# Text Classification Using Deep learning Methods

A DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE AWARD OF DEGREE

OF

MASTER OF TECHNOLOGY

IN

**COMPUTER SCIENCE & ENGINEERING**


Submitted by:

**HRITICK PASBOLA**

**2K21/AFI/14**

Under the supervision of

**Dr. SANJAY KUMAR**

(Assistant Professor)




**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042


JUNE 2023

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

# CANDIDATE'S DECLARATION

I, Hritick Pasbola, 2K21/AFI/14 student of M.Tech in Artificial Intelligence, hereby declare that the report dissertation titled "Text Classification using Deep Learning Methods" which is submitted by me to the Department of Computer Science and Engineering, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship or other similar title or recognition.

Hritick Pasbola

Place: Delhi

Date:

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

# CERTIFICATE

This is to certify that the project report entitled "Text Classification using Deep Learning Methods" prepared by Hritick Pasbola (2K21/AFI/14) for the partial fulfillment of the requirements of the Master of Technology degree, embodies the work, we all are doing during 4th semester of our course under the supervision of the supervisor from this college. This work has not been produced or formed the basis of evaluation for any previous diploma degree fellowship or any other title or recognition.

Hritick Pasbola (2K21/AFI/14)
Date :
Place : Delhi

Dr. Sanjay Kumar
Mentor
Assistant Professor

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

# ACKNOWLEDGMENT

I would like to express my gratitude towards my mentor, Assistant Professor Dr. Sanjay Kumar, for giving me the opportunity to work in an impressive area such as deep learning and his guidance, regular supervision, constructive feedback and suggestions in completing the project.

Also I would like to thank my parents, my elder brother and my friends for their love and continuous support throughout my life. Thank you for always giving me the strength to successfully finish this project.

Hritick Pasbola

# ABSTRACT

Text classification is the process in which documents are classified into categories that are already defined. These days the amount of documents is growing at an exponential rate, thus there is a need for classifying these documents as it will be extremely helpful in text retrieving, news classification, spam detection and many more. We combine BERT with different deep learning techniques(BiGRU, 1-D CNN,GRU-CNN and TCN-CNN) and compare its performance with some of the popular methods used in text classification. We first convert the document into BERT embedding using a pre-trained BERT model and then feed it into each different model. If we are using an ensemble method then we use stacking to merge the outputs of the models to obtain the final result. We compare the performance on the basis of accuracy and observe that the models BERT+GRU-CNN and BERT+TCN-CNN perform the best among the models used in this thesis and as good as some of the popular methods.

# INDEX

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ABBREVIATIONS

1. RNN: Recurrent neural network
2. LSTM: Long Short Term Memory
3. TCN: Temporal Convolutional Network
4. 1-D CNN: 1-Dimension Convolutional Network
5. BERT: Bi-directinal encoder representation of transformers
6. GRU: Gated Recurrent Unit
7. MR: Movie Reviews
8. CR: Customer Reviews
9. SUBJ: Subjectivity
10. TREC: Text Retrieval Conference
11. USE: Universal Sentence Encoder
12. PCA: Principal Component Analysis
13. TF: Term Frequency
14. IDF: Inverse Document Frequency

# CHAPTER 1

# INTRODUCTION

Nowadays, a lot of text is being produced every second, while some of the text provides us valuable information, some of this text might be of no use to us. Classifying a large amount of text, increasing every second, into valuable and invaluable would not be very feasible if done manually. That is where text classification comes in, text classification is a machine learning technique that classifies text into predetermined categories. Text classification is used in many areas such as finding the genre of the movies by analyzing the summary or plot of the movie.

Text classification is one of the important fields in natural language processing. The goal of text classification is to allot a predefined category to the documents based on their content. The amount of textual data is increasing at a very high speed which makes text classification tasks such as sentiment analysis, topic labeling, spam detection, and news classification very difficult. Thus, there is a need for automation of text classification accurately.

Deep Learning based techniques are an effective approach in solving many real-life problems like various problems related to Natural language processing like fake news and rumor detection, multilabel text classification, boat detection, abusive content detection and many others [1-6].

In chatbots, deep learning is used to train models that can understand and respond to human language, enabling more natural and intelligent conversations. In image processing, deep learning algorithms analyze and interpret visual data, enabling tasks like object detection, image recognition, and even generating realistic images. In healthcare, deep learning helps in diagnosing diseases, predicting patient outcomes, and analyzing medical images, allowing for more accurate and efficient healthcare decision-making. Overall, deep learning empowers these fields by leveraging

complex neural networks to extract patterns and make predictions, leading to advancements in communication, visual understanding, and healthcare solutions.[37-41]

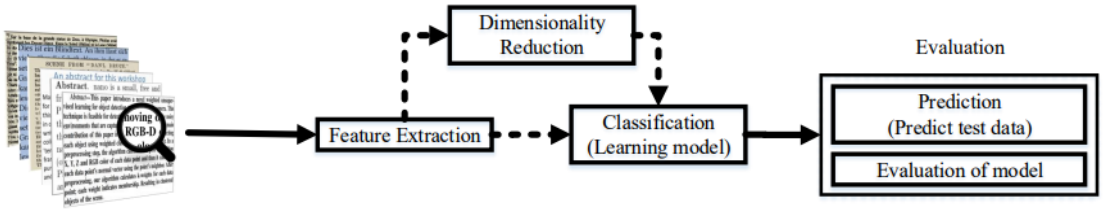


Fig. 1.1 : Text Classification Overview [7]

We can observe the overview of how basic text classification works in figure 1.1. First step is obtaining the data in which we want to perform text classification. After that we can extract features of the document using word embeddings or sentence embeddings. Before feature extraction there can be an extra step of data preprocessing if we use word embedding. In text based problems, data preprocessing is mostly removing stop words, removing hashtags, removing punctuations and others depending on the problem.

In word embeddings we take every word of the document and convert it into a numeric vector. The numbers inside the vector represent the context or meaning of the word, that is words that have similar meaning will have similar vectors. There are many pre-trained word embedding models that are used to achieve greater results such as Glove[8], word2vec[9] and Elmo[10].

Then there is sentence embedding in which we convert the whole sentence into a numeric vector instead of just taking up a word. This obviously is more performance heavy but provides us with better results as the classifier has more context in the numeric vector.

There are many pre-trained models that are used to convert the documents into sentence embedding; some of them include BERT[12], USE[13].

Then the next step in text classification is dimension reduction. This step is only necessary when the data has a lot of unique and uncommon words. As it has a lot of unique words the dimensions of the numeric vector of the embeddings will be a huge amount, which will lead to taking a lot of time in the training phase, so we reduce the dimensions using some of the popular techniques such as PCA[14].

Next is the classification of documents into their correct predefined classes. This is basically the step where we create and train our model. In text classification there are a lot of models that perform well. We can use recurrent neural networks (RNNs) to classify text, they perform well on text based problems as they are good for sequential problems[17]. We can use long short term memory (LSTMs), which are better than RNNs as they have a hidden state which remembers the context of previous words and so on.

In the evaluation phase of the text classification thesis, the goal is to assess the performance of the trained models by comparing their predictions with the actual results. The primary metric used for comparison is accuracy. Here is an expanded version of the explanation:

The evaluation phase is the final step in the text classification process. Its purpose is to measure how well the trained models perform on a specific dataset. To evaluate the models, we compare their predicted classes with the actual results. The accuracy metric is commonly used to assess the performance of classification models. Accuracy represents the proportion of correctly classified instances out of the total number of instances in the dataset.

In this thesis, the approach involves combining static BERT embeddings with multiple models and evaluating their performances based on accuracy. BERT (Bidirectional Encoder Representations from Transformers) has demonstrated excellent results in various text-related problems. It is a type of sentence embedding technique that captures contextual information from text.

It is important to note that the BERT embeddings used in this study are static, meaning that the pre-trained BERT model is not further trained during the training phase. The weights of the pre-trained BERT model remain unchanged throughout the training period.

Once the BERT embeddings are obtained, they are fed into several models, including GRU (Gated Recurrent Unit), 1-D CNN (Convolutional Neural Network), 1-D CNN_GRU (combination of CNN and GRU), and 1-D CNN-TCN (Temporal Convolutional Network). These models process the BERT embeddings and make predictions for the text classification task.

In the results section of the thesis, the performances of these models are compared with each other and potentially with other popular models used in text classification. The main criterion for comparison is accuracy, which provides an indication of how well the models classify the text data. By evaluating the accuracy of the different models, it becomes possible to assess their relative performance and identify the most effective approach for the specific text classification task at hand.

We have used four datasets to observe the performance of the 1-D CNN-TCN, whereas we only use two datasets to compare the performances of the other models. The four dataset we use are Movie Reviews(MR), Customer Reviews(CR), Text Retrieval Conference(TREC), Subjectivity(SUBJ), these are used in the 1-D CNN_TCN model, whereas in other datasets we use only MR and CR. In the conclusion section we provide what could be done further that might improve the performances of the models.

# CHAPTER 2

# PREVIOUS WORKS

As there is a rapid growth in the field of artificial intelligence, many methods were introduced to perform text classification, some of them are machine learning methods, while some of them are deep learning methods or some of them are a hybrid method. In this section we shall discuss all the previous work that is related to this thesis. First we shall discuss the various methods that are used to extract the features of the data such as word embedding and sentence embedding and their pre-trained models that are used to do so, then we will discuss various techniques used to classify the documents into their respective pre-defined classes. Some of these techniques are Recurrent neural network(RNN), multilayer perceptron and 1-D convolutional neural network[18][19].

## 2.1 Projection into feature space

Documents are in the form of multiple strings and these strings need to be converted into some form of numbers so it can be fed into a neural network and can be further predicted which class it belongs to. We will be discussing three ways we project the input data into a feature space; bag of words, word embeddings and sentence embeddings.

### 2.1.1 Bag of Words

In bag of words, we take every word and convert it into one hot encoded vector that is for every unique word there is a position in the vector that will be 1 whereas the other will be 0 for that word. We can clearly see how the problems faced in this method will look like, if the vocabulary of the document is very high then the dimensions of

the vector will be great, thus taking a lot of time in the training phase. This method also ignores the sequential representation or the position of the word in the sentence. Still it manages to achieve great results in text classification. It is used along with term frequency(TF) which encodes the whole document in one vector where the position of the word has the amount of time the word has been encountered in the document. It is further used again inverse document frequency(IDF) which holds the amount of time a word has occurred in the whole corpus. Using IDF, it is able to classify better if a unique word comes up.[20-22][23]

### 2.1.2 Word Embeddings

One of the most popular ways to extract the features of the document is word embedding. In word embedding we take every word in the vocabulary and give it a numeric representation in the form of a vector. So for every word there is a unique representation in the form of a numeric vector. The numbers inside the vector represent the meaning of the word, this means that words with similar meaning will have similar numeric representation. Let us take an example to understand this better. If you take the word “rabbit” and take its word embedding and plot it in a graph. Similarly you take another word “bunny” and plot it in the same graph then the euclidean distance of these two words will be small as compared to euclidean distance with some word that is not related to any of these two words. This method has achieved great results but we can see how or where it is lacking that is not taking the position of the word in the sentence. The position of the word in the sentence gives us a lot of context and word embedding does not take this in while extracting the features. To extract the word embeddings of a document we can either create our model from scratch or use a pre-trained model for word embeddings such as word2vec, GloVe. [24]

### 2.1.3 Sentence Embeddings

Sentence embeddings are similar to word embeddings but instead of considering just a single word we take up a whole sentence and convert it into a numeric vector which represents the context of the sentence. This helps us remove the flaw that wod embedding does take up the position of the word in the sentence because sentence embeddings contain the whole sentence which helps us take positioning of the word also while extracting the features. One more advantage of sentence embeddings over word embeddings is that we do not require to preprocess the document before converting it into sentence embedding, whereas in word embedding some preprocessing is required. Sentence embedding can be done using a newly created model but training it will take a lot of time, so most of the time people use a pre-trained model to obtain the sentence embeddings. There are many available pre-trained models such as BERT, USE. BERT has achieved great results in text classification and is now very commonly used to perform text classification.

## 2.2 Deep Learning Models

After extracting the features there are many ways we can predict the class of the document. There are methods that use machine learning but we will discuss only deep learning methods in this thesis. There are many deep learning methods that are used to classify text, but given below are those that are relevant to this thesis.

### 2.2.1 MultiLayer Perceptron

A multi-layer perceptron (MLP) is a type of artificial neural network that is known for achieving good results in various tasks, including text classification. Let's expand on the explanation:

After extracting the features from the document, such as word embeddings or TF-IDF (Term Frequency-Inverse Document Frequency) representation, these features are then fed into the multi-layer perceptron for predicting the class of the document.

The input layer of the MLP has nodes equal to the vocabulary size in the case of a bag-of-words representation. Each node represents a specific word or feature from the vocabulary. Alternatively, if word embeddings are used, the input layer will have nodes equal to the dimension of the embedding vector. Each node in the input layer corresponds to a specific feature or dimension in the word embeddings.

The hidden layers in the MLP are responsible for learning complex patterns and representations from the input features. These hidden layers can have varying numbers of nodes, and the more layers there are, the deeper the network becomes. Each node in the hidden layers applies a non-linear transformation to the weighted sum of inputs received from the previous layer, allowing the network to learn more abstract and higher-level representations.

Finally, the output layer of the MLP has nodes equal to the number of predefined classes in the text classification task. Each node in the output layer represents a specific class, and the output values from these nodes indicate the model's predicted probabilities or scores for each class. The class with the highest score is typically selected as the predicted class for the document.

During the training phase, the MLP adjusts the weights and biases of its nodes using techniques like backpropagation and gradient descent to minimize the difference between the predicted class and the actual class labels in the training data. This process of adjusting the model's parameters allows the MLP to learn the underlying patterns and relationships in the data, enabling it to make accurate predictions on unseen documents.

In summary, an MLP is a simple yet powerful neural network architecture that can be effectively used for text classification tasks. It takes features extracted from documents, such as word embeddings or TF-IDF representations, as input, and through its hidden layers, it learns to classify documents into predefined classes based on these features. The output layer provides the predicted class probabilities, and by

adjusting the network's parameters during training, the MLP improves its accuracy in classifying documents.

### 2.2.2 Recurrent Neural Networks (RNNs)

RNNs (Recurrent Neural Networks) have indeed played a crucial role in revolutionizing text classification tasks. They remain a popular architecture due to their ability to effectively handle sequential data. RNNs capture the positional information of words within a sentence, providing valuable context that aids in classifying documents.

The distinguishing feature of RNNs is their recurrent nature, which allows them to maintain an internal memory or hidden state. This hidden state retains information from previous inputs in the sequence, enabling the network to capture dependencies and temporal relationships between words. By considering the sequential order of words, RNNs can understand the context and meaning of a sentence more comprehensively.

In text classification, RNNs are commonly combined with word embeddings. Word embeddings are dense vector representations that capture the semantic information of words based on their contextual usage. However, word embeddings alone are unable to inherently capture the sequential nature of word order in a sentence. By integrating RNNs with word embeddings, we can leverage the strengths of both approaches.

The typical workflow involves first extracting features from text using a word embedding model such as Word2Vec or GloVe. These models generate word embeddings by mapping words to high-dimensional vectors based on their contextual meaning in the training corpus. While word embeddings capture semantic information, they lack explicit awareness of word order.

After obtaining the word embeddings, they are fed into the RNN model. The RNN processes the sequence of word embeddings, taking into account the sequential relationships between words. As the RNN iterates through the sequence, the hidden state is updated, incorporating the information from previous words and considering the sentence's contextual context.

By combining word embeddings and RNNs, we can enhance the text classification process by incorporating both semantic and sequential information. This approach is widely popular as it allows RNNs to effectively capture the nuances and dependencies within a sentence, leading to improved performance in text classification tasks.

### 2.2.3 1-D Convolutional Neural Network (1D - CNN)

Convolutional Neural Networks (CNNs) have gained immense popularity and achieved remarkable performance in the field of computer vision. The use of 2D CNNs in computer vision involves extracting important features from images, which are then fed into the neural network for prediction. This approach significantly speeds up model training and prediction as it focuses only on the relevant features rather than processing the entire image. Consequently, 2D CNNs have become a powerful tool in computer vision tasks.

In text classification, 1D CNNs can be employed to extract essential features from the text. The process involves taking patches or segments of the text and applying weights to the words or characters within the patch to obtain the output. This allows the model to capture local patterns and important information present in the text.

Additionally, max pooling or average pooling can be applied within the patch to further summarize the extracted features. Pooling operations reduce the

dimensionality of the data by aggregating the information, such as taking the maximum or average value within the patch. This helps in capturing the most salient features while reducing the overall computational complexity.

One notable advantage of 1D CNNs is their translation invariance property. Since the same kernel or filter is applied to every patch in the data, patterns identified earlier at specific positions can be detected again at different positions. This property allows the model to recognize and extract relevant patterns and features regardless of their specific location within the text.

By utilizing 1D CNNs in text classification, we can effectively extract important features from the text, similar to how 2D CNNs extract features from images in computer vision. This approach enables the model to focus on the crucial information, leading to improved performance and efficiency in text classification tasks.

### 2.2.4 Bi-directional Long Term Short Memory (Bi-LSTM)

To understand about Bi-LSTM we must first get a basic understanding about how LSTMs work.

#### 2.2.4.1 Long Short Term Memory (LSTM)

LSTM are a special variation of recurrent neural networks(RNN) that have the ability to understand long term dependencies. LSTMs were created such that they remove the flaw of RNN of long term dependency. LSTMs also are made up of a chain-like structure like RNNs but there is a big difference in the repeating module.

The horizontal line running above is the main thing in LSTM. It allows information to run through the whole chain easily. We have the ability to add or remove information from this horizontal line called cell state.

The LSTM basically consists of three gates which are further explained below.

**Forget Gate**

The cell state which holds information from previous states, will have some information that may be irrelevant now, so to throw that information out we use the input gate, which uses the sigmoid function to decide how much information has to be removed from the cell state.

**Input Gate**

We also have to add some more information in the cell state, this is done using the Input gate, which is the sigmoid function which helps us in updating the value in the cell state . The other is the tan function which helps us add new information we have to add to the cell state.

**Output Gate**

Now, we have to make decisions so we can give the output, which is based on the cell state, previous output and input.

**2.2.4.2 BiLSTM**

BiLSTM are basically two LSTM implemented together but one goes in the forward direction and the other in the backwards direction that means one LSTM starts from the end of the sentence while the other starts from the start of the sentence. This helps the LSTM store bi directional information as at a point it has information of the left of the sentence and information of the right of the sentence . This helps LSTM get bidirectional context of the sentence, thus helping achieve better accuracy than normal LSTM. [11]

### 2.2.5 Ensemble CNN-LSTM

Random Multimodel Deep Learning (RMDL) is a great deep learning model that can be used to classify anything, it was introduced by K. Kowsari et al. It basically uses Ensemble based learning in which we come up with two or more than two models to get a result. There are two types of Ensemble based learning, in the first one a weight is assigned to the result of each model, after applying the weights, we get the result by adding the result together. The other one is when we feed the results of one model into another model which predicts the result using the previous results of multiple models. The first method is known as weighted average while the second is known as stacking. There are more methods, but in this project we only use weighted average, so we won't discuss it any further.[25]

We use 1D CNN and BiLSTM together to implement an ensemble based learning model. We choose these two models because,

- As BiLSTM takes on information from both sides it works well on temporal data.
- 1D CNN is great at text classification even without much hyperparameter tuning.

### 2.2.6 Temporal Convolutional Network (TCN)

A dilated casual version of convolutional neural network is the TCN which was introduced by Shaojie Bai et al.. Instead of using recurrent neural networks(RNN) we use TCN because if we get a long input sequence while using RNN we tend to get vanishing or gradient problems but this does not happen in TCN. [26][27]

The model we implement in this project is highly inspired by Christof Henkel[28]. The model has the following.

- Using dilation factors as one, two and four and kernel size as three we put two TCN blocks.

- There are 128 filters in the first TCN block whereas there are 64 filters in the second TCN block.
- The result of the final TCN block is passed into two different pooling layers.
- The results of the pooling layers are merged using concatenation and are feed into a sixteen neuron dense layer.

# CHAPTER 3

# PROPOSED WORK

## 3.1 Problem Statement

Text classification is one of the important fields in natural language processing. The goal of text classification is to allot a predefined category to the documents based on their content. The amount of textual data is increasing at a very high speed which makes text classification tasks such as sentiment analysis, topic labeling, spam detection, and news classification very difficult. Thus, there is a need for automation of text classification accurately. There are models that can perform text classification that also with great accuracy but we try to create a model that gives us better accuracy than existing models.

## 3.2 Motivation

Classifying text as fast as possible can help people stop reading offensive comments and fake news. If the comments or news are removed as soon as they are posted, people will not be able to read them. As the amount of data that is being posted online is a lot and cannot be manually checked by people every time, there is a need for a classifier that can classify documents into their respective classes and can be dealt with accordingly. It also helps in classifying millions of documents into correct classes helping people whenever they want to retrieve them. Also performs sentiment analysis on documents which gives the overall emotion that the document sends. Thus, an approach to solve this problem is to create a system that helps in classifying documents in real time with great accuracy.

## 3.3 Methodology

We use BERT embeddings with four different models and compare their performance. In this
section we will discuss all the different models along with BERT embeddings. The four
different models are: BiGRU model, 1-D CNN model,GRU-CNN ensemble and CNN_TCN ensemble.

### 3.3.1 BERT embeddings

BERT has achieved great results in text based problems. We will be using the pre-trained BERT model to convert our documents into BERT embeddings which is basically a numeric vector and the numbers inside represent the meaning of the sentence. The pre-trained BERT will not be fine-tuned during the training phase, thus we will be using static BERT embeddings. This is done in every model used in this thesis. BERT embeddings are sentence embeddings and the pre-trained BERT model has a very huge amount of parameters, thus making it difficult to train it from scratch, so we are using a pre-trained BERT model.

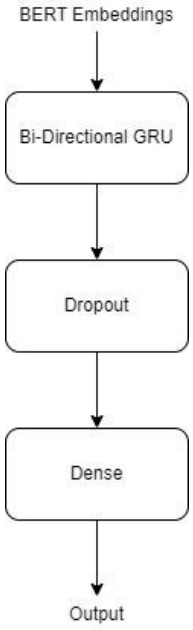


Fig. 2.1 BiGRU Model

### 3.3.2 BiGRU Model

After converting the document into BERT embeddings using the pre-trained BERT model, we feed those embeddings into a bidirectional layer. The Bidirectional layer consists of two GRU networks, in which processes the data from left to right while the other processes data from right to left. This gives the GRU more context and thus performs better. Then it is further fed into a Dropout layer which randomly changes some number of the vectors to zero. This helps in prevention of overfitting. Lastly there is a Dense layer. For better understanding there is a flowchart which represents the sequence of all the layers present in this model in figure 2.1.

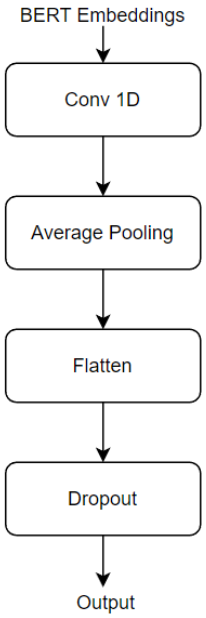


Fig. 2.2 1-D CNN Model

### 3.3.3 1-D CNN Model

BERT embeddings are also fed into the second model which is the 1-D CNN Model whose layers can be seen in figure 2.2. The embeddings are first fed into a Conv1D layer which performs convolutions on the BERT embeddings. It is the same as 2D convolutions but here the filters are a single row and are as in 2D convolutions are applied to a part of the input. It is further followed by an average pooling and flatten layer. In average pooling a part of the input is taken and all the values inside the part are averaged. The average value is the output that will be saved in the output vector. In the flatten layer we reduce dimension to one. Lastly, we have added a dropout layer to prevent overfitting.

### 3.3.4 GRU-CNN Ensemble

The 1-D CNN model is good in capturing the low level features whereas the GRU is good in capturing the high level features. So after the document is converted into numeric vectors using the pre-trained BERT model we simultaneously feed the BERT embeddings in both the 1-D CNN model and the BiGRU model. Ensemble is a model where we combine multiple models to get better results as when they are used alone. Ensemble models use many ways to integrate multiple models such as voting, stacking and many more. We will be using stacking in this model. In stacking the results from multiple models are combined and are then further fed into a final classifier. We take the outputs of both the models, the 1-D CNN model and the BiGRU model and concatenate the outputs into one vector and then further feed it into a dense layer to obtain the final output. So in this model, we basically obtain the BERT embedding, feed it into both the models, concatenate the result and finally pass it through a dense layer to get the output.

### 3.3.5 TCN-CNN Ensemble

We shall discuss the TCN-CNN in two sections, first we explain about the TCN model then how we merge the results of both the models. The CNN model used here is the same as the above CNN model.

#### 3.3.5.1 TCN Model

After obtaining the BERT embeddings, we feed the embedding into the TCN model. The layers present in the TCN model are shown in figure 2.3. It contains a spatial dropout 1D layer which helps prevent overfitting, then it is followed by two TCN blocks which is a dilated version of CNN, further into a global average pooling layer and finally a last dropout layer to further prevent overfitting to obtain the output of this model.

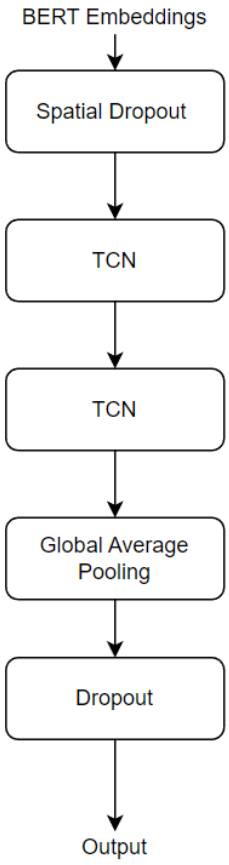


Figure 2.3 TCN model

### 3.3.5.2 Concatenation of outputs into final output

After obtaining the output from both the models, the TCN model and the 1-D CNN model, we take both the outputs concatenate them into a single output and finally feed them into a fully connected neural network to obtain the final output. This can be seen in figure 2.4.

We also note that the embedding from the BERT model is static, that is the weights of the pre-trained are not fine tuned according to the dataset we are using. This helps in training the model faster as the trainable parameters count is lower. So,, the four phases can be summarized as converting the input into BERT embedding using a

pre-trained BERT model, then feeding these embeddings into both, TCN model and 1-D CNN model, lastly merging the outputs and feeding them into a dense neural network to get the final result.

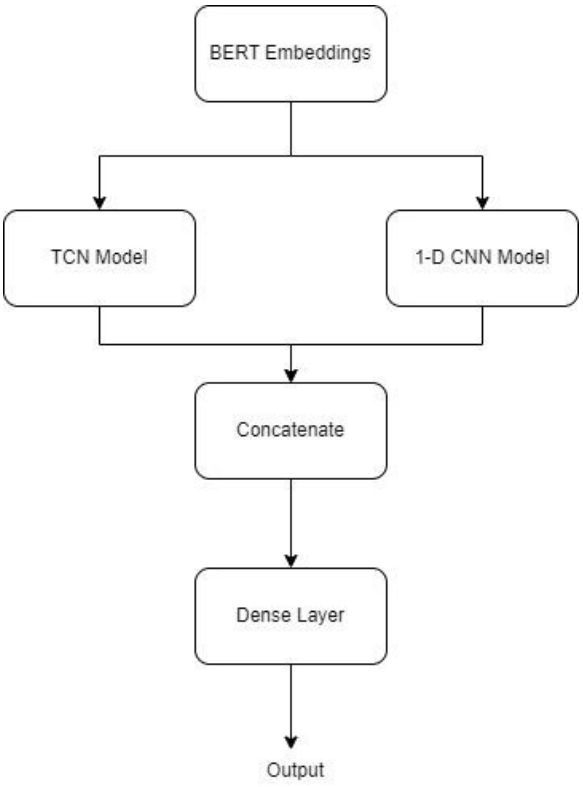


Figure 2.4 Basic view of the model

# CHAPTER 4

# DATASETS

We will be using four datasets to test the performance of the model and we will be using accuracy as a metric to compare with the popular models used for text classification.

| Dataset | Classes | Samples | Average Sentence length |
|---|---|---|---|
| MR | 2 | 10662 | 20 |
| TREC | 5 | 5952 | 10 |
| CR | 2 | 3775 | 19 |
| SUBJ | 2 | 10000 | 23 |

Table 4.1 Dataset statistics

Some dataset have specified splits for train and test but some do not. So for the datasets that do not have any specified splits for train and test we will be using a ten fold cross validation for them. The four datasets we will be using are further discussed below.

## 4.1 Movie review Dataset (MR)

This dataset contains reviews of different movies and those reviews are labeled as positive or negative. It has 10662 reviews with the average length of a sample being 20.[29]

## 4.2 Text Retrieval Conference (TREC)

This dataset contains multiple questions and the answer is classified into one of the six predefined categories. It has 5952 questions with an average length of 10. [30]

## 4.3 Customer Review (CR)

This dataset contains reviews of multiple products and these reviews are classified as positive or negative. It has 3775 reviews with an average sentence length of 19.[31]

## 4.4 Subjectivity (SUBJ)

This dataset contains sentences that have to be classified as either subjective or objective. It has 10000 sentences with an average length of 23. [32]

After training the model in these four datasets we compare it with the accuracy of the popular models used for text classification these days. The metric we will be using for comparing the performance of different models will be accuracy. The formula of accuracy is given below.

$$Accuracy = (TP + TN) \div N$$

In ten fold cross validation, we compute in all 10 splits and take the average of all the results to compute the final result. We will be using tenfold cross validation for MR, CR and SUBJ. For TREC there is a given split for test and train.

# CHAPTER 5

# EXPERIMENTS AND RESULTS

## 5.1 Experiments

The working of the model is written in the methodology section, in this section we implement the models on their respective datasets. There are a total of four models in which the model BERT+TCN_CNN, has been evaluated on four of the mentioned datasets, whereas the other models are evaluated only on the MR and CR dataset.

We will perform comparison only on the basis of accuracy. We will be taking a few models from our references to compare the accuracy of our model with. So we find out the accuracy of our model and compare it with some of our references, for a better understanding we use a charts to provide a visual understanding.

## 5.2 Results

After implementing or training the models on multiple dataset we use the test split in every validation to find out the accuracy and then average the accuracy in every split to obtain the final accuracy. We use column charts on every dataset with the y-axis having the accuracy percentage and the x-axis containing the name of the model and its respective reference. Using visual means to compare the accuracy of the models gives us a better understanding of how our model performs.

### 5.2.1 Accuracy comparison on CR dataset

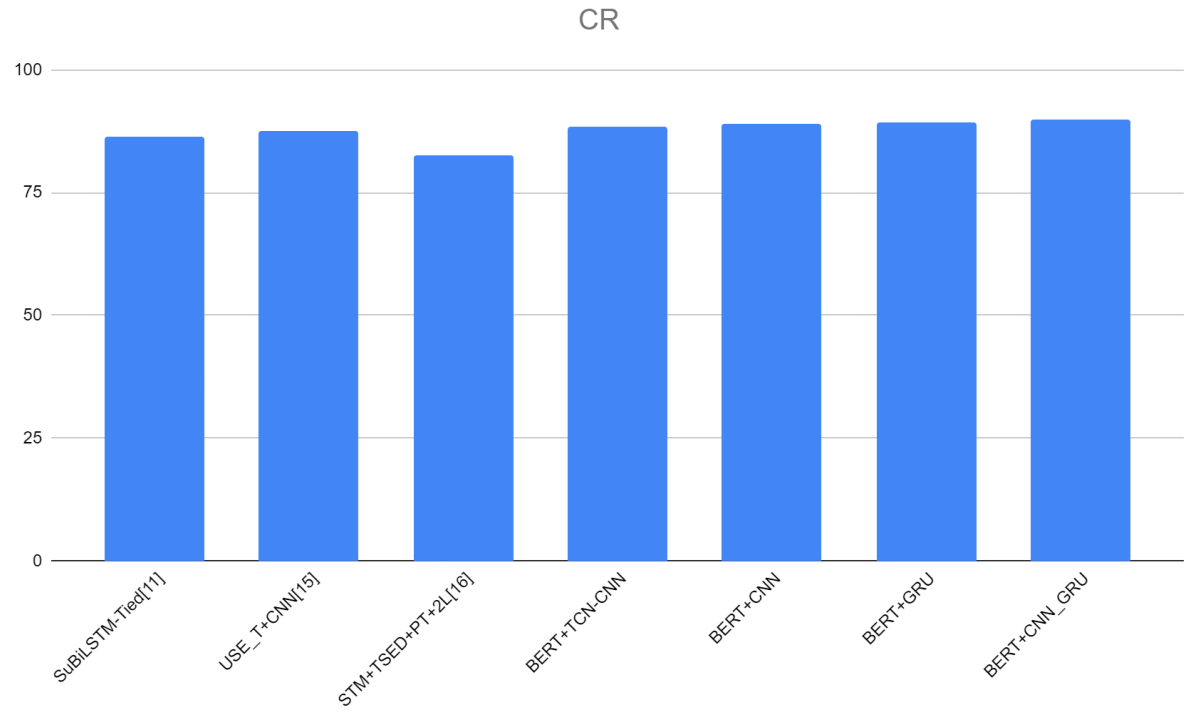


Fig. 5.1 Accuracy comparison on the CR dataset

Accuracy comparison on the CR dataset is shown in figure 5.1. We can observe from the column chart that BERT+CNN_GRU performs the best out of the compared models but the accuracy difference compared with other models that use static BERT embeddings is not that much.

### 5.2.2 Accuracy comparison on MR dataset

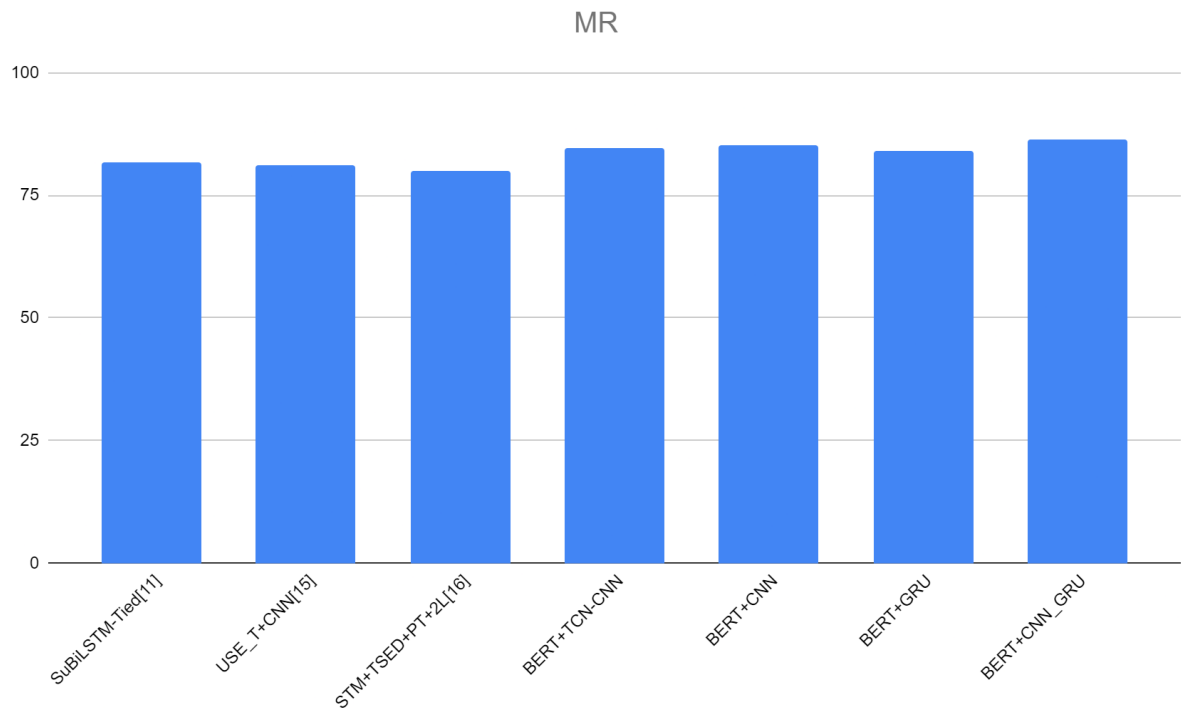


Fig. 5.2 Accuracy Comparison on the MR dataset

The accuracy comparison on the MR dataset is presented in Figure 5.2. The column chart clearly illustrates similarity to the CR dataset that BERT+CNN_GRU exhibits the highest performance among the models compared. However, it is worth noting that the difference in accuracy between BERT+CNN_GRU and other models utilizing static BERT embeddings is relatively small.

### 5.2.3 Accuracy Comparison on SUBJ dataset

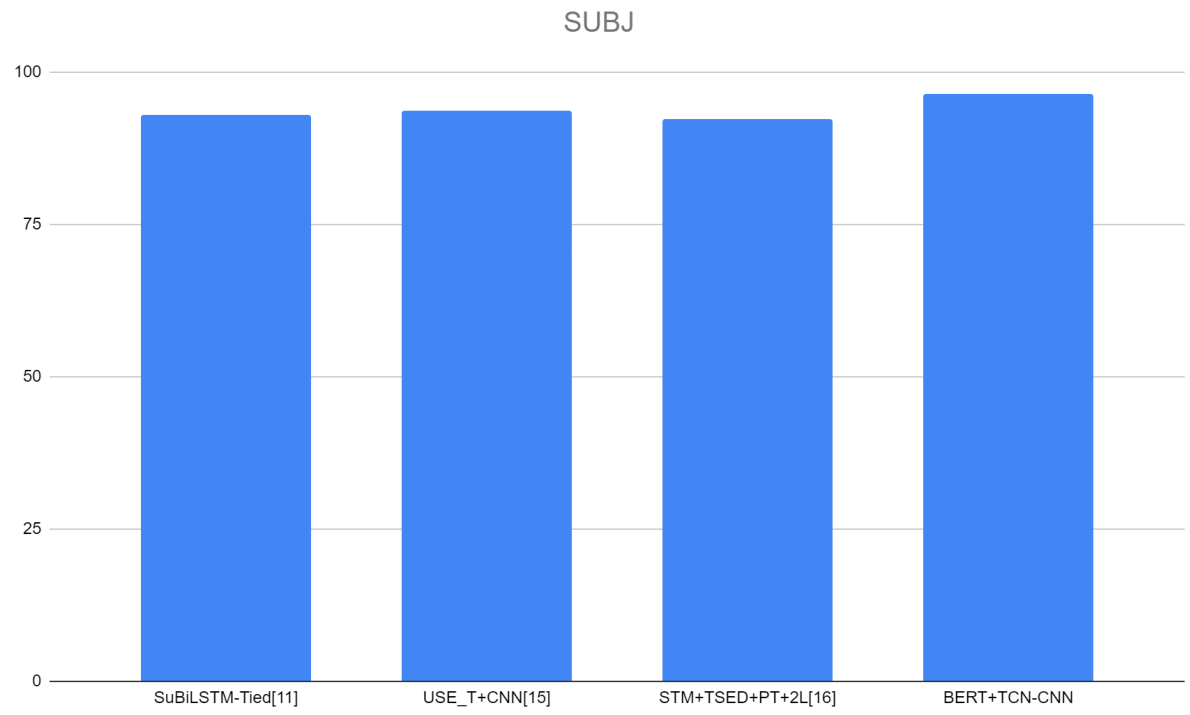


Fig. 5.3 Accuracy Comparison on SUBJ dataset

The accuracy comparison on the SUBJ dataset is given in the figure 5.3. It clearly exhibits that BERT+TCN_GRU performs the best out of the compared models.

### 5.2.4 Accuracy comparison on the TREC dataset

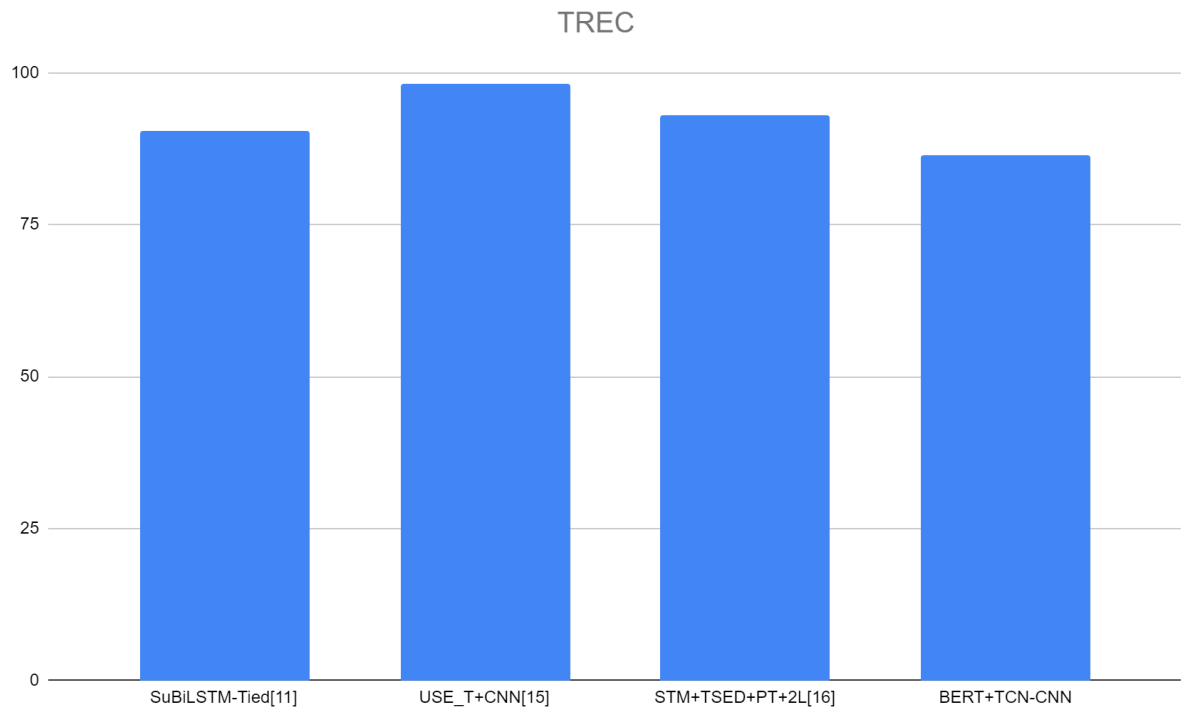


Fig. 5.4 Accuracy Comparison on the TREC dataset

The accuracy comparison on the TREC dataset is given in figure 5.4. The model BERT+TCN_CNN does not perform as well as the other models used to compare in the column chart. From this chart we can observe that BERT+TCN_CNN does not seem to perform that great on datasets with more than two classes.

# CHAPTER 6

# CONCLUSION AND FUTURE WORKS

In this paper, we created an ensemble of TCN and 1-D CNN and before that we used a pre-trained BERT model to convert the documents into BERT embeddings for feature extraction. This model has given comparable results to the popular methods used for text classification. Along with that we also evaluated multiple models when combined with static BERT embeddings. Static BERT embeddings are those when we don't change the weights of the pre-trained BERT model during the training phase. We evaluated the ensemble in four datasets CR, MR, SUBJ and TREC. Along with that we evaluated BERT+CNN, BERT+GRU, BERT+GRU_CNN in two datasets; MR and CR dataset.

We recommend the following as some suggestions for future experiments:

## 6.1 Using Dynamic BERT Embeddings

Instead of using static BERT embeddings, using dynamic BERT embeddings could give us better results. In static BERT embeddings there is no change of weights in the pre-trained BERT model but in dynamic the pre-trained BERT model is also further trained or fine tuned in the training phase. [33]

## 6.2 Explore other pre-trained models

We used static BERT embeddings in this paper, but instead of using a pre-trained BERT model we could use other models such as RoBERT[34][35], DeBERTa[36] or Universal Sentence Encoder(USE). This could further challenge the results obtained by this model.

## 6.3 Kernel size and filters

To improve the performance of TCN and 1D CNN models, it is possible to experiment with different hyperparameters, particularly the kernel size and filter values. By systematically varying and testing these hyperparameters, it is feasible to discover optimal values that can enhance the model's performance. The kernel size determines the receptive field of the convolutional layers. It defines the number of input elements considered by the convolutional filters at each step. Adjusting the kernel size allows the model to capture different patterns and dependencies in the input data. The number of filters determines the depth or width of the convolutional layers. Each filter is responsible for detecting specific features in the input data. By increasing or decreasing the number of filters, the model can be made more or less expressive, respectively.

## 6.4 Deeper Network

Increasing the number of hidden layers in a 1-D CNN or TCN can potentially improve the model's performance. Adding more hidden layers allows the network to learn and represent increasingly complex patterns and relationships in the data, which can enhance its ability to make accurate predictions or classifications.

By introducing additional hidden layers, the network gains more capacity to capture hierarchical features and abstract representations. Each hidden layer in a CNN or TCN learns higher-level features by combining and transforming the lower-level features learned in the preceding layers. This hierarchical learning process enables the model to extract more intricate and discriminative features from the input data.

# REFERENCES


[1] S. Kumar, A. Kumar, A. Mallik, R. R. Singh, “OptNet-Fake: Fake News Detection in Socio-Cyber Platforms Using Grasshopper Optimization and Deep Neural Network”. IEEE Transactions on Computational Social Systems, 2023

[2] A. Kumar, N. Aggarwal, S. Kumar (2022). “SIRA: a model for propagation and rumor control with epidemic spreading and immunization for healthcare 5.0”, Soft Computing, 27, 4307–4320 (2023)

[3] S. Kumar, N. Kumar, A. Dev, S Naorem. "Movie genre classification using binary relevance, label powerset, and machine learning classifiers" Multimedia Tools and Applications 82, no. 1 (2023): 945-968.

[4] S. S. Sengar, S. Kumar, P. Raina, M. Mahaliyan, "Bot detection in social networks based on multilayered deep learning approach". Sensors & Transducers, 244(5), 37-43, 2022

[5] Y. Saini, V. Bachchas, Y. Kumar, S. Kumar, “Abusive text examination using Latent Dirichlet allocation, self organizing maps and k means clustering”. In 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS) (pp. 1233-1238). IEEE, 2020

[6] A. Mallik, A. Khetarpal, S. Kumar,” ConRec: malware classification using convolutional recurrence”. Journal of Computer Virology and Hacking Techniques, 18(4), 297-313.

[7] Kowsari and Jafari Meimandi and Heidarysafa and Mendu and Barnes and Brown,Text Classification Algorithms: A Survey, arXiv:1904.08067

[8] T.Mikolov, I.Sutskever, K.Chen, G. S.Corrado, J.Dean. 2013. “Distributed representations of words and phrases and their compositionality”, In Advances in neural information processing systems, pages 3111–3119.

[9]J.Pennington, R.Socher, C.Manning. 2014. “Glove: Global vectors for word representation”, In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pages 1532–1543.

[10] M. E. Peters, M.Neumann, M.Iyyer, M.Gardner, C.Clark, K.Lee, L.Zettlemoyer. 2018. “Deep contextualized word representations”, arXiv preprint arXiv:1802.05365.

[11]S. Brahma. “Improved Sentence Modeling using Suffix Bidirectional LSTM”, arXiv, September, 2018.

[12] J.Devlin, M.Chang, K.Lee, K.Toutanova. 2018. “Bert: Pre-training of deep bidirectional transformers for language understanding”, arXiv preprint arXiv:1810.04805.
[13] D.Cer, Y.Yang, S.Kong, N.Hua, N.Limtiaco, R. St. John, N. Constant, M. Guajardo-Cespedes, S.Yuan, C.Tar, B.Strope, R.Kurzweil, “Universal Sentence Encoder”, arXiv, April, 2018.
[14]. Ma´ckiewicz, A.; Ratajczak, W. Principal components analysis (PCA). Comput. Geosci. 1993, 19, 303–342.
[15] D.Cer, Y.Yang, S.Kong, N.Hua, N.Limtiaco, R. St. John, N. Constant, M. Guajardo-Cespedes, S.Yuan, C.Tar, B.Strope, R.Kurzweil, “Universal Sentence Encoder”, arXiv, April, 2018.
[16] B.Shin, H.Yang, J.D. Choi (2019) “The pupil has become the master: teacher-student model-based word embedding distillation with ensemble learning” In: Proc Twenty-Eighth Int Jt Conf Artif Intell IJCAI-2019 2019-Augus:3439–3445. 10.24963/ijcai.2019/477
[17]Hochreiter, S.; Schmidhuber, J. Long Short-term Memory. Neural Comput. 1997, 9, 1735–1780.
[18] A.Conneau, H.Schwenk, L.Barrault, Y.Lecun. 2016. “Very deep convolutional networks for natural language processing”, arXiv
preprint arXiv:1606.01781, 2.
[19] R.Johnson, T.Zhang. 2017. “Deep pyramid
convolutional neural networks for text categorization”, In Proceedings of the 55th Annual Meeting of
the Association for Computational Linguistics (Volume 1: Long Papers), volume 1, pages 562–570.
[20] Rodolà, E.; Cosmo, L.; Litany, O.; Bronstein, M.M.; Bronstein, A.M.; Audebert, N.; Hamza, A.B.; Boulch, A.; Castellani, U.;
Do, M.N.; et al. Deformable Shape Retrieval with Missing Parts. In Proceedings of the Eurographics Workshop on 3D Object
Retrieval, Lyon, France, 23–24 April 2017; Pratikakis, I., Dupont, F., Ovsjanikov, M., Eds.; Eurographics Association: Geneva,
Switzerland, 2017.
[21]Gasparetto, A.; Minello, G.; Torsello, A. Non-parametric Spectral Model for Shape Retrieval. In Proceedings of the 2015
International Conference on 3D Vision, Lyon, France, 19–22 October 2015; pp. 344–352.
[22] Pistellato, M.; Bergamasco, F.; Albarelli, A.; Cosmo, L.; Gasparetto, A.; Torsello, A. Robust phase unwrapping by probabilistic
consensus. Opt. Lasers Eng. 2019, 121, 428–440.

[23] Jones, K.S. A statistical interpretation of term specificity and its application in retrieval. J. Doc. 1972, 28, 11–21.
[24]Huang, E.H.; Socher, R.; Manning, C.D.; Ng, A.Y. Improving Word Representations via Global Context and Multiple Word Prototypes. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, Jeju Island, Korea, 8–14 July 2012; Volume 1, pp. 873–882.
[25] K.Kowsari, M.Heidarysafa, D.E. Brown, K.J.Meimandi, L.E.Barnes, "Random Multimodel Deep Learning for Classification", arXiv, April, 2018.
[26] S.Bai, J.Z.Kolter, V.Koltun, "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling", arXiv, April, 2018.
[27]C.Lea, R.Vidal, A.Reiter, G.D. Hager, "Temporal convolutional networks: A unified approach to action segmentation." European Conference on Computer Vision. Springer, Cham, 2016.
[28]C. Henkel, "Temporal Convolutional Network", Kaggle, https://www.kaggle.com/christofhenkel/temporal-convolutional-network, February, 2021.
[29] B. Pang, L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales", In Proceedings of ACL '05, 2005.
[30] X.Li, D.Roth. 2002. "Learning question classifiers". In Proceedings of COLING, pages 1–7.
[31]M.Hu, B.Liu. 2004. "Mining and summarizing customer reviews", In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 168–177.
[32] B.Pang, L.Lee. 2004. " A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts", In Proceedings of ACL, pages 271–278.
[33] J.Devlin, M.Chang, K.Lee, K.Toutanova. 2018. "Bert: Pre-training of deep bidirectional transformers for language understanding", arXiv preprint arXiv:1810.04805.
[34]Lin, Y.; Meng, Y.; Sun, X.; Han, Q.; Kuang, K.; Li, J.; Wu, F. BertGCN: Transductive Text Classification by Combining GNN and BERT. In Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021; Association for Computational Linguistics:Stroudsburg, PA, USA, 2021; pp. 1456–1462.
[34]. Zaheer, M.; Guruganesh, G.; Dubey, A.; Ainslie, J.; Alberti, C.; Ontanon, S.; Pham, P.; Ravula, A.; Wang, Q.; Yang, L.; et al. Big Bird: Transformers for Longer Sequences. arXiv 2020, arXiv:2007.14062.
[36]He, P.; Liu, X.; Gao, J.; Chen, W. DeBERTa: Decoding-Enhanced BERT with Disentangled Attention. In Proceedings of the 2021 International Conference on Learning Representations (ICLR 2021), Vienna, Austria, 4–8 May 2021
[37]Sameer Anand et al. "Integrating node centralities, similarity measures, and ma-

chine learning classifiers for link prediction". In: Multimedia tools and applications 81.27 (2022), pp. 38593–38621.
[38]Sandeep Singh Sengar et al. "Bot detection in social networks based on multilayered deep learning approach". In: Sensors & Transducers 244.5 (2020), pp. 37–43
[39]Neeraj Bhat, Navneet Saggu, Sanjay Kumar, et al. "Generating visible spectrum images from thermal infrared using conditional generative adversarial networks". In: 2020 5th International Conference on Communication and Electronics Systems (ICCES). IEEE. 2020, pp. 1390–1394.
[40]Aditya Miglani et al. "A Literature Review on Brain Tumor Detection and Segmentation". In: 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS). IEEE. 2021, pp. 1513–1519.
[41]Yash Saini et al. "Abusive text examination using Latent Dirichlet allocation, self organizing maps and k means clustering". In: 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS). IEEE. 2020, pp. 1233–1238.