**Ho Quang Phuoc PROJECT NAME: Text classification using Long - Short Term Memory and Convolutional Neural Network 2022**

**THE UNIVERSITY OF DANANG**

**DANANG UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**FACULTY OF INFORMATION TECHNOLOGY**

**GRADUATION PROJECT THESIS**

**MAJOR: INFORMATION TECHNOLOGY**

**SPECIALTY: SOFTWARE TECHNOLOGY**

**PROJECT TITLE:**

**TEXT CLASSIFICATION USING LONG - SHORT TERM MEMORY AND CONVOLUTIONAL NEURAL NETWORK**

Instructor: **PhD. DANG HOAI PHUONG**

Student: **HO QUANG PHUOC**

Student ID: **102170180**

Class: **17T3**

**Da Nang, 03/2022**

**THE UNIVERSITY OF DANANG**

**DANANG UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**FACULTY OF INFORMATION TECHNOLOGY**

**GRADUATION PROJECT THESIS**

**MAJOR: INFORMATION TECHNOLOGY**

**SPECIALTY: SOFTWARE TECHNOLOGY**

**PROJECT TITLE:**

**TEXT CLASSIFICATION USING LONG - SHORT TERM MEMORY AND CONVOLUTIONAL NEURAL NETWORK**

Instructor: **PhD. DANG HOAI PHUONG**

Student: **HO QUANG PHUOC**

Student ID: **102170180**

Class: **17T3**

**Da Nang, 03/2022**

|  |  |
| --- | --- |
| DA NANG UNIVERSITY  **UNIVERSITY OF SCIENCE AND TECHNOLOGY**  FACULTY INFORMATION TECHNOLOGY | **SOCIALIST REPUBLIC OF VIETNAM**  Independence - Freedom - Happiness |

**GRADUATION PROJECT COMMENT**

1. **General information:**
2. Student name: Ho Quang Phuoc
3. Class: 17T3 Student ID: 102170180
4. Topic title: Text classification using Long - Short Term Memory and Convolutional Neural Network
5. Instructor: Dang Hoai Phuong Academic title/ degree: PhD
6. **Reviews of graduation project**
7. About the urgency, novelty, usability of the topic: (2 points)

……………………………………………………………………………………

……………………………………………………………………………………

1. About the results of solving the tasks required by the project: (4 points)

……………………………………………………………………………………

……………………………………………………………………………………

1. About the form, structure, and layout of the graduation project: (2 points)

……………………………………………………………………………………

……………………………………………………………………………………

1. The topic includes scientific value/article/problem solving of the enterprise or school: (1 point)

……………………………………………………………………………………

……………………………………………………………………………………

1. Existing shortcomings need to be supplemented or modified:

……………………………………………………………………………………

……………………………………………………………………………………

1. **Spirit and attitude of the student (1 point):**

…………………………………………………………………………………

…………………………………………………………………………………

1. **Evaluation:**
2. Evaluation point: …./10
3. Suggest: Defense permitted/ Edit to defend/ Defense not permitted

Da Nang, date……month 03, 2022

**Instructor**

**COMMITTEE REPORT OF DEFENCE RESULTS**

…………………………………………………………………………………….

…………………………………………………………………………………….

…………………………………………………………………………………….

…………………………………………………………………………………….

…………………………………………………………………………………….

…………………………………………………………………………………….

…………………………………………………………………………………….

…………………………………………………………………………………….

…………………………………………………………………………………….

…………………………………………………………………………………….

…………………………………………………………………………………….

…………………………………………………………………………………….

…………………………………………………………………………………….

…………………………………………………………………………………….

…………………………………………………………………………………….

…………………………………………………………………………………….

…………………………………………………………………………………….

…………………………………………………………………………………….

…………………………………………………………………………………….

…………………………………………………………………………………….

…………………………………………………………………………………….

…………………………………………………………………………………….

**SUMMARY**

Topic: Text classification using Long - Short Term Memory and Convolutional Neural Network

Student name: Ho Quang Phuoc

Student ID: 102170180 Class: 17T3

In this project, I do research about Long - Short Term Memory (LSTM) and Convolutional Neural Network (CNN) in order to utilize the advantage of each model. Then I build some models which combines both, adding some Batch Normalization and Dropout layers to enhance the performances of the models.

Combinations can be divided into three types. I place the CNN before the LSTM layer in the first model. I invert the order of the elements in the second model. The input is fed to both CNN and LSTM in the final model, after which I concatenate the outputs and pass it to a classification layer. I also test models that just employ CNN or LSTM as a learning algorithm. Google Colab is used to run all the tests. "The Corona Virus Tweets" is the data set I used in this study. I clean the dataset using some basic processes.

After receiving the findings, I assess all the models above using four criteria, including accuracy, precision, recall, and F1-score, and then make some comparisons and draw conclusions.

|  |  |
| --- | --- |
| DA NANG UNIVERSITY  **UNIVERSITY OF SICIENCE AND TECHNOLOGY**  FALCUTY: INFORMATION TECHNOLOGY | **THE SOCIALIST REPUBLIC OF VIETNAM**  Independence - Freedom - Happiness |

**GRADUATION PROJECT REQUIREMENTS**

Student Name: Ho Quang Phuoc Student ID : 102170180

Class: 17T3 Faculty: Information Technology Major: Software Technology

1. *Topic title:* Text classification using Long - Short Term Memory and Convolutional Neural Network.
2. *Project topic:* ☐*has signed intellectual property agreement for final result*
3. *Initial figure and data:*

* The “Cororavirus tweets NLP – Text Classification” data set

*Content of the explanations and calculations:*

* Word2vec
* Activation function
* Softmax and loss function
* Optimizers
* Attention Layer
* Convolution Neural Network
* Long – Short Term Memory

1. *Drawings, charts (specify the types and sizes of drawings):*
2. *Name of instructor: PhD. Dang Hoai Phuong*
3. *Date of assignment : 08/10/2021*
4. *Date of completion : 03/03/2022*

|  |  |
| --- | --- |
|  | *Đà Nẵng, date month 03 year 2022* |
| **Head of Division**…………………. | **Instructor** |

**PREFACE**

Firstly, I would like to express my gratitude to all of the teachers in the Information Technology faculty for providing me with excellent lectures that have enabled me to accomplish this project.

Secondly, I would like to express my gratitude to my instructor, teacher Dang Hoai Phuong, who was always willing to offer me helpful advice and suggestions during the duration of this project.

Despite my meticulous preparation, the project must contain errors and defects due to my lack of knowledge and experience. As a result, I am grateful to hear comments and suggestions from all the faculty members. Without your help, I won't be able to improve my project.

Thank you.

Student Performed

Ho Quang Phuoc

**ASSURANCE**

I am assured that:

1. This project is my original work, which I completed with the help of my tutor, PhD.Dang Hoai Phuong.

2. All of the references used in my project are properly credited in the Reference section.

3. If there is any false information, I will take full responsibility.

Student Performed

Ho Quang Phuoc

**TABLE OF CONTENT**

Summary

Thesis mission

Preface i

Assurance ii

Table of contents iii

List of figures and tables v

List of symbols, acronym vi

[CHAPTER 1: THE OVERVIEW OF TEXT CLASSIFICATION 2](#_Toc97500785)

[1.1. Introduction to Text classification 2](#_Toc97500786)

[1.2. Some analysis 3](#_Toc97500787)

[1.2.1. Text representation 3](#_Toc97500788)

[1.2.2. Data cleaning 6](#_Toc97500789)

[1.2.3. Recurrent Neural Network 6](#_Toc97500790)

[1.2.4. Suggested solution 7](#_Toc97500791)

[1.3. Conclusion 8](#_Toc97500792)

[CHAPTER 2: THEORETICAL BASIS 9](#_Toc97500793)

[2.1. Word2vec 9](#_Toc97500794)

[2.2. Activation function 11](#_Toc97500795)

[2.2.1. Definition 11](#_Toc97500796)

[2.2.2. Common activation functions 12](#_Toc97500797)

[2.3. Softmax and loss function 15](#_Toc97500798)

[2.3.1. Softmax 15](#_Toc97500799)

[2.3.2. Loss function 16](#_Toc97500800)

[2.4. Optimizers 18](#_Toc97500801)

[2.4.1. Introduction 18](#_Toc97500802)

[2.4.2. Equation to update the parameter 18](#_Toc97500803)

[2.5. Batch Normalization 19](#_Toc97500804)

[2.5.1. Introduction 19](#_Toc97500805)

[2.5.2. Computing Batch Normalization 19](#_Toc97500806)

[2.6. Dropout 20](#_Toc97500807)

[2.7. Attention Layer 21](#_Toc97500808)

[2.7.1. Introduction 21](#_Toc97500809)

[2.7.2. Mechanism 21](#_Toc97500810)

[2.8. Convolutional Neural Network 22](#_Toc97500811)

[2.8.1. Introduction 22](#_Toc97500812)

[2.8.2. Kernel 22](#_Toc97500813)

[2.8.3. Stride 22](#_Toc97500814)

[2.8.4. Pooling 23](#_Toc97500815)

[2.9. Recurrent Neural Network and Long – Short Term Memory 24](#_Toc97500816)

[2.9.1. RNN 24](#_Toc97500817)

[2.9.2. LSTM 25](#_Toc97500818)

[2.10. Conclusion 30](#_Toc97500819)

[CHAPTER 3: IMPLEMENTATION AND EVALUATION 31](#_Toc97500820)

[3.1. Dataset 31](#_Toc97500821)

[3.1.1. Some information of the data set 31](#_Toc97500822)

[3.1.2. Preprocessing data 33](#_Toc97500823)

[3.2. Implementation 34](#_Toc97500824)

[3.2.1. Input 34](#_Toc97500825)

[3.2.2. LSTM 34](#_Toc97500826)

[3.2.3. Attention 34](#_Toc97500827)

[3.2.4. Convolutional layer 34](#_Toc97500828)

[3.2.5. Pooling 34](#_Toc97500829)

[3.2.6. Classification layer 35](#_Toc97500830)

[3.2.7. Loss and optimizer 35](#_Toc97500831)

[3.3. Performance and Evaluation 35](#_Toc97500832)

[3.3.1. Performance evaluation metrics 35](#_Toc97500833)

[3.3.2. Evaluation 36](#_Toc97500834)

[3.4. Comparison of Experimental Results 45](#_Toc97500835)

[3.5. Conclusion 47](#_Toc97500836)

[CONCLUSION 48](#_Toc97500837)

[REFERENCE 49](#_Toc97500838)

**LIST OF FIGURES**

[Figure 1.1. A diagram of machine learning for automatic text classification 2](#_Toc97544942)

[Figure 1.2. The sentence's one-hot encoding representation 4](#_Toc97544943)

[Figure 1.3. The sentence's BOW representation 4](#_Toc97544944)

[Figure 1.4. Example of Word2vec 6](#_Toc97544945)

[Figure 1.5. The architecture of proposed model 8](#_Toc97544946)

[Figure 2.1. The CBOW and Skip-gram model 9](#_Toc97544947)

[Figure 2.2. The architecture of the skip-gram model 10](#_Toc97544948)

[Figure 2.3. The lookup table 11](#_Toc97544949)

[Figure 2.4. The graph of sigmoid and its derivative 12](#_Toc97544950)

[Figure 2.5. The graph of tanh and its derivative 13](#_Toc97544951)

[Figure 2.6. The graph of ReLU and its derivative 14](#_Toc97544952)

[Figure 2.7. The example of softmax function 15](#_Toc97544953)

[Figure 2.8. The example of a network using softmax 16](#_Toc97544954)

[Figure 2.9. The example of dropout 21](#_Toc97544955)

[Figure 2.10. The convolution with kernel size 2x2 and stride 1 22](#_Toc97544956)

[Figure 2.11. The maxpooling and average pooling 23](#_Toc97544957)

[Figure 2.12. The architecture of RNN 24](#_Toc97544958)

[Figure 2.13. The LSTM cell 25](#_Toc97544959)

[Figure 2.14. The forget gate 26](#_Toc97544960)

[Figure 2.15. The input gate 26](#_Toc97544961)

[Figure 2.16. The cell state 27](#_Toc97544962)

[Figure 2.17. The output gate 27](#_Toc97544963)

[Figure 2.18. The detailed LSTM cell 28](#_Toc97544964)

[Figure 3.1. Some records of the training data set 31](#_Toc97544965)

[Figure 3.2. The class distribution on the whole data set 32](#_Toc97544966)

[Figure 3.3. The class distribution on each data set 32](#_Toc97544967)

[Figure 3.4. The CNN loss 37](#_Toc97544968)

[Figure 3.5. The CNN accuracy 38](#_Toc97544969)

[Figure 3.6. The LSTM loss 39](#_Toc97544970)

[Figure 3.7. The LSTM accuracy 39](#_Toc97544971)

[Figure 3.8. The CNN-LSTM loss 40](#_Toc97544972)

[Figure 3.9. The CNN-LSTM accuracy 41](#_Toc97544973)

[Figure 3.10. The LSTM-CNN loss 42](#_Toc97544974)

[Figure 3.11. The LSTM-CNN accuracy 42](#_Toc97544975)

[Figure 3.12. The CNN+LSTM loss 43](#_Toc97544976)

[Figure 3.13. The CNN+LSTM accuracy 44](#_Toc97544977)

[Figure 3.14. A comparison of five models' validation loss 45](#_Toc97544978)

[Figure 3.15. A comparison of five models' validation accuracy 45](#_Toc97544979)

[Figure 3.16. The number of trainable parameters in 5 models 46](#_Toc97544980)

[Figure 3.17. The comparison results between 5 models 46](#_Toc97544981)

**LIST OF TABLES**

[Table 3.1. The data set information 31](#_Toc96855950)

[Table 3.2. The class distribution 32](#_Toc96855951)

[Table 3.3. The numbers of tweet in each sentiment 33](#_Toc96855952)

[Table 3.4. Number of tweets in each data set 34](#_Toc96855953)

[Table 3.5. The layer structure of proposed model 35](#_Toc96855954)

[Table 3.6. The proposed model parameters structure 35](#_Toc96855955)

[Table 3.7. The number of parameters of 5 models 36](#_Toc96855956)

**LIST OF SYMBOLS, ACRONYM**

SYMBOL:

* : Hadamard product (element-wise product).
* : Sigmoid function

ACRONYM:

* CNN: Convolutional Neural Network.
* LSTM: Long - Short Term Memory.
* CNN – LSTM: Model in which input data is fed to CNN and then LSTM.
* LSTM – CNN: Model in which input data is fed to LSTM and then CNN.
* CNN + LSTM: Model in which input data is fed to both CNN and LSTM, then the outputs from two models are concatenated.

**INTRODUCTION**

Text classification using Deep Learning is becoming increasingly prominent in text classification tasks. The goal of this project is to investigate the use of Long - Short Term Memory and Convolutional Neural Networks, as well as their combinations, in developing a model to handle a text categorization problem.

Experiments and evaluation of model performance based on a specific data set are the methods employed in this project.

The first chapter provides an overview of text classification in general as well as some examination of relevant topics. The proposed model, which has the best performance, is presented at the end of the chapter.

The second chapter contains the intricacies of the theory's foundation. In this chapter, I go over the mathematics, formulas, algorithms, and approaches that are employed in the project.

In the third chapter, I describe my work's performance and appraisal.

The conclusion and references sections summarize the findings and provide a list of the papers and internet sources used in this project.

# THE OVERVIEW OF TEXT CLASSIFICATION

## Introduction to Text classification

In today's world, the vast amount of data accumulated over many years by businesses, organizations, and cooperatives is proving valuable in a wide range of fields, from economics to politics to culture and society.

Artificial Intelligence (AI) has exploded in popularity as a result of the abundance of large data. Deep Learning (ML) is one of the most widely used disciplines of AI, mainly to the advancement of the graphics processing unit's rapid and parallel computation speed (GPU). Deep learning has made significant progress in the fields of computer vision and speech recognition [2], [3], but it is still advancing in natural language processing.

Text classification research in Natural Language Processing has lately progressed to state-of-the-art (SOTA), making text classification on massive documents more important than ever.

Diagram

Description automatically generated

Figure 1.1. A diagram of machine learning for automatic text classification [1]

## Some analysis

### Text representation

One of the most basic steps for most natural language processing tasks is to convert words into numbers for machines to understand and decode patterns of text within a language. This step plays a significant role in deciding features for the machine learning model or the algorithm.

Text representation can be classified into two sections:

* Discrete text representation
* Distributed/Continuous text representation

#### Discrete text representation

In discreate text representation, words are presented by their corresponding indexes to their position in a dictionary from a larger corpus or corpora.

Two popular kinds of discrete text representation are:

##### One-hot encoding

By using one-hot encoding, each word is represented by a vector in which the position of the word is assigned the value 1, and the rest receive the value 0.

For example, we have the corpus {I, do, like, cat, but, not, dog} and the sentence “I like dog, but I do not like cat.”

Then each word in the sentence is represented as follow:

A picture containing icon

Description automatically generated

Figure 1.2. The sentence's one-hot encoding representation

An obvious advantage of one-hot encoding is that it is easy to understand and implement. However, it can cause explosion in feature space if number of categories are very high. Because each word is encoded based on its position in the dictionary, so people cannot measure the importance of a word as well as the relationship among words in a sentence. Another drawback is that the length of an array of a word depends on the vocabulary size so that it can be memory and computationally expensive because of the high dimensional sparse matrix representation.

##### Bag-of-words representation (BOW)

This kind of representation turns text into fixed-length vectors based on the frequency of each word in the vocabulary. This process is often considered as vectorization.

For example, we have the vocabulary {I, do, like, cat, but, not, dog} and the sentence “I like cat and I do not like dog.” The frequency of words in the sentence is as followed:

Table

Description automatically generated

Figure 1.3. The sentence's BOW representation

So, the sentence is represented as [2, 1, 2, 1, 1, 1, 1].

The advantage of this type of representation of the length of the encoded vector is the length of the dictionary. However, the intuition that high-frequency words are more important or give more information about the sentence fails when it comes to stop words like “is, the, I” and then the corpus is context-specific. For example, in a corpus about covid-19, the word coronavirus may not add a lot of value.

#### Distributed text representation

When a word's representation is not independent or mutually exclusive of another word's, it is called distributed text representation, and its configurations can represent a variety of metrics and concepts in data. As a result, the information about a word is dispersed over the vector in which it is represented. This differs from discrete representation, in which each word is regarded distinct and unrelated to the others.

The following are some of the most often used distributed text representations [4]:

* Word2vec [5]
* GloVe [6]

In the scope of this work, I used Word2vec and therefore just focus on this algorithm. Word2Vec's efficacy stems on its ability to combine vectors of related words together. Word2Vec can create solid guesses about a word's meaning based on its appearances in the text if given a large enough dataset. Word connections with other words in the corpus are derived from these estimations.

For example, words like “King” and “Queen” are remarkably similar to one another. A close approximation of word similarities can be found by performing algebraic operations on word embeddings [7]. The computation can be seen as below.

Table

Description automatically generatedDiagram

Description automatically generated

Figure 1.4. Example of Word2vec

We have

The success of Word2vec may be attributed to two primary designs. The architectures of the skip-gram and CBOW. The mechanism of Word2vec is represented in the first part of Chapter 2.

### Data cleaning

Text data contains a lot of noise, such as grammar issues, erroneous words, and slang, as compared to picture data, therefore cleaning it is one of the most critical processes before feeding it to the model.

### Recurrent Neural Network

The classic neural network is made up of three layers: the input layer, the hidden layer(s), and the output layer. It can be shown that the network's input and output are independent of one another. As a result, it is unsuitable for issues where the input is in sequence data [8].

The RNN (recurrent neural network) was created with the goal of leveraging memory to preserve knowledge from earlier processes to provide the most accurate prediction possible. They are, without a doubt, in use. RNNs have had remarkable success in the previous several years when applied to several tasks, including speech recognition, language modeling, translation, and picture captioning.

One of the allures of RNNs is the possibility of connecting earlier data to the current job, such as using previous video frames to inform comprehension of the current frame. RNNs, on the other hand, can't always achieve this, or, to put it another way, it depends.

Sometimes all we need to do is glance at recent data to complete the work at hand. Consider a language model that tries to anticipate the next word based on the ones that came before it. RNNs can learn to use past knowledge if the distance between relevant information and the place or term we wish to forecast is minimal.

However, there are times when extra information is required. If individuals wish to anticipate the last word in the sentence "I was born and raised in Vietnam, thus I can speak fluently...", for example. The appropriate word to fill in the blank should be Vietnamese. To do so, it is necessary to understand the historical backdrop of Vietnam. Then there is a good chance that the gap between relevant data and where it is required will widen dramatically. RNNs, on the other hand, become unable to learn to connect the dots as the distance widens.

Long - Short Term Memory networks (LSTM) - a kind of RNN capable of learning long-term dependencies - were created to address this problem.

### Suggested solution

Because of its ability to extract local features, the convolutional neural network (CNN) is one of the most popular architects in computer vision; however, it lacks the ability to learn sequential correlations; on the other hand, the LSTM is specialized for sequential modeling but unable to extract features in a parallel manner. To take use of each network's strengths, I opted to merge these two architectures into LSTM-CNN for text categorization tasks. The hybrid model may be seen in the diagram below.

A picture containing chart

Description automatically generated

Figure 1.5. The architecture of proposed model

In the second chapter, I'll go through the mechanism in depth, as well as provide highly thorough math and modeling explanations.

## Conclusion

This chapter presents an overview of Text Classification using Deep Learning in general, as well as some analyses of relevant topics. The proposed model, which achieves the maximum performance in 5 models, is represented at the end of the chapter.

# THEORETICAL BASIS

## Word2vec

Word2vec is a word-representation approach based on prediction. The vector representation of a word is called a word embedding. Each word has a set vector size that captures its semantic and syntactic relationships with other words [4].

The architecture of word2vec is a single hidden layer network with a short depth. The embedding of the word is represented by the weights of the hidden layer, which we change using a loss function (normal backprop).

The vector representation is created by word2vec using two techniques:

* CBOW
* Skip-Gram

Diagram

Description automatically generated

Figure 2.1. The CBOW and Skip-gram model [9]

The distributed representations of context are integrated in the CBOW model to predict the middle word. In Skip-gram model, it predicts the context words using the main word. These two designs' algorithms are quite similar. As a result, I'll describe Word2vec using the Skip-gram to make it easier to understand.

To begin, a word is transformed to one-hot vectors, which are essentially a vector of the same length as the vocabulary's size, filled with zeros except for the index that represents the word, which is assigned one. The word embeddings are the weights of the hidden layer, which is a conventional fully linked layer. From the vocabulary, the output layer provides probabilities for the target words.

Shape, circle

Description automatically generated

Figure 2.2. The architecture of the skip-gram model [9]

The word vectors are the rows of the hidden layer weight matrix.

A picture containing bar chart

Description automatically generated

Figure 2.3. The lookup table [9]

I utilized pre-trained vectors that were trained on a portion of the Google News dataset (about 100 billion words) for step text vectorization. For 3 million words and phrases, the model has 300-dimensional vectors [10].

## Activation function

### Definition

Activation functions are functions used in neural networks to computes the weighted sum of input and biases, of which is used to decide if a neuron can be fired or not. It manipulates the presented data through some gradient processing usually gradient descent and afterwards produce an output for the neural network, that contains the parameters in the data [11].

### Common activation functions

#### Sigmoid

Because it is a non-linear function, the sigmoid function is the most often employed activation. The sigmoid function changes data in the 0 to 1 range. It is defined as:

Derivative

Chart, line chart

Description automatically generated

Figure 2.4. The graph of sigmoid and its derivative

#### Tanh

Tanh function is a continuous and differentiable function with values ranging from -1 to 1. The gradient of the tanh function is steeper than that of the sigmoid function. Tanh is chosen over sigmoid function because it provides gradients that do not have to change in a certain direction and is zero centered. It can be defined as:

Derivative

Line chart

Description automatically generated with medium confidence

Figure 2.5. The graph of tanh and its derivative

#### ReLU

The rectified liner unit, or ReLU, is a non-linear activation function commonly employed in neural networks. The advantage of employing ReLU is that not all neurons are stimulated at the same time. This means that a neuron will only be destroyed when the linear transformation output is zero. It can be mathematically defined as:

Derivative

Chart, line chart

Description automatically generated

Figure 2.6. The graph of ReLU and its derivative

## Softmax and loss function

### Softmax

The softmax function is a sigmoid function that has been combined with other sigmoid functions. Because a sigmoid function gives values in the range of 0 to 1, they might be interpreted as probability of data points in a specific class. Unlike sigmoid functions, which are utilized for binary classification, the softmax function may be used to solve multiclass classification issues. The probability is returned by the function for each data point in each of the various classes. It can be written as:

Where K is the length of vector .

Example: supposed we have a vector x with three values:

Then feed into softmax function, we have the result as below.

Diagram

Description automatically generated

Figure 2.7. The example of softmax function

### Loss function

#### Loss

Loss is the value used to measure the difference between predict output and the actual output.

In this project, I use “Categorical Crossentropy” loss.

#### Categorical Crossentropy

At the classification layers, to make it simpler, supposed we have two output classes, equivalent to two nodes at dense layer. The classification layer can be visualized as below.

Diagram

Description automatically generated

Figure 2.8. The example of a network using softmax

Derivate

With , we have:

With , we have:

## Optimizers

### Introduction

Optimizers are algorithms or methods used to change the attribute of the neural network such as weights and learning rate to reduce the losses.

Adam works with momentums of first and second order.

is the biased first moment.

is the biased second moment.

is the bias-corrected first moment which is the mean of the gradients.

is the bias-corrected second raw moment which is the uncentered variance of the gradients.

### Equation to update the parameter

The value for is 0.9, for is 0.999, for is [12].

is the learning rate, in this project, I use .

## Batch Normalization

### Introduction

Batch Normalization is an approach that speeds up and stabilizes the training of deep neural networks. It entails utilizing the current batch’s first and second statistical moments (mean and variance) to normalize activation vectors from hidden layers.

### Computing Batch Normalization

Batch Normalization is computed differently during the training and the testing phase.

* In the training phase

Input data for a node in Batch Normalization layer with m is the mini-batch size. At each hidden layer, Batch Normalization transforms the signal as follow [13]:

+ Compute mean and variance

+ Normalize

is a very small number

+ Scale and shift

and are two learning parameters.

* In the testing phase

Unlike the training phase, the evaluation phase may not have a complete batch to feed into the model. To solve this issue, it calculates the estimated and in all training mini batch. Supposed there are n mini batch with size m, then the estimated and are calculates as follow:

## Dropout

When training a neural network, overfitting is an issue. It's possible that the model obtains extremely high accuracy on the initial training data but very poor accuracy on the testing data because of this.

Overfitting may be addressed in a variety of methods, with dropout being one among them.

The concept of dropout is that we forbid some neurons and give chance to others. It works like this [14]:

* We assign a dropout rate, which is the fraction of neurons that fail to function (e.g., 10 % of neurons).
* We eliminate random neurons according to a predetermined proportion at each step.
* The final output is calculated using the combined data of the remaining neurons.

Diagram

Description automatically generated

Figure 2.9. The example of dropout [14]

## Attention Layer

### Introduction

Attention can be considered as a mechanism that can help a neural network to memorize long sequences of the information or data. In this work, I create an attention mechanism to create a shortcut between the input and the context vector where weights of the shortcut connection can be changeable for every input.

### Mechanism

In the proposed model, I use the customed attention layer after the LSTM layer. Supposed all vector forming matrix is hidden states of LSTM layer. All the equations below are referenced from this paper [15].

The context vector depends on a sequence of hidden states . The context vector is then computed as a weighted sum of these hidden states :

The weighted of each hidden state is computed by:

Where , with is the weight of customed attention layer and is the bias.

## Convolutional Neural Network

### Introduction

Over the last decade, Convolutional Neural Networks have achieved breakthroughs in a range of domains connected to pattern identification, ranging from image processing to speech recognition. The greatest advantage of CNNs is that they reduce the number of parameters in Artificial Neural Networks (ANN). This accomplishment has motivated academics and developers to consider bigger models to perform challenging tasks that were previously impossible to solve with traditional ANNs [16].

### Kernel

Kernel in CNN is a filter that extracts the features from the input data. The kernel is a matrix that traverses across the input data, performs a dot product with the sub-region of data, and return a matrix of dot products as the output.

### Stride

The stride of a kernel is the number of pixels the kernel travels over the input matrix.

Diagram

Description automatically generated

Figure 2.10. The convolution with kernel size 2x2 and stride 1

### Pooling

Pooling layers, also known as down sampling, reduce dimensionality by reducing the number of parameters in the input. Like the convolutional layer, the pooling procedure sweeps a filter across the whole input, but this filter does not include any weights. Instead, the kernel employs a function to populate the output array with receptive field values. Pooling may be divided into two categories:

* Max pooling: as the filter moves across the input, it selects the pixel with the maximum value to send to the output array. As an aside, this approach tends to be used more often compared to average pooling.
* Average pooling: as the filter moves across the input, it calculates the average value within the receptive field to send to the output array.

Chart

Description automatically generated

Figure 2.11. The maxpooling and average pooling

While a lot of information is lost in the pooling layer, it also has several benefits to the CNN. They help to reduce complexity, improve efficiency, and limit risk of overfitting.

## Recurrent Neural Network and Long – Short Term Memory

### RNN

The primary principle behind RNN is to utilize information in a sequential order. All inputs and outputs in classic neural networks are independent of one another. That is, they are not linked to one another in a chain. However, many issues are incompatible with these models. For example, if we want to guess the next word in a phrase, we must first understand how the previous words appear. RNNs are referred to as regressions because they do the identical task for all components of a sequence, with the outcome based on both prior calculations. In other words, RNN can recall information that has already been calculated.

A screenshot of a computer

Description automatically generated with low confidence

Figure 2.12. The architecture of RNN [8]

RNN can theoretically use information from a very long text, but it can only recall a few prior stages in practice. The next part will introduce about Long – Short Term Memory (LSTM), a special kind of RNN, being capable of learning long-term dependencies.

### LSTM

#### Introduction

Long - Short Term Memory networks (LSTMs) were introduced by Hochreiter & Schmidhuber (1997). It is a novel recurrent network architect in conjunction with an appropriate gradient-based learning algorithm [17]. LSTMs are explicitly designed to avoid the long-term dependency problem.

A screenshot of a game

Description automatically generated with medium confidence

Figure 2.13. The LSTM cell [8]

: the input at timestamp t

: the hidden state

Diagram

Description automatically generated

#### Gates

##### Forget gate

Diagram

Description automatically generated

Figure 2.14. The forget gate [8]

Using a sigmoid layer, the forget gate layer determines what information is discarded from the cell state. It examines and and returns a number between 0 and 1 for each number in the cell state . A 1 indicates that the information should be kept totally, whereas a 0 indicates that is should be completely removed.

##### Input gate

Diagram

Description automatically generated

Figure 2.15. The input gate [8]

In the next step is to assess new information to see if it can be stored in the cell state. There are two components to this. The input gate layer, a sigmoid layer, chooses which values will be updated. After that, a tanh layer generates , a vector of fresh candidate values that might be added to the state. These two values are then concatenated in the following step to form a state update.

##### Cell state

Diagram

Description automatically generated with medium confidence

Figure 2.16. The cell state [8]

The previous cell state, , is replaced with the new cell state . Because the preceding phases have already chosen what to do, this step will simply process the values.

The former state is multiplied by to forget the items that were previously determined to be forgotten. Then use to combine it. This is the new set of candidate values, scaled by how much each state value was updated.

##### Output gate

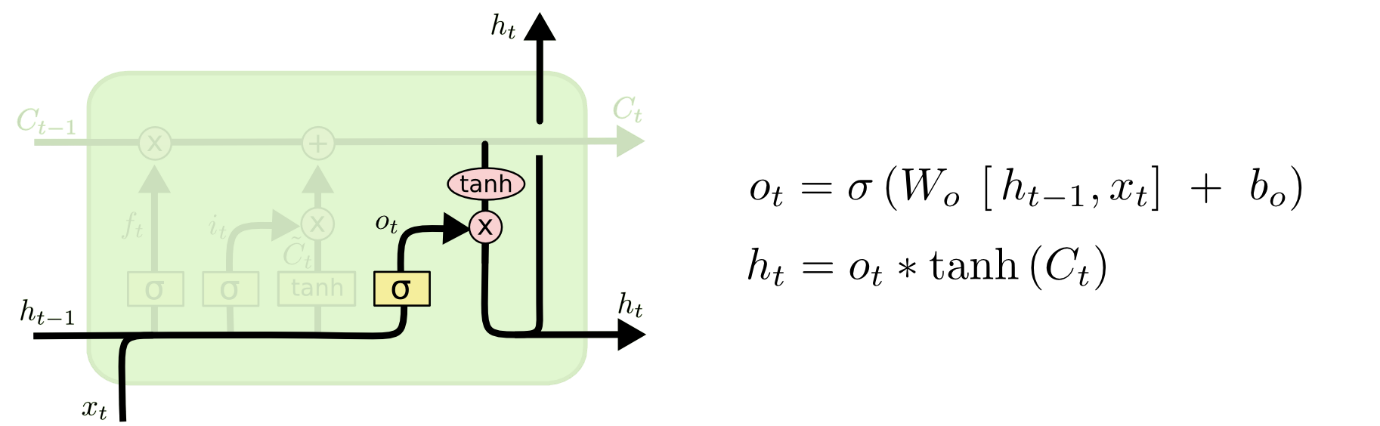


Figure 2.17. The output gate [8]

The output gate is where the process ends. This output will be filtered and will be dependent on the cell status. A sigmoid layer is first utilized to determine which elements of the cell state are output. The cell state is then placed via activation to push the values between -1 and 1, and then multiplied by the output of the sigmoid gate, ensuring that only the sections we choose to output are output.

#### Backpropagation

To make it more understandable, a LSTM call can be visualized as below

Diagram

Description automatically generated

Figure 2.18. The detailed LSTM cell

Assume J is the loss of the next layer. We can calculate the derivative as below [18]:

##### Hidden state

##### Output gate

##### Cell state

##### Input gate

##### Forget gate

##### Input

These equations will be calculated t times in each training iteration, where t is the number of time steps in each training iteration. The weights will be updated using the accumulated cost gradient with respect to each weight for all time steps at the end of each training iteration.

To update the weights in this project, I utilize Adam optimizer.

## Conclusion

The theoretical foundation for this project is presented in this chapter, which includes formulas, derivatives, and procedures.

# IMPLEMENTATION AND EVALUATION

## Dataset

### Some information of the data set

The dataset I use in this project is the Coronavirus tweets [19]. The tweets have been pulled from Twitter and manual tagging has been done then. The dataset is split into two files, Corona\_NLP\_test.csv and Corona\_NLP\_train.csv.

The data set is described as below

Table 3.1. The data set information

|  |  |
| --- | --- |
| Training set shape | (41157, 6) |
| Training set memory usage | 1.88 MB |
| Test set shape | (3798, 6) |
| Test set memory | 0.17 MB |

There are 6 columns in the dataset, including coded UserName, ScreenName, Location, TweetAt, OriginalTweet and Sentiment. In this case, I just focus on 2 columns, Original Tweet as input and Sentiment as output.

Graphical user interface

Description automatically generated with low confidence

Figure 3.1. Some records of the training data set

The Sentiment has five types of labels: Extremely Negative, Negative, Neutral, Positive and Extremely Positive. Then I turn them into 3 classes by grouping Extremely Negative with Negative and Positive with Extremely Positive. At the end, the dataset has totally 3 classes, including Negative, Neutral and Positive.

Then I make a class distribution and have the result below:

Table 3.2. The class distribution

|  |  |  |
| --- | --- | --- |
| Class distribution on the whole dataset (train+test) | | |
| Sentiment | Number of tweets | Proportion |
| Negative | 17031 | 38% |
| Neutral | 8332 | 18% |
| Positive | 19592 | 44% |

Figure 3.2. The class distribution on the whole data set

Chart, bar chart

Description automatically generated

Figure 3.3. The class distribution on each data set

### Preprocessing data

* Step 1: Clean emojis from the text
* Step 2: Remove punctuations, links, mentions and \r, \n new line characters
* Step 3: Clean hashtags at the end of the sentence
* Step 4: Filter special characters such as & and $ present in some words
* Step 5: Remove multiple spaces
* Step 6: Check misspelling words
* Step 7: Class balancing using RandomOverSampler

After the step 6, the three classes in training data are imbalanced.

Table 3.3. The numbers of tweet in each sentiment

|  |  |
| --- | --- |
| Sentiment | Number of tweets |
| Negative | 15364 |
| Neutral | 7560 |
| Positive | 17999 |

Therefore, I proceed with oversampling the training data to remove bias towards the majority classes.

After preprocessing, I split the train dataset into dataset for training and validation, with the proportion of 90% and 10% respectively.

Example of a raw tweet: *“Demand for poultry grade maize has been hit owing to Covid-19. However it will spring back in couple of weeks. @amithstar from @AgribazaarA on maize prices crash and MSP procurement.* [*https://t.co/sGBw2QZT0x*](https://t.co/sGBw2QZT0x')*”.*

After preprocessing, the tweet is cleaned as: *“demand for poultry grade maize has been hit owing to covid19 however it will spring back in couple of weeks from on maize prices crash and msp procurement”.*

At the end, we have the dataset as below:

Table 3.4. Number of tweets in each data set

|  |  |
| --- | --- |
|  | Number of tweets |
| Training data | 48597 |
| Validation data | 5400 |
| Testing data | 3787 |

## Implementation

### Input

In the first step, I tokenizer the tweets using Keras tokenizer. After that, Keras embedding layers makes use of word2vec embedding for transforming the tokens into word-vectors.

### LSTM

The next layer is LSTM with units of 128. I applied 0.1 for both dropout and recurrent dropout. The activation for this layer is ReLU, and for kernel initialization I use He uniform.

### Attention

To enhance the output of LSTM before the data is delivered to the next layer, I use a custom layer named attention. The purpose of this layer is to weigh the output to determine which elements have the biggest effects.

### Convolutional layer

After being fed to attention layer, the data is delivered to the convolutional layer. In this layer, I use 128 filters with kernel size 5 and stride 1. To remain the size of the input data, the same padding is applied. Like the LSTM layer, the activation is ReLU and for kernel initialization, I use He uniform.

### Pooling

In the previous chapter, I have mentioned Max pooling and Average Pooling. Although the function of these two types of Pooling is quite different, the results are not too much different. To make it simpler, I choose to use Max Pooling.

### Classification layer

At the end of the network is a dense layer. Because the data have 3 classes, so the dense layer has 3 units. To convert to output into probabilities, I use softmax for the activation function and still use He uniform for kernel initialization.

### Loss and optimizer

I use Categorial Crossentropy for loss function and Adam optimizer with learning rate 0.0001.

Table 3.5. The layer structure of proposed model

|  |
| --- |
| LSTM-CNN |
| LSTM (128 neurons, activation=’relu’) |
| Dropout (0.2) |
| Conv1D (5x5, @64), activation=”relu” |
| Global max pooling |
| Softmax (3-class) |

Table 3.6. The proposed model parameter’s structure

|  |  |  |
| --- | --- | --- |
| Layer (type) | Output Shape | Param# |
| Embedding | (128, 300) | 14603700 |
| LSTM | (128, 128) | 219648 |
| Customed attention | (128, 128) | 256 |
| Conv1D | (128, 64) | 41024 |
| Max pooling | (64) | 0 |
| Dense | (3) | 195 |

## Performance and Evaluation

### Performance evaluation metrics

I employ accuracy (A), precision (P), recall (R), and F1-score (F) as assessment measures to compare and evaluate models. Equations 7 and 8 are used to calculate precision and recall. F1-score, on the other hand, is the harmonic mean of accuracy and recall, as shown in equation.

The ratio of properly classified positive class to the total of correctly and mistakenly categorized positive class values is used to calculate precision P. It informs about the factualness of the model.

The ratio of properly categorized positive class to the total of correctly classified positive class values and mistakenly classified negative class values is used to compute the recall rate. It informs about the completeness of the model.

The F1-score determines the accuracy of the model for each class. When the dataset in unbalanced, the F1-score measure is commonly utilized.

### Evaluation

I compare the proposed model with 4 other model:

* Using CNN only
* Using LSTM only
* Using hybrid model CNN-LSTM. The input is fed to CNN and then LSTM
* Using hybrid model CNN+LSTM. The input is fed to both CNN and LSTM, then I concatenate the two result and deliver to a softmax layer to get the final result.

Table 3.7. The number of parameters of 5 models

|  |  |
| --- | --- |
| Model | Number of trainable parameters |
| CNN | 216,975 |
| LSTM | 412,687 |
| CNN-LSTM | 324,803 |
| CNN + LSTM | 457,531 |
| LSTM-CNN (proposed model) | 261,507 |

Below is the performance of 5 models, with classification metrics, the loss and accuracy of each model.

#### CNN

Chart, line chart

Description automatically generated

Figure 3.4. The CNN loss

The loss reduces with each epoch, as can be shown. However, after the 30th epoch, the validation loss tends to rise.

Shape

Description automatically generated

Figure 3.5. The CNN accuracy

During the training phase, the training accuracy and validation accuracy improve. There is, however, a discrepancy between them, and the validation accuracy is only approximately 83 percent.

CLASSIFICATION METRICS FOR TESTING DATA

precision recall f1-score support

Negative 0.71 0.80 0.75 1629

Neutral 0.67 0.54 0.59 614

Positive 0.75 0.72 0.74 1544

accuracy 0.80 3787

macro avg 0.71 0.68 0.69 3787

weighted avg 0.72 0.72 0.72 3787

#### LSTM

Chart, line chart

Description automatically generated

Figure 3.6. The LSTM loss

Both the loss and the validation loss diminish, and the latter's pace is slower than the others.

Chart, line chart

Description automatically generated

Figure 3.7. The LSTM accuracy

The training and validation accuracy improves with time, with the validation score always exceeding the training score. This is because I added some Dropout layers to the model.

CLASSIFICATION METRICS FOR TESTING DATA

precision recall f1-score support

Negative 0.83 0.80 0.82 1629

Neutral 0.69 0.76 0.72 614

Positive 0.82 0.83 0.83 1544

accuracy 0.80 3787

macro avg 0.78 0.79 0.79 3787

weighted avg 0.81 0.80 0.80 3787

#### CNN-LSTM

Chart, line chart

Description automatically generated

Figure 3.8. The CNN-LSTM loss

The CNN-LSTM model has a loss pattern that is quite similar to the CNN model.

Chart, line chart

Description automatically generated

Figure 3.9. The CNN-LSTM accuracy

The training and validation accuracy both rise at the same rate throughout the first ten epochs. The greater the distance between these two models, however, the more the model gets taught.

CLASSIFICATION METRICS FOR TESTING DATA

precision recall f1-score support

Negative 0.74 0.86 0.80 1629

Neutral 0.70 0.71 0.70 614

Positive 0.85 0.70 0.77 1544

accuracy 0.77 3787

macro avg 0.76 0.76 0.76 3787

weighted avg 0.78 0.77 0.77 3787

#### Proposed model (LSTM-CNN)

Chart, line chart

Description automatically generated

Figure 3.10. The LSTM-CNN loss

In this model, the validation loss is relatively smooth, and the training and validation loss decrease at a faster rate than in previous models.

Chart, line chart

Description automatically generated

Figure 3.11. The LSTM-CNN accuracy

The accuracy of training and validation improves quickly, particularly in the first ten epochs. The scores can reach 95 percent and 91 percent, respectively, by the end of the training time.

CLASSIFICATION METRICS FOR TESTING DATA

precision recall f1-score support

Negative 0.86 0.82 0.84 1629

Neutral 0.75 0.79 0.77 614

Positive 0.84 0.86 0.85 1544

accuracy 0.83 3787

macro avg 0.82 0.82 0.82 3787

weighted avg 0.84 0.83 0.83 3787

#### CNN+LSTM

Chart, line chart

Description automatically generated

Figure 3.12. The CNN+LSTM loss

Although the training loss decreases, the validation loss does not have the same pattern after the 10th epoch, which fluctuates.

Chart, line chart

Description automatically generated

Figure 3.13. The CNN+LSTM accuracy

After the 10th epoch, the validation accuracy has the tendency to remain.

CLASSIFICATION METRICS FOR TESTING DATA

precision recall f1-score support

Negative 0.82 0.78 0.80 1629

Neutral 0.67 0.71 0.69 614

Positive 0.79 0.82 0.81 1544

accuracy 0.78 3787

macro avg 0.76 0.77 0.76 3787

weighted avg 0.78 0.78 0.78 3787

## Comparison of Experimental Results

Chart, line chart

Description automatically generated

Figure 3.14. A comparison of five models' validation loss

It is clearly seen that the LSTM-CNN model (proposed model) has the smallest validation loss, while the CNN model has the highest score.

Chart, line chart

Description automatically generated

Figure 3.15. A comparison of five models' validation accuracy

The LSTM-CNN model has the best performance at the end of the training period, and it has a significant advantage over the CNN model. The variations in performance between the remaining three models are not significant.

Figure 3.16. The number of trainable parameters in 5 models

Figure 3.17. The comparison results between 5 models

The CNN model has the smallest number of trainable parameters, at 216,975 parameters. When compared to the multi-dense layers network, this is the strength of the convolutional neural network's utilization parameter. The LSTM-CNN figure is somewhat higher, at 261,507. The CNN-LSTM model includes 324,803 parameters, whereas the LSTM and CNN+LSTM models each have over 400000.

The suggested model (LSTM CNN) has the greatest accuracy and F score, with 83% for both, as shown in the above figure. With more or less than 78% accuracy and F1 - Score, the CNN-LSTM and CNN+LSTM models produce similar outcomes. When it comes to primitive architecture, the CNN model surpasses the LSTM model with an accuracy of 80%, which is 3% higher than the LSTM, while the F1 - Score reaches 72%, lesser then 5% compared to the LSTM.

## Conclusion

The information about the data set used to test the models is presented in this chapter. The implementation is covered in the second section, which includes information about hyper parameters and functions. The findings and evaluations of the performance of five models are then presented.

# CONCLUSION

Through an experimental comparison analysis, this work evaluates and summarizes predecessors' theoretical and practical experience and presents a text categorization approach based on the LSTM-CNN hybrid model. When compared to simply employing the CNN or LSTM model, the suggested approach produces much better results and reduces the number of parameters in the LSTM model. In comparison to the CNN-LSTM and CNN+LSTM models, the accuracy and F score of the LSTM-CNN model is somewhat enhanced, despite the suggested model having less parameters.

Because the data set utilized in the experiment is somewhat small, and cleaning data is difficult to achieve 100% accuracy, it may differ from reality. Multiple data sets will be gathered for experimentation in the future, bringing the study closer to its intended use. Simultaneously, continue to research the use of deep learning and other technologies in the field of text categorization.

# REFERENCE

*[1] Wan, Mingyu & Fang, Alex & Huang, Chu-Ren. (2019). The Discriminativeness of Internal Syntactic Representations in Automatic Genre Classification. Journal of Quantitative Linguistics. 26. 1-35. 10.1080/09296174.2019.1663655.*

*[2] A. Schmidt and M. Wiegand, ‘‘A survey on hate speech detection using natural language processing,’’ in Proc. 5th Int. Workshop Natural Lang. Process. Social Media, 2017, pp. 1-10.*

*[3] X. Xue-Feng and Z. Guo-Dong, ‘‘A survey on deep learning for natural language processing,’’ Acta Autom. Sinica, vol. 42, no. 10, pp. 1445–1465, 2016.*

*[4] Sundaresh Chandran, “Introduction to Text Representations for Language Processing – Part 2”, Towards Data Science. [Online]. Available:* [*https://towardsdatascience.com/introduction-to-text-representations-for-language-processing-part-2-54fe6907868*](https://towardsdatascience.com/introduction-to-text-representations-for-language-processing-part-2-54fe6907868)*. [Accessed: November 11, 2021].*

*[5] T. Mikolov, K. Chen, G. Corrado, and J. Dean, ‘‘Efficient estimation of word representations in vector space,’’ 2013, arXiv:1301.3781. [Online]. Available: https://arxiv.org/abs/1301.3781.*

*[6] J. Pennington, R. Socher, and C. Manning, ‘‘Glove: Global vectors for word representation,’’ in Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP), 2014, pp. 1532–1543.*

*[7] Vatsal, “Word2Vec Explained”, Towards Data Science. [Online]. Available:* [*https://towardsdatascience.com/word2vec-explained-49c52b4ccb71*](https://towardsdatascience.com/word2vec-explained-49c52b4ccb71)*. [Accessed: November 10, 2021].*

*[8] Christopher Olah, “Understanding LSTM Networks”, Colah’s blog. [Online]. Available:* [*https://colah.github.io/posts/2015-08-Understanding-LSTMs/*](https://colah.github.io/posts/2015-08-Understanding-LSTMs/)*. [Accessed: December 13, 2021].*

*[9] Julian Gilyadov, “Word2Vec Explained”, Hacker’s Blog. [Online]. Available:* [*https://israelg99.github.io/2017-03-23-Word2Vec-Explained/*](https://israelg99.github.io/2017-03-23-Word2Vec-Explained/)*. [Accessed: November 10, 2021].*

*[10] Google, “word2vec”. [Online]. Available:* [*https://code.google.com/archive/p/word2vec/*](https://code.google.com/archive/p/word2vec/)*. [Accessed: November 25, 2021].*

*[11] Nwankpa, Chigozie & Ijomah, Winifred & Gachagan, Anthony & Marshall, Stephen. (2020). Activation Functions: Comparison of trends in Practice and Research for Deep Learning.*

*[12] Kingma, Diederik & Ba, Jimmy. (2014). Adam: A Method for Stochastic Optimization. International Conference on Learning Representations.*

*[13]* *Ioffe, Sergey & Szegedy, Christian. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.*

*[14] Ali Masri, “An Intuitive Explaination to Dropout”, Towards Data Science. [Online]. Available:* [*https://towardsdatascience.com/an-intuitive-explanation-to-dropout-749c7fb5395c*](https://towardsdatascience.com/an-intuitive-explanation-to-dropout-749c7fb5395c)*. [Accessed: January 20, 2022].*

*[15] Bahdanau, D., Cho, K. H., & Bengio, Y. (2015).*Neural machine translation by jointly learning to align and translate*. Paper presented at 3rd International Conference on Learning Representations, ICLR 2015, San Diego, United States.*

*[16] S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), 2017, pp. 1-6, doi: 10.1109/ICEngTechnol.2017.8308186.*

*[17] Sepp Hochreiter, Jürgen Schmidhuber; Long Short-Term Memory. Neural Comput  1997; 9 (8): 1735–1780. doi:*[*https://doi.org/10.1162/neco.1997.9.8.1735*](https://doi.org/10.1162/neco.1997.9.8.1735)*.*

*[18] Christina Kouridi, “Deriving the backpropagation equations for a LSTM”, Christina’s blog. [Online]. Available:* [*https://christinakouridi.blog/2019/06/19/backpropagation-lstm/*](https://christinakouridi.blog/2019/06/19/backpropagation-lstm/)*. [Accessed: December 15, 2021].*

*[19] Aman Miglani, “Coronavirus tweets NLP – Text Classification”, Kaggle. [Online]. Available:* [*https://www.kaggle.com/datatattle/covid-19-nlp-text-classification*](https://www.kaggle.com/datatattle/covid-19-nlp-text-classification)*. [Accessed: December 03, 2021].*