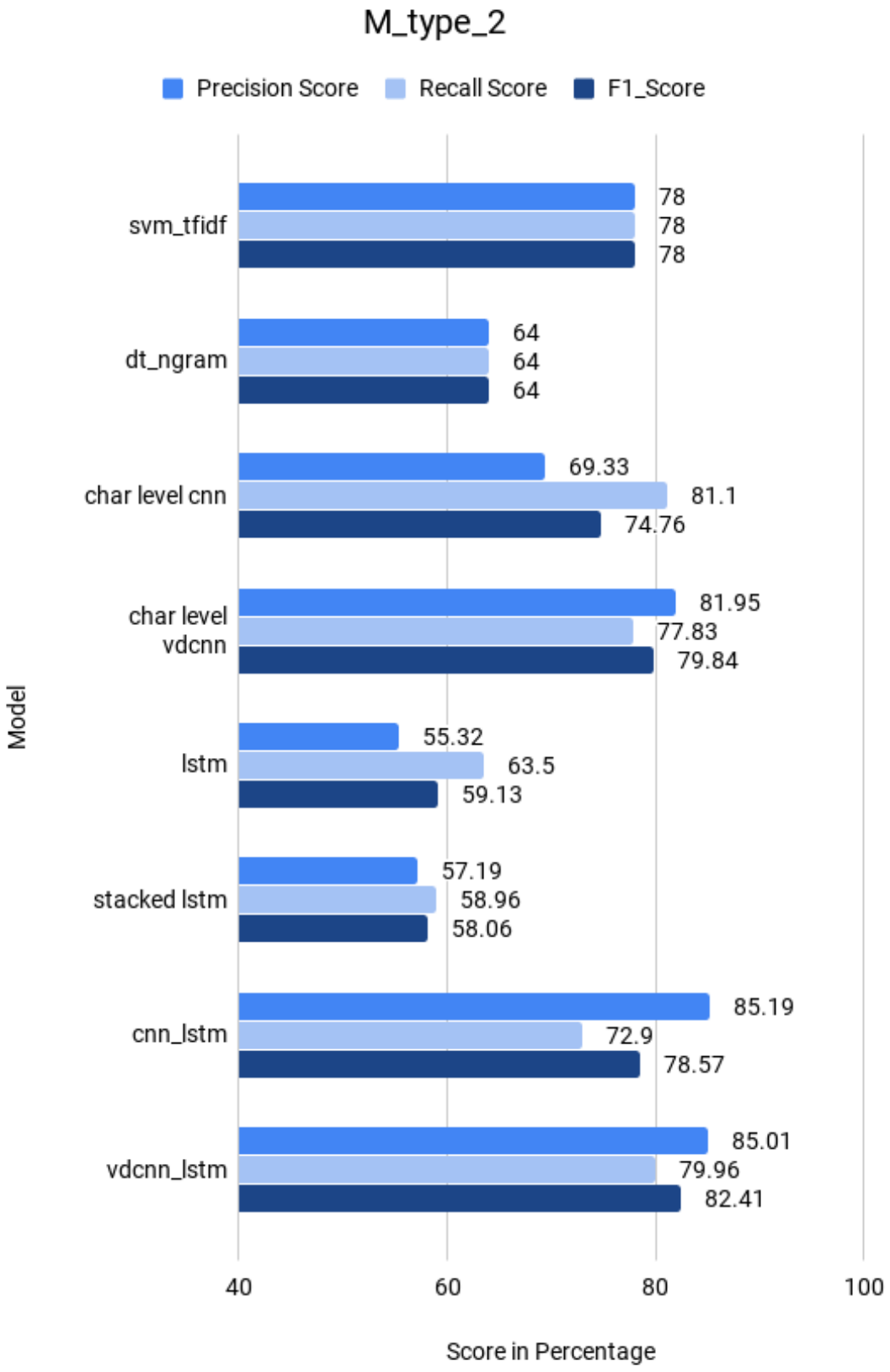


Figure 5.15: M_type_2 test set

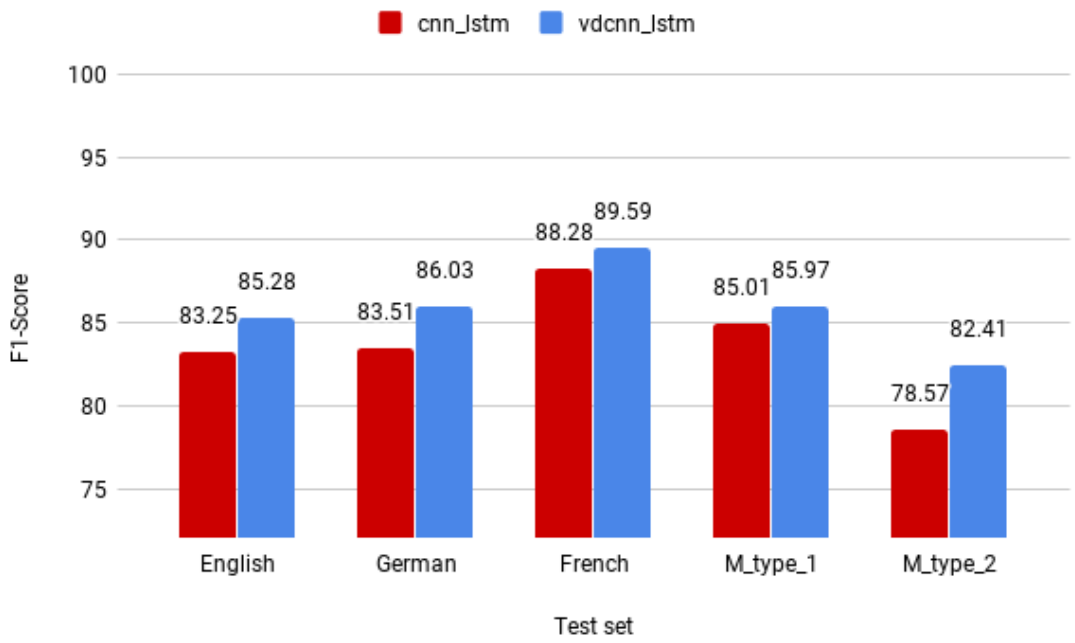


Figure 5.16: F1-scores of proposed models

5.2.3.2 Increasing Network Depth

From the Table 5.3, *we can observe that the network performance improved when the depth of the network increases irrespective of dataset size.* The model accuracy was improved from the depth 10 to 30 for all languages. The more representations were extracted from the text without having a loss of semantic information. Hence, having more layers were became an advantage. Overall, the model with a layer depth of 30 performed best by improving accuracy by *1-2%* and also outperformed baseline models with an accuracy difference of *2-7%*. We can also observe that the effect of an increase in the number of layers even in the individual deep learning models. Unfortunately, the Experimentation of the LSTM models is limited to two layers due to limited resources.

5.2.3.3 Multilanguage Model Performance

Figure 5.16 gives an overview of the F1-scores for *CNN_LSTM* and *VDCNN_LSTM*, when the depth was increased. Based on the observations of precision, recall and F1-scores from Figure 5.11, Figure 5.12, Figure 5.13, Figure 5.14 and Figure 5.15, the two different architectures performed better for single language than the mix of all languages. The difference in model performance was 4-11% between single language and multi-language. However, *the multi-language model has the ability to deal with the text in mixed languages.* In my thesis, two types of mixed language datasets namely M_type_1 and M_type_2 were used. M_type_1 dataset relatively performed better than English and German language models. Since more features were contributed from the French language, the model was able to predict French language test samples correctly. Therefore, to compensate the French language with other languages, a small dataset with 18000 samples having an equal distribution

of randomly selected samples per language was used (i.e. M_type_2). For M_type_2 dataset, relatively shown less performance than individual language because of fewer samples contribute less features.

Model	Depth	English	German	French	M_type_1	M_type_2
svm_tfidf	-	82.23%	82.66%	82.93%	78.05%	78.31%
dt_ngram	-	75%	69.66%	77.08%	73.2%	64.23%
char level cnn	6	73.48%	75.18%	80.23%	77.33%	72.61%
char level vdcnn	9	83.86%	80.23%	87.88%	84.52%	80.35%
lstm	1	59.58%	59.73%	63.11%	57.98%	56.11%
stacked lstm	2	62.25%	65.08%	64%	66.05%	57.41%
Proposed models						
cnn_lstm	10	84.06%	84.18%	88.13%	85.03%	80.11%
vdcnn_lstm	30	**85.18%**	**86.03%**	**89.31%**	**85.65%**	**82.93%**

Table 5.3: New test data accuracy

However, from the Table 5.3 shows that than the baseline models have performed better than the char level CNN, LSTM and stacked LSTM models. Nevertheless, baseline models lack the ability to learn representation from the data. The CNN models have outperformed LSTM models. The following reasons might have affected the LSTM model performance:

- More data needed for training the models and it was time-consuming (approx three days to one week sometime even more than that). Hence, it was became huge difficult to find the optimal hyperparameters.
- In DNN, usually the network performance depends on the number of layers and hyperparameters. Especially in LSTM network, one lstm layer fails to capture all features along with its long term dependencies. Hence, LSTM models have not experimented with more than one lstm layer, with optimal hyperparameters and bidirectional training of the network.

5.3 Summary

In this chapter, we have seen the research questions and datasets used for the sentiment analysis systems. The training process of all the models were also shown per epoch and the corresponding epoch accuracy and loss were stated for training and testing. It proved that the method with combination of techniques performed better than the individual techniques. The study also showed that, the increase in network layer depth lead to improvement in performance. Though the single language model results were better than the multi language model, the multi language model would be comparatively reliable.

6. Conclusion

In this thesis, a novel approach on sentiment analysis was developed by combining two different techniques to predict the sentiment of the text. In addition, the performance was compared between languages and a combination of multiple languages. Furthermore, the current work does neither rely on pre-trained word embeddings nor data preprocessing (also known as data cleaning). I have addressed the factors to influence the model performance through the experimentation. Firstly, a method to combine CNN and LSTM techniques was demonstrated. Secondly, a performance comparison was provided between combination of the techniques and individual techniques, multilanguage and single language and the effect of increase in layer depth on performance by utilizing the lowest atomic representation of the text, i.e. characters. The proposed method was performed better than the existed state-of-the-art methods. Though the single language model results were better than the multi language model, the multi language model would be comparatively reliable.

7. Future Work

Now we know that we can build reliable models with multiple languages by using a combination of techniques. Nonetheless, the method was not tested with more LSTM layers due to limited computing resources and with different parameters. More LSTM layers might improve the model performance even further as some of the previous studies showed that single LSTM layer can't learn all sequential features from data. The proposed technique can also be used for different NLP tasks such as question answering, relation classification (RC), answer selection (AS) and Topic modelling etc. However, the proposed method requires network changes to give reliable results for different NLP tasks. The changes might be by adding more dropout layers, different max pooling operations and parameter changes. Also, the model might perform better with GRU technique instead of LSTM for above mentioned NLP tasks because GRU trains faster and more efficient than LSTM. Above research thoughts are left for the future.

Bibliography

[1] Pang,B., et.al., "Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval," *Information Processing and Managemen*, vol. 2, pp. 1–2, 2008. (cited on Page 1 and 7)

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org. (cited on Page 1)

[3] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *ieee Computational intelligenCe magazine*, vol. 13, no. 3, pp. 55–75, 2018. (cited on Page 1 and 27)

[4] Y. Bengio, I. J. Goodfellow, and A. Courville, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 326–365, 2015. (cited on Page 1)

[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997. (cited on Page 2 and 32)

[6] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very deep convolutional networks for text classification," *arXiv preprint arXiv:1606.01781*, 2017. (cited on Page 2, 28, 30, 31, 35, and 37)

[7] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in neural information processing systems*, pp. 649–657, 2015. (cited on Page 2, 26, 30, 32, and 37)

[8] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," *arXiv preprint arXiv:1607.01759*, 2016. (cited on Page 2 and 26)

[9] A. Karpathy, J. Johnson, and L. Fei-Fei, "Visualizing and understanding recurrent networks," *arXiv preprint arXiv:1506.02078*, 2015. (cited on Page 2)

[10] Gunes, H., and B. Schuller, "Categorical and dimensional affect analysis in continuous input: Current trends and future directions," *Image and Vision Computing*, vol. 31, no. 2, pp. 120–136, 2012. (cited on Page 5)

[11] Alessia et. al., "Approaches, Tools and Applications for Sentiment Analysis Implementation," *International Journal of Computer Applications*, vol. 125, Sept. 2015. (cited on Page 5)

[12] Mohammad, S.M., and P.D. Turney, "Crowd sourcing a word-emotion association lexicon," *Computational Intelligence*, vol. 29, no. 3, p. 436–465, 2013. (cited on Page 6)

[13] Mohammad, S.M., X. Zhu, S. Kiritchenko, and J. Martin, "Sentiment, emotion, purpose, and style in electoral tweets," *Information Processing and Managemen*, vol. 51, no. 4, p. 480–499, 2015. (cited on Page 6)

[14] Saif M. Mohammad, "Challenges in Sentiment Analysis," National Research Council Canada, 2015. (cited on Page 6)

[15] Alessia et. al., "Approaches, Tools and Applications for Sentiment Analysis Implementation," *International Journal of Computer Applications(0975 – 8887)*, vol. 125, no. 3, 2015. (cited on Page 7)

[16] Niu, Y., Zhu, X., Li, J., Hirst, G, "Analysis of polarity information in medical text," 2005. (cited on Page 7)

[17] Ku, L.W., Li, L.Y., Wu, T.H., Chen, H.H, "Major topic detection and its application to opinion summarization," 2005. (cited on Page 7)

[18] Balahur, A., Kozareva, Z., Montoyo, A, "Determining the polarity and source of opinions expressed in political debates," 2009. (cited on Page 7)

[19] Medhat, W., Hassan, A., Korashy, H., "Sentiment analysis algorithms and applications: A survey," 20014. (cited on Page 8)

[20] M. S. Vohra and J. Teraiya, "Applications and challenges for sentiment analysis: A survey," *International journal of engineering research and technology*, vol. 2, no. 2, pp. 1–6, 2013. (cited on Page 8)

[21] S. Baccianella, A. Esuli, and F. Sebastiani, "Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining.," in *Lrec*, vol. 10, pp. 2200–2204, 2010. (cited on Page 8)

[22] K. Dave, S. Lawrence, and D. M. Pennock, "Mining the peanut gallery: Opinion extraction and semantic classification of product reviews," in *Proceedings of the 12th international conference on World Wide Web*, pp. 519–528, ACM, 2003. (cited on Page 8)

[23] Ms. Anjali Ganesh Jivani, "A Comparative Study of Stemming Algorithms," Nov. 2011. (cited on Page 12)

[24] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016. (cited on Page 13 and 15)

[25] S. Hochreiter and J. Schmidhuber, "Lstm can solve hard long time lag problems," in *Advances in neural information processing systems*, pp. 473–479, 1997. (cited on Page 22)

[26] Jeffrey Pennington, Richard Socher and Christopher D. Manning, "GloVe: Global Vectors for Word Representation," (cited on Page 23)

[27] Piotr Bojanowski, Edouard Grave, Armand Joulin and Tomas Mikolov, "Enriching Word Vectors with Sub word Information," 2017. (cited on Page 23)

[28] Anuroop Sriram and Yashesh Gaur, "Robust Speech Recognition Using Generative Adversarial Networks," Nov. 2018. (cited on Page 26)

[29] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr, "Conditional random fields as recurrent neural networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 1529–1537, 2015. (cited on Page 26)

[30] Zichao Yang et al., "Hierarchical Attention Networks for Document Classification," 2016. (cited on Page 26)

[31] Soroush Vosoughi et al., "Tweet2Vec: Learning Tweet Embeddings Using Character-level CNN-LSTM Encoder-Decoder," 2016. (cited on Page 27)

[32] Shoushan Li et al., "Active Learning for Imbalanced Sentiment Classification," 2012. (cited on Page 27)

[33] Sandeep Sricharan Mukku, Subba Reddy Oota and Radhika Mamidi, "Active Learning for Telugu Sentiment Analysis," p. 355–367, 2017. (cited on Page 27)

[34] Jeremy Howard and Sebastian Ruder, "Universal Language Model Fine-tuning for Text Classification," May 2018. (cited on Page 27)

[35] Wenpeng Yin et al., "Comparative Study of CNN and RNN for Natural Language Processing," Feb. 2017. (cited on Page 27)

[36] Matiur Rahman Minar et.al, "Recent Advances in Deep Learning: An Overview," July 2018. (cited on Page 27)

[37] C. Zhou, C. Sun, Z. Liu, and F. Lau, "A c-lstm neural network for text classification," *arXiv preprint arXiv:1511.08630*, 2015. (cited on Page 27)

[38] V. D. Van, T. Thai, and M.-Q. Nghiem, "Combining convolution and recursive neural networks for sentiment analysis," in *Proceedings of the Eighth International Symposium on Information and Communication Technology*, pp. 151–158, ACM, 2017. (cited on Page 27)

[39] Q. Huang, R. Chen, X. Zheng, and Z. Dong, "Deep sentiment representation based on cnn and lstm," in *2017 International Conference on Green Informatics (ICGI)*, pp. 30–33, IEEE, 2017. (cited on Page 28)

[40] M. Cieliebak, J. M. Deriu, D. Egger, and F. Uzdilli, "A twitter corpus and benchmark resources for german sentiment analysis," in *5th International Workshop on Natural Language Processing for Social Media, Boston, MA, USA, December 11, 2017*, pp. 45–51, Association for Computational Linguistics, 2017. (cited on Page 28)

[41] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4580–4584, IEEE, 2015. (cited on Page 28)

[42] J. Deriu, A. Lucchi, V. De Luca, A. Severyn, S. Müller, M. Cieliebak, T. Hofmann, and M. Jaggi, "Leveraging large amounts of weakly supervised data for multi-language sentiment classification," in *Proceedings of the 26th international conference on world wide web*, pp. 1045–1052, International World Wide Web Conferences Steering Committee, 2017. (cited on Page 28)

[43] A. M. Alayba, V. Palade, M. England, and R. Iqbal, "A combined cnn and lstm model for arabic sentiment analysis," in *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pp. 179–191, Springer, 2018. (cited on Page 28)

[44] S. Narr, M. Hulfenhaus, and S. Albayrak, "Language-independent twitter sentiment analysis," *Knowledge discovery and machine learning (KDML), LWA*, pp. 12–14, 2012. (cited on Page 28)

[45] "Deep learning for nlp with pytorch." (cited on Page 32)

[46] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer, "cudnn: Efficient primitives for deep learning," *arXiv preprint arXiv:1410.0759*, 2014. (cited on Page 32)

[47] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of documentation*, vol. 28, no. 1, pp. 11–21, 1972. (cited on Page 32)

[48] M. Hermans and B. Schrauwen, "Training and analysing deep recurrent neural networks," in *Advances in neural information processing systems*, pp. 190–198, 2013. (cited on Page 35 and 37)

[49] P. Prettenhofer and B. Stein, "Cross-language text classification using structural correspondence learning," in *Proceedings of the ACL*, 2010. (cited on Page 38)

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Magdeburg, den 14 März 2019