NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATION

BSC THESIS

Suicide risk detection on social media using neural networks

Panagiota P. Ploumidi

Supervisor: **Panagiotis Stamatopoulos,** Assistant Professor NKUA

ATHENS

MAY 2022

ΕΘΝΙΚΟ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Εντοπισμός κινδύνου αυτοκτονίας στα κοινωνικά δίκτυα με χρήση νευρωνικών δικτύων

Παναγιώτα Π. Πλουμίδη

Επιβλέπων: **Παναγιώτης Σταματόπουλος** ,Επίκουρος Καθηγητής ΕΚΠΑ

ΑΘΗΝΑ

ΜΑΙΟΣ 2022

BSC THESIS

Suicide risk detection on social media using neural networks

Panagiota P. Ploumidi
S.N.: 1115201700126

SUPERVISOR: Panagiotis Stamatopoulos, Assistant Professor NKUA

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Εντοπισμός κινδύνου αυτοκτονίας στα κοινωνικά δίκτυα με χρήση νευρωνικών δικτύων

Παναγιώτα Π. Πλουμίδη
Α.Μ.: 1115201700126

ΕΠΙΒΛΕΠΩΝ: **Παναγιώτης Σταματόπουλος**, Επίκουρος Καθηγητής ΕΚΠΑ

ΠΕΡΙΛΗΨΗ

Σύμφωνα με τον Παγκόσμιο Οργανισμό Υγείας, προσεγγιστικά 280 εκατομμύρια άνθρωποι υποφέρουν από κατάθληψη [30] και πάνω από 700.000 αυτοκτονούν [30], ενώ η αυτοκτονία είναι η τέταρτη σε αριθμό θυμάτων αιτία θανάτου εφήβων και νέων ανθρώπων [28]. Παρόλο που η κατάθλιψη επιδέχεται θεραπεία, η πλειοψηφία των νοσούντων αρνείται να αποδεχτεί ότι νοσεί και να ζητήσει ψυχιατρική βοήθεια, ειδικά στις αναπτυσσόμενες χώρες όπου υπάρχει κοινωνικό στίγμα όσον αφορά τις ψυχικές ασθένειες.
Οι πλατφόρμες κοινωνικών δικτύων φιλοξενούν ανθρώπους από όλες τις δημογραφικές ομάδες με διαφορετικά χαρακτηριστικά και έχουν ιδιαίτερη απήχηση στους νέους. Για τους περισσότερους ανθρώπους τα κοινωνικά δίκτυα αποτελούν έναν ασφαλή χώρο όπου μπορούν να εκφράσουν τις σκέψεις και τις ανησυχίες τους, ειδικά όταν τους παρέχεται ανωνυμία. Πλατφόρμες όπως το Facebook και το Instagram έχουν δημιουργήσει επιλογή αναφοράς δημοσίευσης όπου χρήστες μπορούν να αναφέρουν μια δημοσίευση που υπονοεί αυτοκτονικές κινήσεις. Είναι υψίστης σημασίας αυτή η διαδικάσία να αυτοματοποιηθεί ούτως ώστε να μην παραβλέπονται δημοσιεύσεις με τέτοιο περιεχόμενο και επιπλέον να υπάρχει η δυνατότητα πρόβλεψης καταθλιπτικών τάσεων το νωρίτερο δυνατό. Σκοπός αυτής της εργασίας είναι η πρόταση και δημιουργία ενός μοντέλου που θα εκπαιδεύεται σε δημοσιεύσεις της πλατφόρμας κοινωνικών δικτύων Reddit και θα προβλέπει με αξιοπιστία αν ένας χρήστης εμφανίζει σημάδια κατάθλιψης εξετάζοντας της δημοσιεύσεις του. Το προτεινόμενο μοντέλο νευρωνικών δικτύων αποτελεί ένα υβριδικό μοντέλο που συνιστάται από συνελικτικά και αναδρομικά δίκτυα, καθώς και από έναν μηχανισμό προσοχής για τη μεγιστοποίηση της ακρίβειας των προβλέψεων.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Νευρωνικά Δίκτυα

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: τεχνητή νοημοσύνη, νευρωνικά δίκτυα, κοινωνικά δίκτυα, εντοπισμός συναισθήματος, επεξεργασία φυσικής γλώσσας

ABSTRACT

According to World Health Organization records, approximately 280 million people suffer from depression [30] and over 700.000 people die due to suicide [30] while suicide is the fourth leading cause of death among adolescents and young people [28]. Whilst depression is a treatable condition, most people refuse to accept that they are affected and therefore seek psychiatric help, due to social stigma associated with mental disorders, especially in middle-income countries.

Social media platforms host people of all kinds of demographic groups and characteristics and they thrive on young people. For most people social media set a safe space where they can share thoughts and concerns, especially when they are covered by anonymity. Platforms like Facebook and Instagram have created report options for such cases where users can report a post that implies suicidal actions. It is of high importance that this procedure becomes automated, so that no users in need slip our attention and also depressive tendencies can be predicted when it is still early. To resolve this problem, this thesis suggests a NN model that is trained on Reddit users' posts and can reliably predict if a user shows depressive or suicidal signs by examining his posts. The suggested NN is a hybrid model that combines CNN and Bi-LSTM networks and also uses an attention mechanism to optimise predictions.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Artificial Neural Networks

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: artificial intelligence, neural networks, social media, emotion detection, nlp

ACKNOWLEDGMENTS

First of all, I would like to thank my supervisor, Mr.Stamatopoulos for his continuous help and guidance through the development of this thesis and also for his immediate responses.

Furthermore, I wish to thank with all my heart all people close to me that supported me through the four-year journey of my undergraduate studies.

CONTENTS

PREFACE 10

1. INTRODUCTION 11

2. BACKGROUND 13

2.1 Neural Networks 13

2.1.1 Artificial Neuron 13

2.1.2 Learning Process 14

2.1.3 Multi-Layered Perceptrons (MLP) 16

2.1.4 Gradient Descent 17

2.1.5 Backpropagation Algorithm 19

2.1.6 Learning Rate 20

2.2 Convolutional Neural Networks 20

2.3 Recurrent Neural Networks 23

2.3.1 RNN Architecture 23

2.3.2 LSTM Networks 25

2.3.3 Bi-LSTM Networks 26

2.4 Attention Mechanism 27

3. RELATED WORK 29

3.1 Natural Language Processing 29

3.2 Depression and Suicide Risk on Social Media 30

3.3 Other Approaches 30

4. PROPOSED APPROACH 33

4.1 Task Definition 33

4.2 Tools and Technologies 33

4.3 Dataset 33

4.4 Data Pre-Processing 33

4.4.1 Balancing Data 34

4.4.2 Data Cleansing 34

4.5 Embeddings 35

4.6 Model Design . . . 36

4.7 Parameter tuning . . . 36

4.7.1 Learning Rate . . . 38

4.7.2 Epochs . . . 38

4.7.3 Batch Size . . . 39

4.7.4 Activation and loss functions . . . 39

4.7.5 LSTM nodes . . . 39

5. RESULTS . . . **40**

5.1 Model Loss . . . 40

5.2 Model Accuracy . . . 40

5.3 ROC - AUC curve . . . 41

5.4 Comparison to similar models . . . 41

6. ETHICS IN DEPRESSION DETECTION ON SOCIAL MEDIA PLATFORMS . . . **43**

7. CONCLUSION . . . **45**

7.1 Conclusions . . . 45

7.2 Future work . . . 45

ABBREVIATIONS . . . **46**

REFERENCES . . . **47**

LIST OF FIGURES

Figure 1: AI vs ML vs DL [2] 11

Figure 2: Artificial Neuron Model [2] 13

Figure 3: Common Activation Functions [2] 14

Figure 4: Multilayer model architecture [2] 16

Figure 5: Parabola Graphic Representation [41] 17

Figure 6: Large learning rate on the left and small on the right [32] 20

Figure 7: Number of connections with global connectivity (left) and with local connectivity (right) [2] 21

Figure 8: Group of three neurons [27] 22

Figure 9: Parameter sharing [2] 22

Figure 10: Edge detection kernel [2] 23

Figure 11: RNN Architecture [38] 24

Figure 12: LSTM Architecture [40] 25

Figure 13: Bi-LSTM Architecture [25] 27

Figure 14: Application of the Attention Mechanism [9] 28

Figure 15: Model Architecture 36

Figure 16: Overview of the created model 37

Figure 17: Overview of the created model 39

Figure 18: Model Loss 40

Figure 19: Model Accuracy 41

Figure 20: Model ROC - AUC curve 42

Figure 21: Model AUC 42

LIST OF TABLES

Table 1: Parameters used in model architecture and training 37

PREFACE

This thesis is submitted as part of my Bachelor's Degree in Computer Science at the Department of Informatics and Telecommunications of the National and Kapodistrian University of Athens. The study lasted two academic semesters, where the first one was focused on studying neural networks and also on research about similar NLP problems. The second semester was devoted to the development of the necessary code that implements the neural network, the actualization of multiple experiments, the production of the final results and the writing of this thesis.

1. INTRODUCTION

Artificial intelligence (AI) has been in the spotlight of the technological world for the past few years since it has introduced an entirely different operating method than classic computer systems. A perfectly short definition to describe AI could be “Artificial intelligence leverages computers and machines to mimic the problem-solving and decision-making capabilities of the human mind” [42].

Machine Learning (ML) and Deep Learning (DL) are both considered to be sub-fields of AI that are wrongfully used interchangeably, though they have important differences. As ML we define the method according to which a machine is trained on stimuli (input), adapts to the environment in which it is embedded and gains the ability to make decisions based on experience.

On the other hand “Deep Learning is the use of large multi-layer (artificial) neural networks that compute with continuous (real number) representations, a little like the hierarchically organized neurons in human brains” [24]. To be precise, DL is a sub-field of ML.

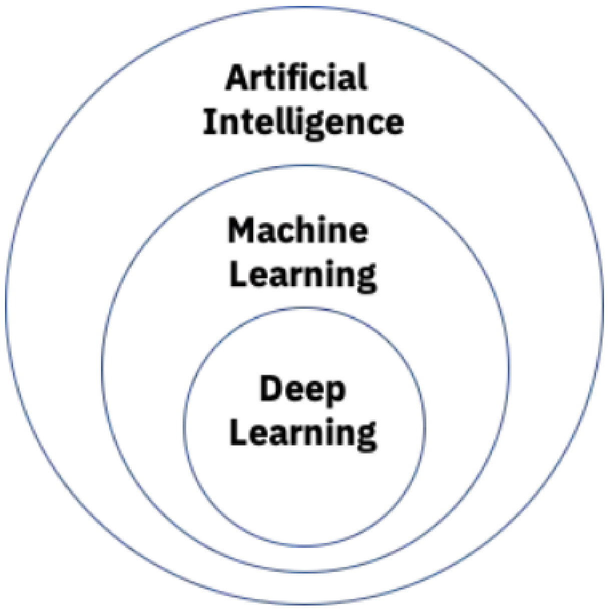


Figure 1: AI vs ML vs DL [2]

AI has been widely applied to resolve numerous problems such as speech recognition, computer vision and recommendation systems. In this thesis we will study the role of AI in a natural language processing (NLP) problem.

Depression is a mental disorder that tortures numerous people and sometimes it can turn out to be physically damaging or even lead to suicide attempts [43]. People, especially adolescents, tend to express themselves more openly on social media, finding it easier to share stressful thoughts and concerns. Since social media are widely used by the majority of people, they offer a potentially great means through which people in need can be offered the help they need. Under this perspective multiple platforms, such as Facebook, have

already taken action in order to confront this problem [37] [3], but almost all the solutions presented until recently wrongfully rely on the fact that the post indicating suicidal ideation will be noticed by another user and will be reported, which may never happen. Of course social media could never replace the role of a mental health specialist, but could serve as a useful tool for the recognition of the problem by the person in need.
The purpose of this thesis is to build a NN model trained on social media posts that will be able to reliably predict if a post implies suicide ideation without the need of human interference. In Chapter 2, the background required for the rest of the thesis is presented in order to make sure that the basic components of neural networks used in the development of this project are well understood.

2. BACKGROUND

2.1 Neural Networks

Neural networks have been for the spotlight in the past few decades and the reason why is because they simulate the function of the human brain which computes in an entirely different way from a digital computer, being able to recognize patterns. It is estimated that the human brain has 100 billion neurons that are connected with 10^{15} links. Similarly to the brain, an artificial neural network (ANN) is composed by artificial neurons and connections between them. If we consider of a neural network as a graph, we can imagine neurons as nodes and connections as edges. Each node represents a processing unit and the connections between them represent the relationship between the processing units. In order to understand better the structure of an ANN it is useful to introduce the elementary component of every ANN, the artificial neuron.

2.1.1 Artificial Neuron

The artificial neuron as a basic unit of every ANN consists of four elements, as shown in the figure below:

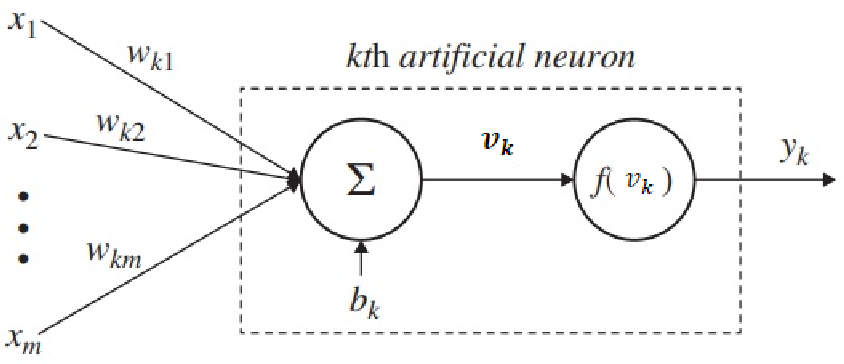


Figure 2: Artificial Neuron Model [2]

1. A set of connecting to the adder links from different inputs x_i, each of them is accompanied by a weight w_{ki}, where k refers to the number of the neuron in question and i refers to the input the weight refers to. The weights' values can be both positive and negative.
2. The bias, b_k, which can be considered as analogous to the role of a constant in a linear function. Its purpose is to shift the activation function, which we will explain later, by adding a constant.
3. An adder that computes the sum of weighted inputs. If we think of bias as $b_k = w_{k0}$ and a default input $x_0 = 1$, then the result of the adder will be $v_k = \sum_{i=0}^{m} x_i w_{ki}$
4. An activation function that decides whether the neuron should be activated or not. This decision is considered as the output y_k of the neuron. To make it easier to understand, it should be explained that the concept of the activation function was inspired by activities that happen inside our brain where different neurons get activated by

different stimulus. For example, if we taste something delicious, certain neurons will be fired in order to help us sense the taste.

The most commonly used activation functions are presented in the table below.

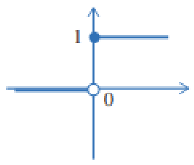
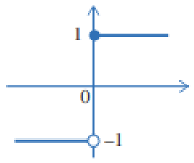
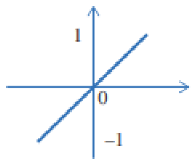
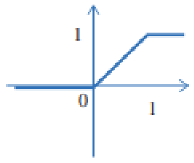
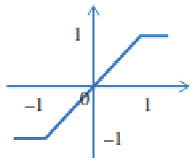
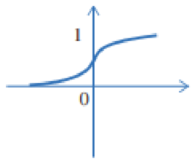
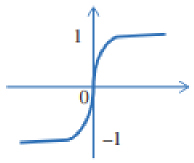
Activation Function	Input–Output Relation	Graph
Hard limit	$y = \begin{cases} 1 & \text{if net} \geq 0 \\ 0 & \text{if net} < 0 \end{cases}$	
Symmetrical hard limit	$y = \begin{cases} 1 & \text{if net} \geq 0 \\ -1 & \text{if net} < 0 \end{cases}$	
Linear	$y = \text{net}$	
Saturating linear	$y = \begin{cases} 1 & \text{if net} > 1 \\ \text{net} & \text{if } 0 \leq \text{net} \leq 1 \\ 0 & \text{if net} < 0 \end{cases}$	
Symmetric saturating linear	$y = \begin{cases} 1 & \text{if net} > 1 \\ \text{net} & \text{if } -1 \leq \text{net} \leq 1 \\ -1 & \text{if net} < -1 \end{cases}$	
Log-sigmoid	$y = \frac{1}{1 + e^{-\text{net}}}$	
Hyperbolic tangent sigmoid	$y = \frac{e^{\text{net}} - e^{-\text{net}}}{e^{\text{net}} + e^{-\text{net}}}$	

Figure 3: Common Activation Functions [2]

2.1.2 Learning Process

The major advantage of the neural networks is that, in contrast to a classical information-processing system, no mathematical model needs to be formulated in order to build the system. An ANN learns by observing the environment in which it is embedded and by processing real life data. Depending on the information it receives, it constantly adjusts its weights in order to achieve a greater knowledge of the environment through time. Thus, the learning process is a task of primary significance. In order to better understand the process, let's consider the neuron of the Figure 1. Neuron *k* has an input vector $X(n)$, where n denotes the time step of the iterative process of adjusting the weights x_{ki}.For each input $X(n)$ the neuron produces $y_k(n)$ as output:

$$y_k = f(\sum_{i=1}^{m} x_i w_{ki}) \tag{1}$$

Similarly, each data sample for ANN training consists of an input vector $X(n)$ and the desired output $d(n)$:

Sample	Inputs	Output
k	$x_{k1}, x_{k2}, ..., x_{km}$	d_k

By comparing the output value $y_k(n)$ to the desired value $d_k(n)$ in each time step, an error $e_k(n)$ is produced where, by definition, is:

$$e_k(n) = d_k(n) - y_k(n) \tag{2}$$

Using this error $e_k(n)$ we can calculate the Cost Function $E(n)$ which is used to quantify how wrong the output of the neural network is. Usually, the MSE (Mean Square Error) function is used which can be expressed in terms of $e_k(n)$ as:

$$E(n) = \frac{1}{2} {e_k}^2(n) \tag{3}$$

This error can be used as a control mechanism during the learning process in order to adjust the weights of the ANN properly. The goal is to change the weights in a way that the $E(n)$ reaches its minimum value. This means that our goal is to achieve minimum difference between the predicted and the desired output values. The learning process based on the minimization of the cost function is known as error-correction learning.
The minimization of the $E(n)$ leads to the *delta rule*, who is being used to define the weight adjustment of the neuron k:

$$\Delta w_{ki}(n) = \alpha \cdot e_k(n) \cdot f'(v_k(n)) \cdot x_i(n) \tag{4}$$

where $w_{ki}(n)$ is the weight of neuron k with input $x_i(n)$ at timestamp n, $f'(v_k(n))$ is the derivative of the activation function applied on the $v_k(n)$, which is equal to $\sum_{i=0}^{m} x_i(n) w_{ki}(n)$, and α is the learning rate that defines the speed of convergence. The learning rate should be carefully selected as it is one of the factors that affect the stability of the learning process of our model. Knowing the result of the delta rule, we can now update the weight value of the neuron according to the following relation

$$x_{ki}(n+1) = w_{ki}(n) + \Delta w_{ki}(n) \tag{5}$$

which means that the new weight value is the old value corrected by $\Delta w_{ki}(n)$. This process continues to repeat itself as long as the *stopping criteria* are not satisfied. Stopping criteria is a parameter or a set of parameters that define the end of the iterative process, such as maximum number of iterations or a threshold on the error of the output.

2.1.3 Multi-Layered Perceptrons (MLP)

Until now, we have only examined models that consist of a single neuron. Nontheless, ANN models usually consist of tens or even thousands of neurons organized in layers. The most popular type of ANNs is the multilayer feedforward networks, usually referred as multilayer perceptrons (MLPs). As shown in the figure below, the typical architecture of an MLP consists of an input layer, multiple hidden layers and an output layer. Moreover, the network is fully connected, which means that each node in the network is connected to all the nodes in the previous layer. In such architectures the input propagates through the network from layer to layer ending up producing an output. In this scenario where there are multiple neurons with multiple input samples, the procedure we follow to adjust the weights during the learning process is going to be a little bit different.

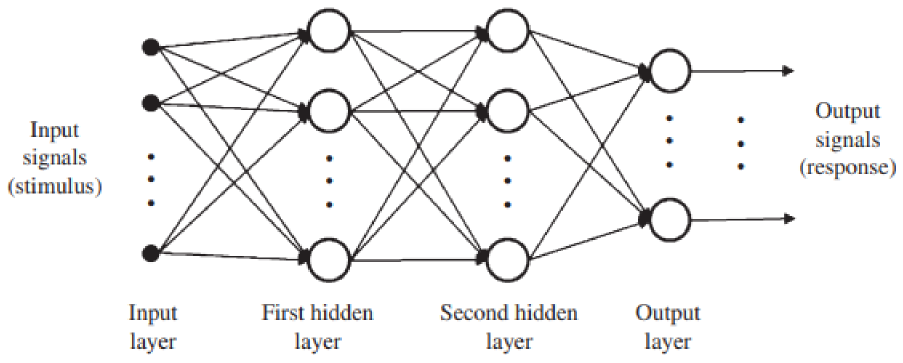


Figure 4: Multilayer model architecture [2]

As mentioned before, the error between the desired output and the actual output of the neuron j of our model is calculated as

$$e_j(n) = d_j(n) - y_j(n) \tag{6}$$

As we did before, we are going to use the MSE function to quantify the error of the network, but this time we should have in mind that we have to do with multiple neurons. Thus, the wanted relation becomes

$$E(n) = \frac{1}{2}\sum_{j\in C} e_j^2(n) \tag{7}$$

where *C* is the set that includes all the neurons of the output layer. We only use the neurons of the last layer because they are the only “visible” ones and the ones for which the error can be calculated directly. To explain this further, the relation above expresses the error of the entire network by adding the error values over all the neurons of the output layer. Now, assuming that there are *N* input samples, the average squared error is the normalized sum of $E(n)$ over all n with respect to size N

$$E_{av} = \frac{1}{N}\sum_{n=1}^{N} E(n) \tag{8}$$

The E_{av} function describes all the free parameters of the model and is called *Cost Function*. Similarly to before, the goal of the learning process is to minimize the Cost Function. This means that the goal is to find these weights for which the predicted values of our model are the closest possible to the desired ones.

2.1.4 Gradient Descent

Gradient descent is a mechanism that leads to the minimization of any function at any dimension. Examining it carefully, we can see that the cost function is of type $Y = X^2$, which means that its graphic representation forms a parabola, as shown in the figure below.

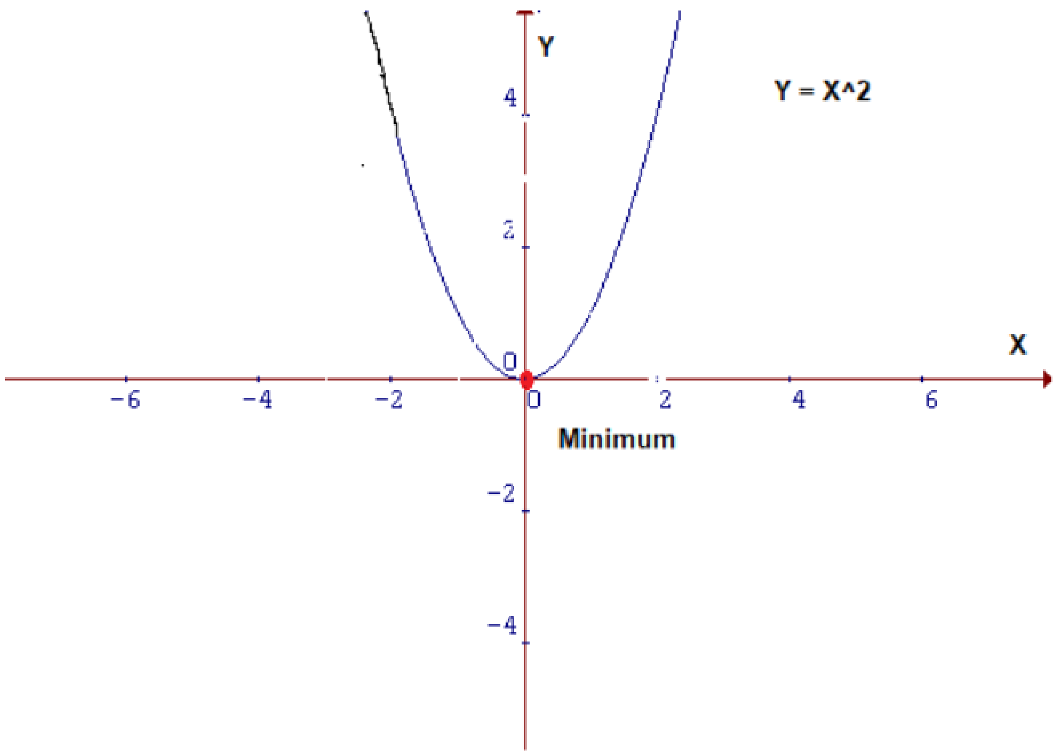


Figure 5: Parabola Graphic Representation [41]

When in two-dimensional space, the minimum of the function is pretty easy to find, but in higher dimensions it may become difficult, especially when local minima and plateaus are involved in the graph of the function. For cases like that, it is necessary to apply the gradient descent method.

Imagine being at a random point of the graph and searching for the minimum which you cannot see. There are two things that need to be decided in order to reach the minimum. First, we need to decide which direction to follow on the graph, upwards or downwards.

Second, we need to decide how fast we want to get there. It could be assumed that the faster we get to the minimum, the faster the learning process will terminate, the better our model is, but if it is examined closely this is not true. If we decide to go fast searching for the minimum, we may surpass it and lose it, so it is crucial to strike a balance between speed and accuracy, but it will be analysed more extensively later.
Now lets focus on the gradient descent. Gradient descent is an algorithm that helps us make these decisions referred above with the use of derivatives. Using derivatives we can calculate the tangent of the graph on a specific point. The direction of the tangent will show us the direction to which we should move. As we approach the minimum point, the tangent becomes less steep and that is an indication that smaller steps are needed to reach the minimum. At this point, it should be reminded that the function whose minimum we want to find is the Cost Function. At each iteration of the learning process (*epoch*) small changes occur on a sample-by-sample basis on the weight values in order to find these values for which the cost function gets its minimum value. Let's denote this weight value change on output neuron *j* with input sample *i* at iteration *n* as $\Delta w_{ji}(n)$. Since the weights change for every input sample, we may use the $E(n)$ function, which is a function of weights *w*.

$$E(n) = \frac{1}{2}\sum_{j \in C} e_j^2(n) \tag{9}$$

The change $\Delta w_{ji}(n)$ is proportional to the partial derivative $\frac{\partial E(n)}{\partial w_{ji}(n)}$. Applying the chain rule, we get

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \cdot \frac{\partial e_j(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)} \cdot \frac{\partial v_j(n)}{\partial w_{ji}(n)} = -e_j(n) \cdot f'(v_j(n)) \cdot x_i(n) \tag{10}$$

where, as mentioned earlier in this chapter, $v_j(n) = \sum_{i=1}^{m} w_{ji}(n)x_i(n)$ and $y_j(n) = f(v_j(n))$. The minus sign in the equation above is used to indicate the direction towards which we should move to update the weights so as to reduce the value $E(n)$. By the delta rule, the change applied to the weight $w_{ji}(n)$ is

$$\Delta w_{ji}(n) = \alpha \cdot e_j(n) \cdot f'(v_j(n)) \cdot x_i(n) \tag{11}$$

where again α is the learning rate. If we assume that $\delta_j(n) = e_j(n) \cdot f'(v_j(n))$, then

$$\Delta w_{ji}(n) = \alpha \cdot \delta_j(n) \cdot x_i(n) \tag{12}$$

This value $\delta_j(n)$ is known as the local gradient. Finally, the new weight value is

$$w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji}(n) \tag{13}$$

The minimum is found when convergence is achieved.
There are three types of gradient descent that differ in the amount of data they use. Previously, we examined the *Stochastic Gradient Descent (SGD)* where the weights are being updated after each input sample, which leads to computational expensiveness. By con-

trast, there is the *Batch Gradient Descent* that calculates the error for each sample but updates the weights after all samples have been processed. The advantages of the Bach Gradient Descent are its computational efficiency and the stability it provides to the error of the gradient descent and its convergence. The third method is the *Mini-Batch Gradient Descend* which is a combination of the other two methods. This method splits the dataset into small batches and updates the weights for each one of them. The Mini-Batch Gradient Descent is the most commonly used method among the three of them.

2.1.5 Backpropagation Algorithm

Now that we are familiar with the concept of gradient descent, let's return to the MLPs in order to better understand the connection between these two. The MLPs have been used extensively to solve a number of problems in a supervised manner using the popular *backpropagation algorithm*.

The backpropagation algorithm consist of two phases, a forward pass and a backward pass. During the forward pass, all the weights of the network are fixed and initialized in random values. Then, an input sample is applied to the input nodes of the network and then it propagates through the network layer by layer. Finally, an output is produced at the output nodes of the model.

On the other hand, during the backward pass the weight values are being updated and adjusted according to the delta rule in order to minimize the error of the output for the given input sample. The delta rule applied during the backward pass is the same rule as the one referred in gradient descent method. The problem that occurs is that when applying the gradient descent method previously, only the output nodes were taken into consideration, but now we need to adjust all the weight values of the network, including the ones in the hidden layers that are not directly accessible as they too affect the output of the model. To resolve this problem we will consider of two different ways of computing the $\Delta w_{ji}(n)$, depending on the position of the neuron *j* inside the network.

If the neuron *j* is located in the output layer of the network, then all previous relations presented above are valid without any alteration. But if the neuron *j* belongs to a hidden layer, then the sum of all the local gradients of the next layer must be calculated and included in the calculation of the local gradient value for neuron *j* as the product of the derivative of the activation function $f'(v_j(n))$ of neuron *j* and the weighted sum of all local gradients of the neurons of the next layer that are connected to neuron *j*. Thus, the redefined local gradient becomes

$$\delta_j(n) = f'(v_j(n)) \cdot \sum_k \delta_k(n) w_{ki}(n), k \in D \tag{14}$$

where *D* is the set of neurons of the next layer (hidden or output), which are connected to the neuron *j*.

Since we go backwards, when it is time to calculate the local derivative of neuron *j*, the local derivatives of all the neurons *k* of the next layer have already been calculated. In this manner, all the weights are updated recursively. An iteration of the algorithm is completed when all weights of the network have been updated once for each input sample. Usually, it is necessary that many iterations should be applied with the same set of input samples in order to sufficiently minimize the cost function and stop the algorithm.

2.1.6 Learning Rate

As mentioned before, the learning rate is a hyper-parameter whose choice is of great importance. If we choose to proceed with large steps, we may achieve a faster convergence but there is a high risk of overshooting the minima as it bounces back and forth. On the opposite, if the learning rate is small enough, more time will be needed to achieve convergence. This problem is graphically represented in the figure below. Usually a fixed learning rate is used in gradient descent that varies from 10^{-6} to 1. Otherwise, a large learning rate can be chosen at the beginning of the learning process and then gradually reduce it linearly during training.

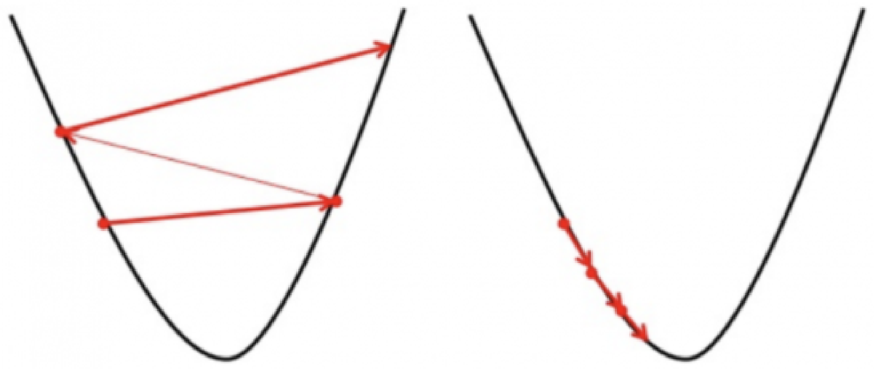


Figure 6: Large learning rate on the left and small on the right [32]

A simple method to avoid the risks of the learning rate is the insertion of a *momentum* term in the delta rule

$$\Delta w_{ji}(n) = \alpha \cdot \delta_j(n) \cdot x_i(n) + a \cdot \Delta w_{ji}(n-1) \tag{15}$$

where a is a constant, usually in range [0,1], and $\Delta w_{ji}(n-1)$ is the adjustment of the specific weight for a previous input sample. Adding a momentum guarantees that there is no big deviation between consecutive adjustments on the same weight value and smooths the weight updating process. Further, it also offers the advantage of preventing the learning process from terminating in a local minimum rather than the minimum of the cost function.

2.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) were first introduced by LeCun in 1989 and have played a very important role in the history of neural networks, having given solution to a big number of problems such as image classification and feature extraction. They are specialized in processing data in the form of multidimensional arrays, 1D for signals, 2D for images and 3D for videos and volumetric data, like 3D medical scans. There are four elements that differentiate CNNs from traditional ANNs and make them successful and

efficient: (1) local connections, (2) shared weights,(3) pooling and (4) the use of many layers.In the pages that follow these four elements will be further explained so that the concept of CNNs can be better understood.
In traditional ANNs, if we want to analyse a series of data to make some predictions, the simplest way to do so is to connect all data to a fully connected layer. This means that there is a bunch of different neurons and every input connects to every neuron. On the other hand, CNNs offer a more sophisticated approach that is based on local connectivity and sparse connections. For example, in the case of a 1000x1000 pixels image, in a traditional ANN of two layers all these pixels would be connected to all the neurons of the next layer producing $10^6 \cdot 10^6 = 10^{12}$ connections, given that the two layers have the same number of nodes. In the case of a CNN, only local segments of the image would be connected with the next layer producing this way a much smaller total number of connections in the network. Specifically, in the figure below, the reduction of the number of connections is demonstrated when local connectivity is used. In the case of 5 input nodes and 5 neurons, global connectivity produces 25 connections when local connectivity produces only 13.

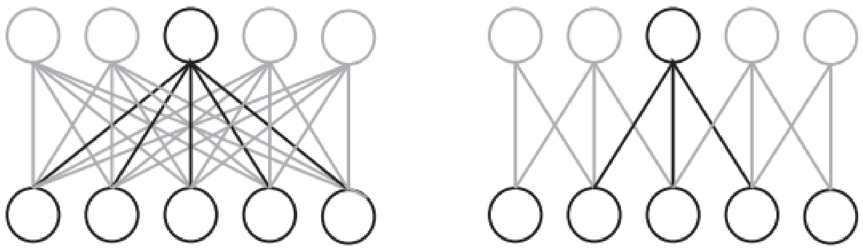


Figure 7: Number of connections with global connectivity (left) and with local connectivity (right) [2]

Another very important characteristic of CNNs that makes them so efficient is the parameter sharing. In traditional ANNs each neuron was assigned a weight value, meaning that there were as many parameters as the number of neurons in the network. What is different with CNNs is that, instead of neurons, we focus on groups of parallel neurons that all get the same input and compute different features. For example, in a group of three neurons (Figure 7), one neuron may detect horizontal edges, another vertical edges and another might detect the contrast between colors.

Inside a group, each neuron is assigned a weight and identical groups have identical weight values on their neurons. As a consequence, a CNN using identical groups of three neurons may have thousands of neurons but only three weight parameters. Thus, the number of parameters inside the network is significantly reduced. For example, in the figure below we can see that instead of 13 parameters with global connectivity, only 3 are necessary when applying local connectivity.

Local connectivity and parameter sharing are the basis of *convolution kernels*. Convolu-

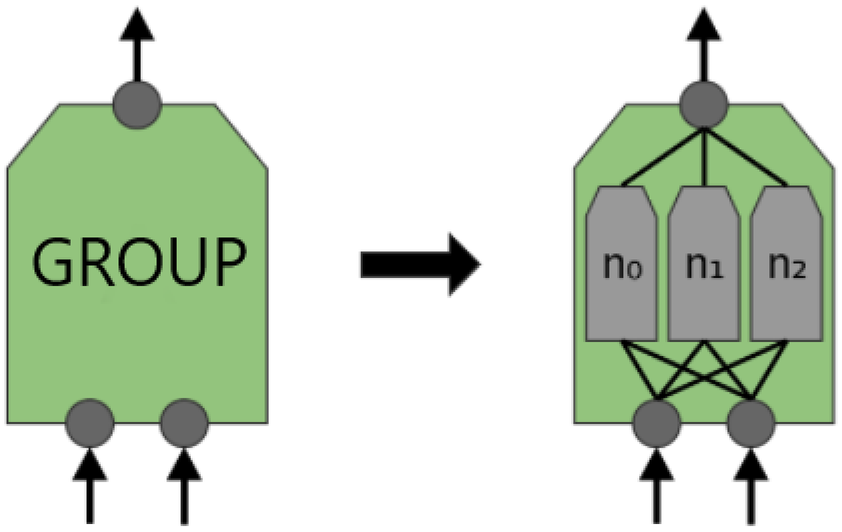


Figure 8: Group of three neurons [27]

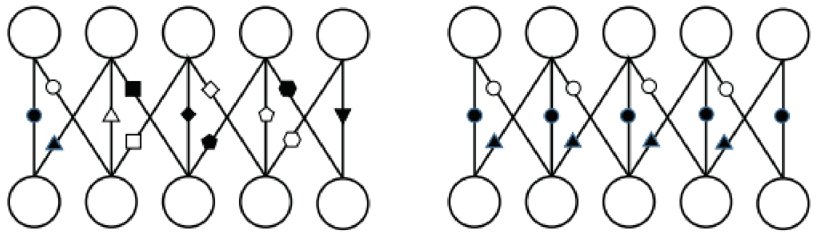


Figure 9: Parameter sharing [2]

tion kernels are small local matrices that are slid across the input matrix, for example an image, and multiplied with the input in order to extract specific features. There are kernels created to detect horizontal and vertical lines, corners and shapes as higher-level features of the image.

$$\begin{vmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{vmatrix}$$

Figure 10: Edge detection kernel [2]

Furthermore, CNNs often use interweaved pooling layers so as to additionally modify the output of the previous convolution layer. The advantage of pooling layers is that, when applied, the model is focused on the presence of a feature rather than its exact position. For example, when trying to detect a face on an image we do not care about the exact pixel-perfect accuracy position of the eyes as long as we know that there is one eye on the left side of the face and one on the right side. The most popular pooling procedure is called *max-pooling*. In max pooling, the pooling layer outputs the maximum activation function in a specified input region so we know that a feature was present in a region of the previous layer, but not its exact position.
Finally, the last advantage of CNNs is their ability to perform well when multiple convolution layers are used. CNNs are composable, meaning that the output of one layer can become the input of the next one. With each layer the model can detect higher-level and more abstract features. All these abilities of CNNs are what made them so successful in image analysis related problems, recognition problems and natural language processing.

2.3 Recurrent Neural Networks

2.3.1 RNN Architecture

Recurrent neural networks (RNNs) is the topmost method to process sequential data due to their ability to allow information to persist. In our every day life we are called to make decisions based on previous sequential information. While watching a movie we can predict its ending because our understanding of the plot depends on the part of the movie we have seen so far. When a new scene begins we do not forget everything we have seen before and keep watching the movie as this scene is the first one. This is something traditional neural networks cannot do, but RNNs can because of the loops that allow them to have memory of things of the past. This is the main reason why RNNs have been effectively applied in many problems such as speech recognition, sentiment classification and machine translation.
Apart from their ability to remember, what differentiates RNNs from MLPs is the fact that RNNs share parameters. To better understand this, let's examine the architecture of an RNN.

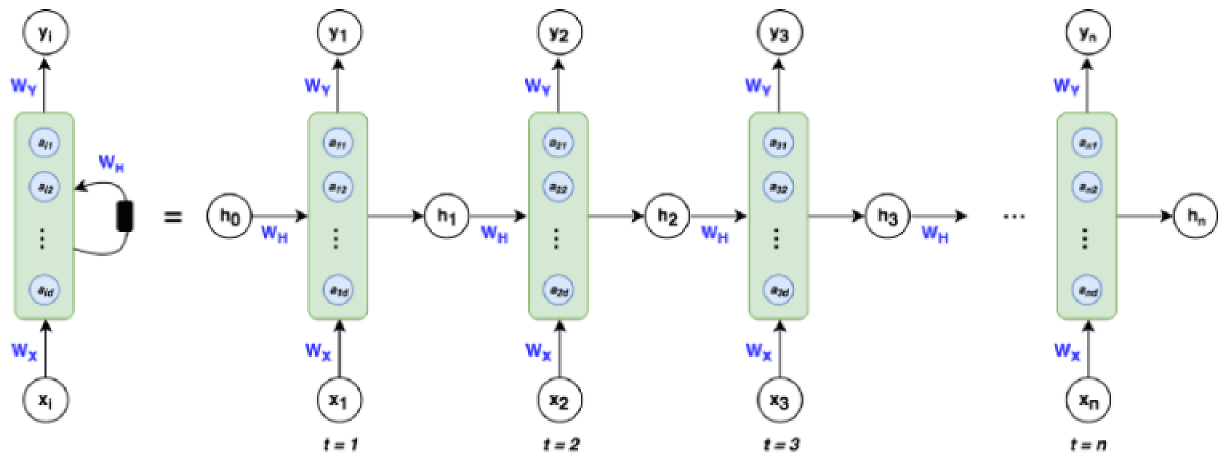


Figure 11: RNN Architecture [38]

In the figure above (figure 10), on the left side we see the shorthand notation of an RNN and on the right side we see its expanded version. The green blocks in the picture are called *hidden states* and the blue circles inside are called *hidden units*. The number of the hidden units inside a state is defined by a parameter *d*. Imagine that each green block acts as an activation function of the blue circles. x_i is the input vector to the node in the form of numerical values on timestep *i* and y_i is the prediction of the hidden state on the current timestep. h_i is the notation for the output of each hidden state after the activation function has been applied on the hidden units. What happens inside the RNN is that at each timestep *i* the output of the previous timestep $i-1$ becomes its input creating a *feedback loop*. In this way the model takes into consideration information from the past to compute the output of the present iteration. Matrices W_x, W_y, W_h are the weights of the RNN which, as mentioned before, are shared. This means that the weights remain the same during all timesteps.

Now let's focus on the equations applied during an RNN calculation. There are three important equations that explain the whole process:

$$a_t = W_H h_{i-1} + W_x X_t \tag{16}$$

$$h_t = tanh(a_t) \tag{17}$$

$$y_t = softmax(W_Y h_t) \tag{18}$$

First of all, in order to compute the hidden units, we need the weighted output of the previous hidden state h_{t-1} added to the weighted input x_t. As we explained above, the hidden state acts as an activation function on the hidden units, thus the output of the hidden state is the result of *tanh* applied on a_t. The prediction made is the result of the softmax function applied on the weighted output of the state.

The problem that occurs when using RNNs is the *vanishing gradient*. But, what exactly

causes that problem? For example, if we were given the sentence “the cute dog, which was playing with the cat, was a golden retriever” and had to predict the words “golden” and “retriever”, we would have to take into account the words “cute” and “dog” that are words which describe the golden retriever. The problem is that the words “cute” and “dog” are far from the words “golden” and “retriever” inside the sentence.
Similarly to MLPs, RNNs use backpropagation to update their weights by applying the chain rule. The longer the sentence is, the longer the chain rule becomes, since we need to compute the derivatives from the words we want to predict all the way back to the start of the sentence recurrently. As the chain rule becomes longer, it approaches zero (vanishing) since it is formed by the product of the derivatives of the activation function *tanh*, which maps input values between -1 and 1, causing the magnitude of these derivatives to be less than 1. Thus, the longer the chain rule, the less impact the first words of the sentence have on the prediction of the last words. This means that, though in theory RNNs can learn long term dependencies if the are given the right parameters, in practice they are not able to do so. At this point, LSTMs come to the foreground to resolve this problem.

2.3.2 LSTM Networks

LSTM is a type of RNNs that overcomes the problem of the vanishing gradient. Learning long term dependencies is not something LSTMs struggle to do. Actually, it is their default behavior. LSTMs have a similar architecture to the typical RNNs, but instead of simple RNNs, LSTMs are composed out of four neural network layers that interact in order to manage to remember past information (Figure 11).

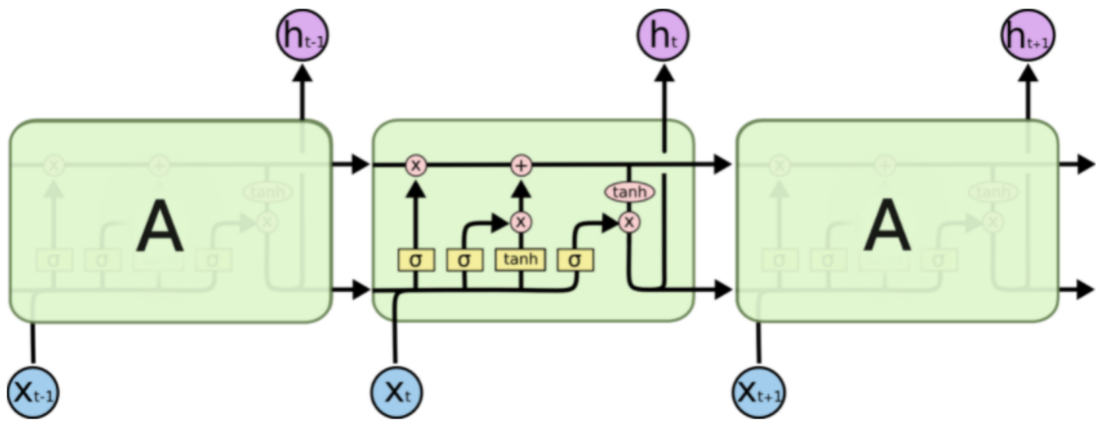


Figure 12: LSTM Architecture [40]

As we see in the figure above (figure 11), the LSTM consists of two very important elements, the *cell state* and the *gates*. The cell state is the horizontal line that runs through the top of the diagram. The information flows along the cell state and the gates intervene to alter it, by removing and adding information. The gates, depicted as the yellow rectangles in the diagram, are structures that regulate the information going in and out of the cell state. They are composed out of a sigmoid neural network and a pointwise multiplication.

The output of the sigmoid NN is a number between 0 and 1 that defines the amount of information that will go in or out of the cell state.
There are four gates and each one of them plays a very important role in altering the information that flows along the cell state. First of all, the *forget gate* is responsible for deciding which information is to be removed from the cell state. It takes the output of the previous timestep h_{t-1} and the current input x_t as input and outputs a vector of numbers between 0 and 1 that has the size of the cell state of the previous timestep C_{t-1}. In the equation below, W_f is the weight matrix, σ symbolizes the sigmoid activation function and b_f is the bias.

$$f_i = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{19}$$

After removing from the cell state what needs to be forgotten, it is time to decide the information we want to add to the cell state. Similar to the procedure followed to forget something, the *input gate* first selects the information to be added and its amount:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{20}$$

Then, a vector of new candidate values C'_t is created by applying a *tanh* layer:

$$C'_t = tanh(W_c[h_{t-1}, x_t] + b_c) \tag{21}$$

Then, having decided and calculated the information that will go in and out of the cell, it is time to update the cell state by multiplying the old cell state by the result of the forget gate and the candidate values vector by the result of the input gate:

$$C_t = f_t * C_{t-1} + i_t * C'_t \tag{22}$$

Finally, we need to decide the output, which will be a filtered version of the cell state:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{23}$$

$$h_t = o_t * tanh(C_t) \tag{24}$$

As before, the o_t plays the role of the filter and the *tanh* is applied to push the values between -1 and 1.

2.3.3 Bi-LSTM Networks

As explained above, LSTM networks have the ability to keep information from the past in order to make predictions based on the context. The constraint that exists is that it is sometimes difficult to make predictions based exclusively on past information. To further explain this, think of the following example:

“Numerous stars exist in the universe.”
“Numerous stars where photographed on the red carpet for the Oscars ceremony.”

In this case, it is difficult for a NN to understand the meaning of the word “stars” using only information from the past since the meaning is formed according to the context that follows. This problem can be resolved with the application of bi-LSTM neural networks. Bi-LSTM is a NN that operates as a combination of two LSTMs, one processing the input from the beginning to the end and the other processing the input in reverse order (Image 12). Thus, bi-LSTMs have the ability to preserve information from both the past and the future at any time. In this way, each word is better understood as its meaning is a result of past and future information and the NN is aware of the full context of the sentence before making any predictions.

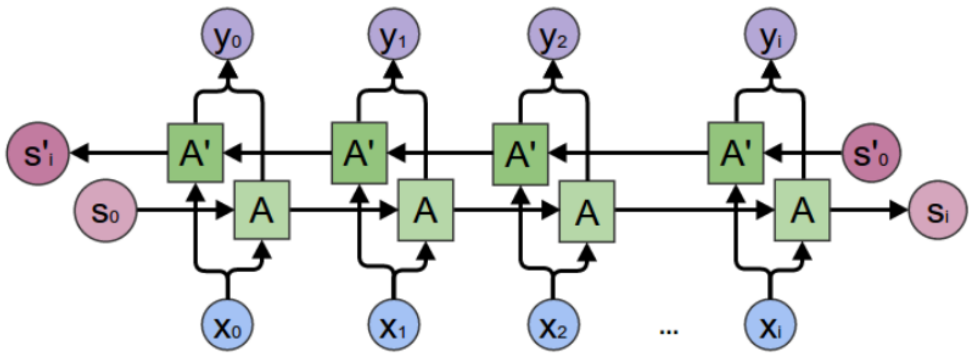


Figure 13: Bi-LSTM Architecture [25]

Due to their ability to have full supervision on information from both the past and the future, bi-LSTMs have been of great importance to the solution of various Natural Language Processing (NLP) problems achieving better results than traditional LSTMs.

2.4 Attention Mechanism

When describing an object, our brains focus on different aspects of the object. Firstly we describe the shape, then the material, the color, the usage etc. Each time we pay attention on one property of the object ignoring all the others. To simulate this function of the human brain, Neural Networks use an *Attention Mechanism*. The attention mechanism helps the NN to focus only on the part of the input that is relevant to the word to be predicted. Hence, the NN ignores irrelevant details and makes predictions according to information that is directly related to the word that is going to be predicted. For example, in figure 13 we can see the connections between the words and the context used for translation. In this case, the translation of the word *the* needs the words *European*, *Economic* and *Area* to determine the gender of the french article but nothing more than that, so the focus is only on these four words.

Bahdanau et al [5] introduced the concept of attention in the context of an encoder-decoder model. In the traditional model, the encoder reads the input as a sequence of vectors $x = (x_1, ..., x_{T_x})$ and transforms it into a vector c. This transformation is usually an RNN

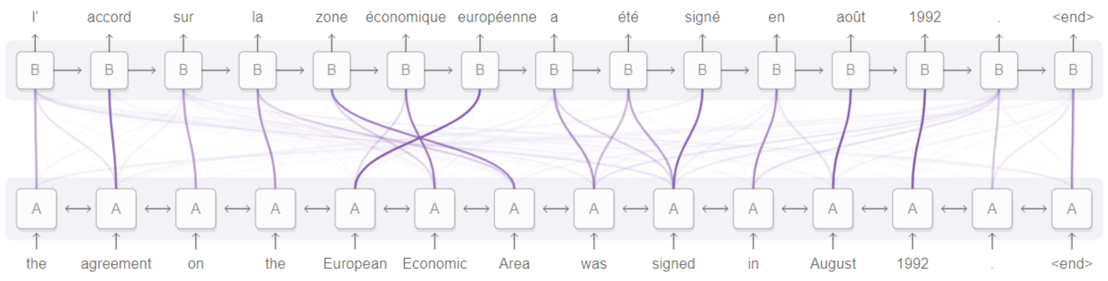


Figure 14: Application of the Attention Mechanism [9]

with a hidden state $h_t = f(x_t, h_{t-1})$ and a vector generated from the sequence of the hidden states $c = q(h_1, ..., h_{T_x})$, where f and q are non-linear functions. The decoder is trained to predict the next word $y_{t'}$ given the context c and all the previously predicted words $y_1, ...y_{t'-1}$. The difference Bahdanau introduced is that now the decoder will be given a different context for each word to be predicted and not a general one as in the traditional model. The new context vector c_i depends on a sequence of annotations $(h1, ..., h_{T_x})$ that occur as result of the encoder. Each annotation contains information about the whole sentence but has a strong focus on words surrounding the word to be predicted. This means that the words on which the most attention is payed are the words that will affect the prediction the most. When trying to predict the i-th word y_i of a sentence, the context vector c_i is computed as the weighted sum of the annotations h_i:

$$c_i = \sum_{j=1}^{T_x} a_{ij} h_j \tag{25}$$

In the equation above, a_{ij} is the weight of each annotation computed as:

$$a_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})} \tag{26}$$

where

$$e_{ij} = a(s_{i-1}, h_j) \tag{27}$$

is an alignment model that scores how well the inputs around position j match the output at position i. Here, s_{i-1} symbolizes the RNN hidden state.

The attention mechanism acts as a feedforward NN that can be trained along all the other components of our model. Moreover, when applying the attention mechanism, we relieve the encoder from the burden to fit all the information of the input into a fixed-size vector. With this approach, the information is distributed along the annotations and can be retrieved from the decoder selectively.

3. RELATED WORK

3.1 Natural Language Processing

Natural Language Processing (NLP) describes a problem category concerning the way the machine understands and processes the human speech. Every day, we are in touch with numerous applications of NLP problems without even acknowledging it, with the most famous example being Siri assistant. Though we gave a definition of the NLP problem, it is quite generalized and remains vague. To better understand the concept of the NLP, we are going to analyse some specific tasks a computer is called to perform that are directly related to the NLP and we are familiar with.

Speech recognition is the task of reliably understanding human speech and also converting it into text. This may sound an easy task for a machine to perform, writing down what it hears, but it is actually pretty challenging to achieve since people usually talk more freely than they write, skipping syllables, concatenating words, using local accents and ignoring grammar rules. Moreover, the machine is called to understand first the sayings of the speaker in order to capture the essence of the sentence properly before converting it to text. Speech recognition is applied very successfully in *spam detection*, which is the process of classifying emails to spam and non-spam. This is achieved by scanning each email for bad grammar, misspelled words and threatening content. Spam detection is a problem that is considered to be mostly solved. Speech recognition is also used in every application that includes voice commands, like chatboxes, Apple’s Siri and Amazon’s Alexa.

Apart from speech recognition, virtual assistants like Siri and Alexa use another application of NLP problems, *natural language generation*, in order to answer to the commands or questions of the user. Natural language generation is the opposite procedure of speech recognition, where the machine is called to recreate human speech in order to communicate structured information. Both these problems referred above are applied in machine translation too,a technology we all have used at least once.

Another interesting NLP problem is *sentiment analysis* where there is an attempt to extract the emotions (positive, negative, neutral) of a person towards something from their voice or from text. An application of this problem is *social media sentiment analysis* where users’ posts can be analysed in order to extract people’s emotions about everything: products, ideas, news etc. In such manner, companies can detect the feelings of consumers towards their products and generally the mass’ sentiments towards a subject can be monitored.

Similar to sentiment analysis, but with distinguishing differences, is *emotion detection*. The two problems might seem identical but they are not. The main difference is that sentiment analysis is focused on the classification of the input into positive, negative or neutral feelings, while the goal of emotion detection is to identify distinct human emotions like happiness, anger, anxiety, depression. It sounds fascinating that a computer might have the ability to understand our emotional state at the same time some people are lacking this, but the applications of this technology are even more interesting. In this thesis there will be an attempt to use emotion detection in order to classify social media users to those who show suicidal ideation and to those who do not. An application of science like this could be beneficial to the society since people with mental disorders could get the help they need at an early stage.

3.2 Depression and Suicide Risk on Social Media

According to World Health Organization records, approximately 280 million people suffer from depression [30]. When depression appears as a chronic disease it can change the affected person's life, completely lowering their standard of living, leading people to self-harm and, in worst cases, to suicide. It is estimated that over 700.000 people die due to suicide [30] while suicide is the forth leading cause of death among adolescents and young people [28]. Financial instability, fast pace of life and the recent pandemic outburst only seem to increase the percentages of depressive people.

Whilst depression is a treatable condition, most people (more than 75%) refuse to accept that they are affected and seek psychiatric help, due to social stigma associated with mental disorders, especially in middle-income countries. Social media platforms host people of all kinds of demographic groups and characteristics and they thrive on young people. For most people social media set a safe space where they can share thoughts and concerns, especially when they are under the cover of anonymity. On social media and behind anonymity, users can speak their minds and express their emotions unfiltered.

Platforms like Facebook and Instagram have created report options for such cases, where users can report a post that implies suicidal actions [37]. However, nobody can guarantee that this post will come to other users' attention or that the latter will notice that the poster is in danger. This reports mechanism is suitable for cases of very advanced depression where the subjects harm themselves or express suicidal thoughts. To successfully face depressive episodes it is necessary to be offered help at an early stage. For these reasons, it is of high importance that this procedure becomes automated, so that no users in need slip our attention, and any depressive tendencies of a user to be predicted when it is still early. To resolve this problem, this thesis suggests a NN model that is trained on Reddit users' posts and can reliably predict if a user shows depressive or suicidal signs by examining his/her posts.

3.3 Other Approaches

Many models and methods have been proposed in order to resolve this or similar problems. Starting with a simple machine learning application, Mandar Deshpande and Vignesh Rao [11] chose to apply emotional artificial intelligence in order to recognize depression signs in textual tweets. Building a binary classifier, they tested the performance of classic machine learning methods (SVM and Multinomial Naïve Bayes) in an NLP problem. The results showed that Naïve Bayes outperformed SVM with 83% accuracy versus 79% on SVM.

Apart from classic machine learning methods, neural networks were widely applied in NLP problems like this in multiple combinations of layers. With the application of the simplest neural network model, using contextualized word embeddings (pre-trained ELMo) and a backpropagation algorithm, Yaakov Ophir et al [20] created two different models of fully connected layers; a single task model (STM) to predict suicide risk from Facebook posts directly and a multi task model (MTM) that took under consideration theory-driven risk factors for suicide and participants' psychiatric and personal data.
Taking it one step further, in the paper of *Jingcheng Du et al* [13], the authors not only detected suicidal tweets using a CNN to classify them, but they also tried to extract the psychiatric stressors that leaded to these tweets in the first place. In other words, they attempted to discover the factors that lead people to suicidal thoughts. In order to achieve this, they faced the problem as a named entity recognition (NER) task and created an RNN (bi-LSTM with transfer learning) to resolve it.
Except for the combinations of different NN layers, more complicated methods were applied in an attempt to optimize the models' predictions. For example, in the study of *Ahmed Husseini Orabi et al* [15] an optimized-word-embedding model was produced using multi-task deep learning (MTL) and compared to the commonly used word-embedding models on different types of ANN architectures (CNN and RNN models). The word-embedding model that was implemented for the purposes of this study was trained on theCLPPsych2015 dataset and was optimized so as to learn a feature representation of health-specific tasks. As a result, CNN models performed better than the RNN ones and it was also noticed that the models that used the optimized word embeddings maintained performance with generalization ability.
So far, we have only mentioned papers whose the object of study was textual, but in social media people communicate with visual and audiovisual means too, like images and videos. In the last few years many studies that tried to detect depression signs from visual context were published and *SenseMood* [18] was one of them. SenseMood is a system whose purpose is to detect and analyze the behavior of potentially depressive users on Twitter. What is different with this approach of depression detection is that the input used to train the model is a combination of textual and visual data, thus this study is about a multimodal deep learning model. Concerning the visual data, a binary CNN-based classifier was implemented and for the classification of textual data pre-trained BERT was used. The system is trained offline and is used for online detection and analysis. Apart from the classification of users to depressive and non-depressive, SenseMood also provides a depression analysis report for each user.
Since we mentioned audiovisual content as a means of communication on social media,the paper of *Anshu Malhotra and Rajni Jindal* [19] comes to expand previous studies on the subject by examining not only text and images, but users' videos in real time too. State-of-the-art neural networks were used in this project, such as VGG-16 for image processing and feature extraction and Faster-RCNN for person detection in videos. Fur-

thermore, to exploit all information we can get from a user's post, even emoticons where taken under consideration as a frequently used means of emotional expression.
Due to the variety of posts depressed people may share on social media, only a small amount of them indicates depression, making the selection of relevant indicator texts and images even more difficult. In order to overcome this obstacle, the study of *Tao Gui et al* [16] introduced a multi-agent reinforcement learning method that selects indicator texts and images which then are given as an input in a binary classifier which predicts if the user is depressed. This method showed better results than other multimodal methods and also obtained a strong and stable performance in realistic scenarios.
Since depression detection from social media textual posts is a text classification problem, it is quite similar to the subject of the study of *Beakcheol Jang et al* [17]. In order to classify text into different categories and predict emotions, Beakcheol Jang et al proposed an attention based Bi-LSTM + CNN hybrid model that exploits both the advantages of LSTM networks (i.e. it can capture long-term dependencies between word sequences) for sentiment classification and CNNs (i.e it extracts low-level features to reduce the number of dimensions very fast). Moreover, the attention mechanism (Bahdanau attention [5]) can increase the prediction accuracy and decrease the number of learnable weights that are necessary for the prediction. As a result, the proposed model achieved a better performance than other popular models used in text classification, such as CNNs, LSTMs and MLPs.
Nevertheless, in their study *Chaitanya Bhargava at al* [22] applied methods we previously saw in sentiment analysis and text classification to detect signs of depression in twitter users. More specifically, they created a hybrid model that combines CNNs and LSTMs in order to detect depression in Twitter. The results showed that the combination of CNNs and LSTMs performs slightly better than LSTMs with 91.35% accuracy score. Similar to the work of Chaitanya et al is the study of *Michael Mesfin Tadesse et al* [21] that detects suicide ideation in Reddit users' posts. The main difference is that, in contrast to previously mentioned studies, Michael Mesfin Tadesse et al reversed the order of CNN and LSTM layers achieving equally good results in prediction accuracy since it takes advantage of the LSTM's ability to maintain context information resolving in this way the vanishing gradient problem and also uses a CNN layer for the extraction of the local pattern.

4. PROPOSED APPROACH

4.1 Task Definition

The purpose of this thesis is to implement a model capable of classifying social media posts and distinguish them according to their content to suicidal and non-suicidal. To better explain this, the goal is to design a model which will be trained on textual social media posts, specifically the ones from Reddit, so that it can detect patterns and extract the meaning of the sentences. Finally, after being trained, the model should be able to reliably predict if a user's post indicates suicidal thoughts and behavior.

4.2 Tools and Technologies

The code development was implemented in Python 3.6 and is structured in the form of a notebook (Google Colab and Kaggle) so that it is easier to read. Notebooks also offer two very useful features. First of all, the developer is responsible for defining the order of execution of the cells and, secondly, the developer can take advantage of the available and easy to use GPUs that can significantly speed up the training process of the network. The network was developed using Google's Tensorflow 2.0 and Keras libraries. Keras is a user-friendly library that acts as a wrapper over Tensorflow and offers consistent and simple APIs that reduce the developer's effort to the minimum. As a result, using Keras the developer is able to easily and effortlessly create a network of any complexity and define parameters like activation and loss functions in just a few lines. Other remarkable libraries that were used for data pre-processing, result plotting and scientific calculations are, among others: pandas, nltk, numpy, sklearn, matplotlib and also tqdm in order to inspect the progress of the execution.

4.3 Dataset

The dataset we used was taken by the platform *Kaggle* [23] and lists more than 200 thousands posts from Reddit. Reddit is a social media platform where users can post, using nicknames. Reddit is organized in groups based on the subject of the posts and these groups are called "subreddits". Our dataset's posts are obtained from "SuicideWatch" and "Depression" subreddits and contain posts that go back to 2008. The posts are labeled as "suicidal" and "non-suicidal", depending on whether the content of the text indicates predisposition to suicidal acts.

4.4 Data Pre-Processing

In order to achieve the best results possible, it is of great importance that the input data are not misleading for the model. If the data are of bad quality, it becomes impossible for the model to locate patterns and, by extension, learn. With garbage data, even the best algorithms fail in producing decent results. For this reason, the very first step of this project was to pre-process and clean the dataset in order to transform it and make the most out of it.

4.4.1 Balancing Data

To begin with, it is preferred that the dataset is balanced, meaning that, since we are talking about binary classification, the data are equally distributed between the two classes. If the data are not balanced, then it is possible that the metrics we have used in order to evaluate the predictive capability of our model wrongfully show that the model is successful. To better explain this, let's say, for example, that we use accuracy as a metric to evaluate our model, defined as the percentage of successful predictions on all predictions made. If the data are imbalanced and 90% of the data are labeled as they belong to class A and our model predicts that all data belong to class A, then we achieve an accuracy of 90% but this does not mean that our model has learnt to discriminate the two classes. This problem could be solved by applying more metrics, but there is an even more important one arising than just a misleading evaluation metric. The actual problem is that if the dataset is imbalanced then it is probable that the model does not have enough information to extract the distinguishing features between the two classes [4]. In our case the dataset was balanced so there was no need to experiment with techniques and algorithms for imbalanced datasets.

4.4.2 Data Cleansing

Since the purpose of this project is to create a model that can understand the text it is processing, it is important to simplify it enough in order to make the model's work easier. The process during which we simplify the text is called *Data Cleansing*.

First of all, we convert all uppercase letters to lowercase so that the same words have the same meaning no matter how they are written, given that they are spelled correctly. We will separately process words that are spelled incorrectly in future steps. Furthermore, we remove punctuation, numbers and emojis since the purpose of this project is to train our model on plain text only, even though both emojis and punctuation are important means of expression on social media platforms.

The next step is to remove stopwords, meaning that our final text will contain only words that are actually useful for the model and this process will exclude frequently used words that their absence will not make a big difference in the meaning of the sentence, such as articles and pronouns. An equally important step is *lemmatization*, during which words with the same root and similar meanings are replaced by one single word. For example, the word "people" will be replaced by the word "person".

These last two steps (removing stopwords and lemmatization) have one more advantage that needs to be mentioned. By removing words that are not necessary to understand the meaning of the sentence and replacing others, we also reduce the volume of the different words used in our dataset. Therefore, our data are more flexible and easy to manipulate and it also becomes less time-consuming to create embeddings and train models.

Finally, we observed that some words are spelled incorrectly, which is a usual phenomenon on social media. Misspelled words might make it difficult for the model to capture the meaning of the text and also a lot of versions of the same misspelled word will increase the volume of our vocabulary (list of different words used in the dataset). Thus, using the library TextBlob we tried to fix these words in order to be spelled correctly. The issue that arose was that some misspelled words and abbreviations were replaced by words with a totally different meaning, changing completely the substance of the sentence. After some experiments, we decided that replacing these words was more harmful than beneficial, so we decided to skip this step, since it did not make a significant difference in the volume of the vocabulary, and thus the training time, anyway.

4.5 Embeddings

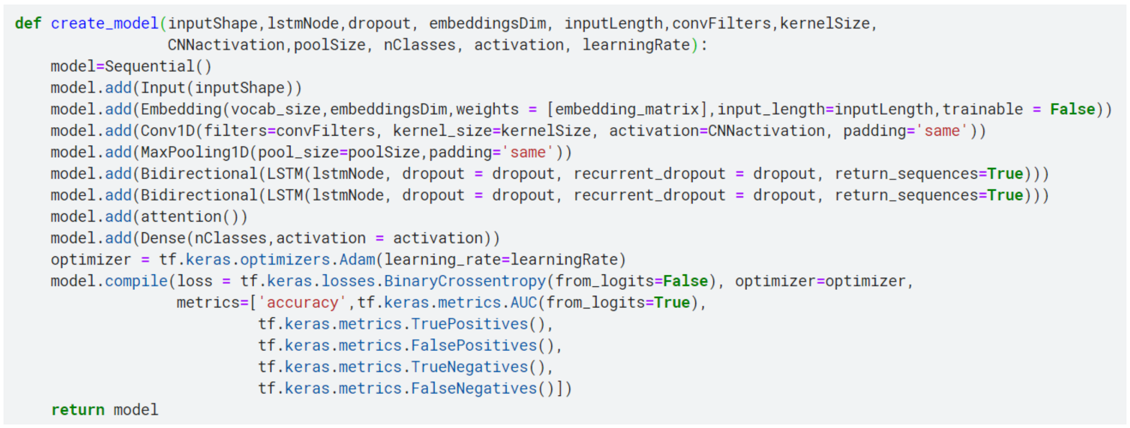
Word embeddings, also known as *word vectors*, is a way of representing words using numerical vectors, taking into account the semantics of each word so as words that appear in the same context have similar representations [36]. In other words, we comprehend the meaning of a word by its context. There are two major model families of word vectors in literature:

- Global matrix factorization methods, like *LSA* [1], that are methods which use large matrices to capture statistical information from the text.
- Local context window methods, like *skip-gram of Mikolov* [35], that are methods which capture the meaning of the word within local context windows.

Both these families have drawbacks. Even though global matrix factorization methods efficiently extract statistical information from the corpus, they are poor in the word analogy task, meaning that they fail in finding similarities between pairs of words. On the other hand, local context windows methods achieve good results in the analogy task but they are unable to utilize the statistics of the corpus since the models are trained on local context windows that do not overlap with each other.
In order to overcome this obstacle and render the meaning of the words in our text the best way possible, we decided to use *GloVe* embeddings [6], which is a combination of the above methods. In GloVe embeddings, with the use of a co-occurrence matrix, the statistics of the corpus are captured directly by the model, achieving better results than other methods.
The other great advantage of Glove is that there are pre-trained embeddings published by Stanford available to download [31] and use directly, therefore saving training time. Despite the important advantage of saving time by the application of pre-trained embeddings, there are some inconveniences that prevented us from using them. To be more specific, the pre-trained embeddings Stanford offers do not capture very well the meaning of the words used in our dataset, since they were trained on completely different datasets (Wikipedia and Twitter posts). As someone could imagine, the embeddings extracted from the Wikipedia dataset are based on an entirely different vocabulary and, as a result, many words used in our dataset would not find a match in the Wikipedia embeddings. Strangely,

some words would not find a match in the Twitter embeddings either, even though both datasets were extracted from social media platforms, but diving a little deeper into the problem we understand this inconsistency and we impute it to the fact that the two platforms address different audiences and age groups that use different language. As a result, we proceeded to train our own Glove Embeddings following Stanford's instructions [7], since we have a sufficiently big dataset.

4.6 Model Design



```
def create_model(inputShape,lstmNode,dropout, embeddingsDim, inputLength,convFilters,kernelSize,
                 CNNactivation,poolSize, nClasses, activation, learningRate):
    model=Sequential()
    model.add(Input(inputShape))
    model.add(Embedding(vocab_size,embeddingsDim,weights = [embedding_matrix],input_length=inputLength,trainable = False))
    model.add(Conv1D(filters=convFilters, kernel_size=kernelSize, activation=CNNactivation, padding='same'))
    model.add(MaxPooling1D(pool_size=poolSize,padding='same'))
    model.add(Bidirectional(LSTM(lstmNode, dropout = dropout, recurrent_dropout = dropout, return_sequences=True)))
    model.add(Bidirectional(LSTM(lstmNode, dropout = dropout, recurrent_dropout = dropout, return_sequences=True)))
    model.add(attention())
    model.add(Dense(nClasses,activation = activation))
    optimizer = tf.keras.optimizers.Adam(learning_rate=learningRate)
    model.compile(loss = tf.keras.losses.BinaryCrossentropy(from_logits=False), optimizer=optimizer,
                  metrics=['accuracy',tf.keras.metrics.AUC(from_logits=True),
                           tf.keras.metrics.TruePositives(),
                           tf.keras.metrics.FalsePositives(),
                           tf.keras.metrics.TrueNegatives(),
                           tf.keras.metrics.FalseNegatives()])
    return model
```

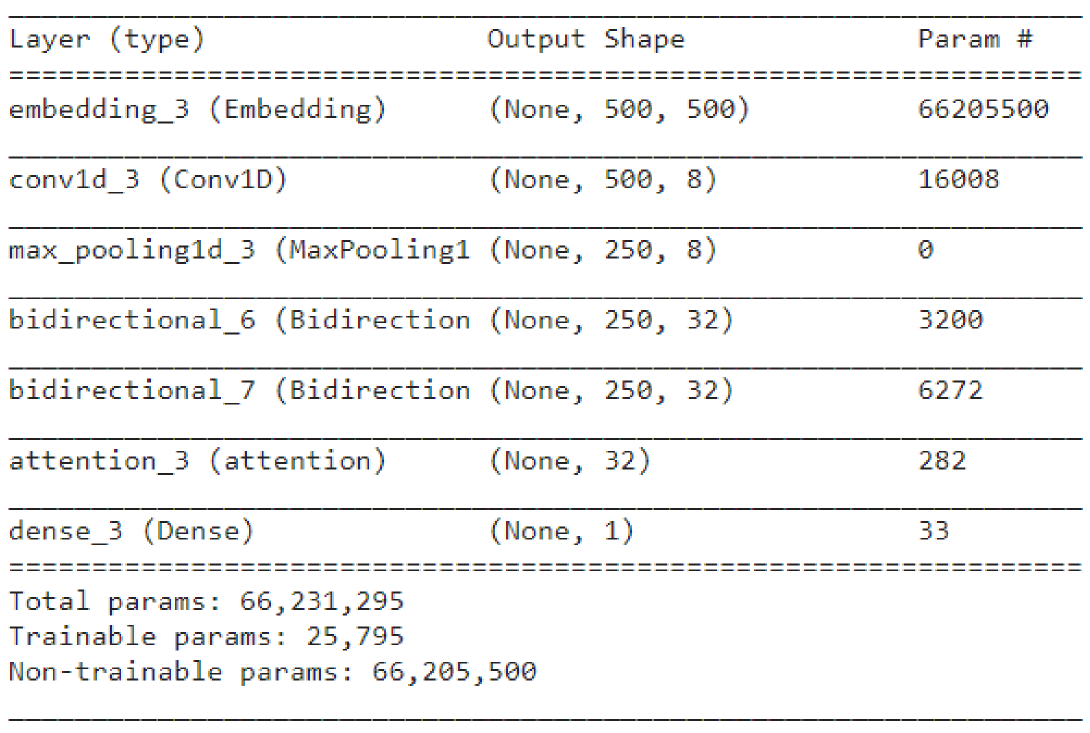
Figure 15: Model Architecture

In the figure above (figure 14), the code that generates the model is displayed. As we can see, a hybrid model was chosen with both CNN and LSTM layers, and also an attention mechanism on top of the second LSTM layer. Finally, one dense layer is added to support the classification of the input.

The model starts with an input layer in order to instantiate the tensors used by the model. Consequently, an embeddings-layer is added, whose utility is extensively explained in the above sections of this thesis. As we can see, this layer is flagged as not trainable since the embeddings have been trained separately in previous steps. After that, there is a CNN layer accompanied with a Max Pooling one, whose combination not only extracts important high level features, but also reduces the dimensionality of the data and thus makes the training process faster. The Max Pooling layer is followed by two bi-LSTM layers, a number which was decided after multiple tests and experiments. LSTMs are probably the most important layers of this architecture since these are the ones that will detect the relationships between the words and sentences and learn patterns. In order to boost the performance of the LSTM layers, the model was enhanced with a custom attention layer based on Bahdanau's attention mechanism. Finally, a dense layer is in charge of the final classification between the two classes (suicidal and non-suicidal posts). As a result, the generated model is displayed in figure 15 below.

4.7 Parameter tuning

Some important parameters used in the architecture and during the training of the model are displayed in the table below (Table 1):



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_3 (Embedding)      (None, 500, 500)          66205500
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 500, 8)            16008
_________________________________________________________________
max_pooling1d_3 (MaxPooling1 (None, 250, 8)            0
_________________________________________________________________
bidirectional_6 (Bidirection (None, 250, 32)           3200
_________________________________________________________________
bidirectional_7 (Bidirection (None, 250, 32)           6272
_________________________________________________________________
attention_3 (attention)      (None, 32)                282
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 33
=================================================================
Total params: 66,231,295
Trainable params: 25,795
Non-trainable params: 66,205,500
_________________________________________________________________
```

Figure 16: Overview of the created model

Table 1: Parameters used in model architecture and training

Learning Rate	0.01
Epochs	25
Dropout	0.2
CNN Filters	8
Batch Size	1024
Kernel Size	4
Pool Size	2
LSTM Nodes	16
Activation Function	Sigmoid
Loss Function	Binary Cross Entropy

In ML projects the combinations of parameters are uncountable and it is impossible to manually test each one of them, though there is a specific logic behind developers' choices. The changes on some parameter values have an impact on the behavior of others. For example, a model with a lower learning rate will need more epochs to adapt to the problem. Thus, changes to the learning rate cause changes to the number of epochs too. Apart from strictly following guidelines like the one described above, the most common method applied to chose parameters from is through trial and error. Even if some guidelines are followed, they only describe the relationships between parameters and not the amount at which a parameter value should increase or decrease, so trial and error remains an indefeasible part of the process. In the following paragraphs we will attempt to explain the reasons and the importance of each choice we made.
The values of dropout, CNN filters, kernel size and pool size, all of which are not specifically explained, were decided after multiple experiments and those who produced the best results were chosen. Of course, all of our choices were validated by experiments, even those which originate from the theory behind neural networks.

4.7.1 Learning Rate

Learning rate is probably the most important hyperparameter of the model which controls how fast the model adapts to the problem. Smaller learning rates mean that smaller changes are made to the weights each time they update. Consequently, models with smaller learning rates demand a higher number of training epochs. Initially, the default value of Adam optimizer was used, which is 0.001. With this value, as seen in the figure below (figure 16), the accuracy of the model started to decrease after a number of epochs. This behavior could indicate that the learning rate value is either too large for this model and as a result the model bounces around the minimum, as explained in paragraph 2.1.6, or too small so the model is trapped in a local minimum [34]. To overcome this obstacle, we changed the value of the learning rate and after some experiments we found that the most appropriate value was 0.01.

4.7.2 Epochs

Although we expected that an increase in the learning rate value would make the model require less training epochs, multiple experiments showed that more epochs produced slightly better results, so we used 25 epochs to train our model instead of 20 as we used initially. It is probable that even more epochs could produce even better results, but at this point the changes between the last epochs are insignificant and do not indicate great progress in the learning procedure. Furthermore, it should not be forgotten that more epochs mean more training time, thus there is a trade off between greater accuracy and faster training. In this case we attempted to strike a satisfying balance between these two and accelerate the process without compromising the model's reliability.

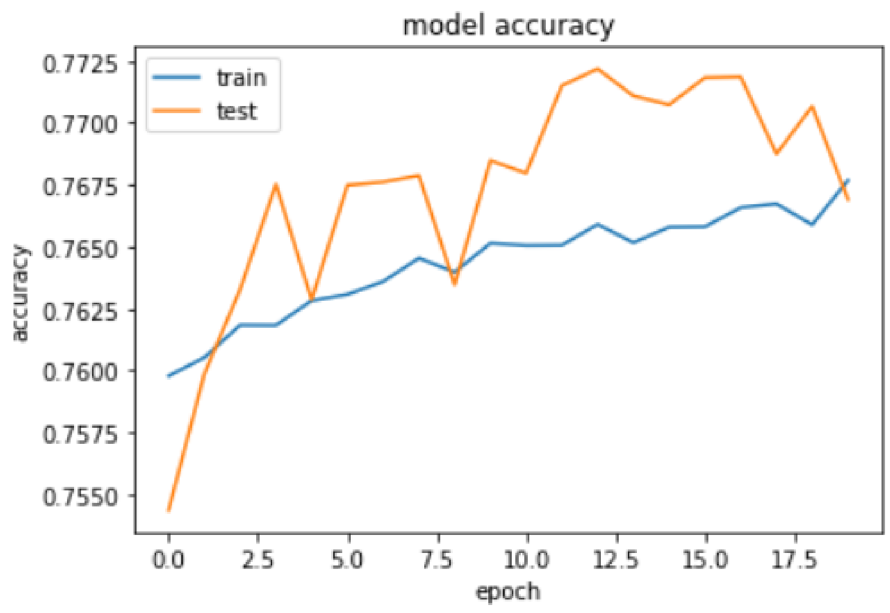


Figure 17: Overview of the created model

4.7.3 Batch Size

Batch size is a hyperparameter that defines the number of samples used to train the model before updating its weights and is strongly attached to the learning rate. The most common practice is that the batch size's values follow the changes of the learning rate's values, meaning that they increase and decrease together. Using small batch sizes usually leads to faster convergence to good solutions since the model starts learning before it sees all the data [29], but it does not mean that the good solution is the best solution possible. Using small batch size could cause the model bouncing around the minimum. On the other hand, bigger batch sizes may lead to poor generalization capability (overfitting). Finally, batch size is limited by the available GPU memory [33]. After some experiments we ended up using 1024 samples in each batch.

4.7.4 Activation and loss functions

Sigmoid function was chosen as an activation function and the main reason why is because it gives an output between (0,1) and is useful for binary classification problems, like the one we are studying. Similarly, binary cross entropy was selected as loss function. Binary cross entropy compares the probability of each prediction made with its expected value which can be ether 0 or 1. Then, it calculates the penalizing error based on the difference between the predicted value and the real one [26]. It is clear from the definition of the binary cross entropy that it is a loss function targeted to binary classification problems.

4.7.5 LSTM nodes

We decided to add 16 LSTM nodes per layer while creating the model. This number was mostly calculated through trial and error, having in mind that we want to adopt the simplest model that can produce the same results and also taking into consideration the dropout rate.

5. RESULTS

5.1 Model Loss

Loss is a value that expresses the summation of errors in each training epoch of our model. The smaller the loss value, the less errors occur, the better our model becomes. To calculate the model's loss, a loss function is used, in our case the binary cross entropy which is suitable for binary classification problems as this one. Each loss function has a unique way to penalize errors. Binary cross entropy's method is described in paragraph 4.7. In the figure that follows below (figure 17) we observe the progress of the loss values during training and validation procedures.

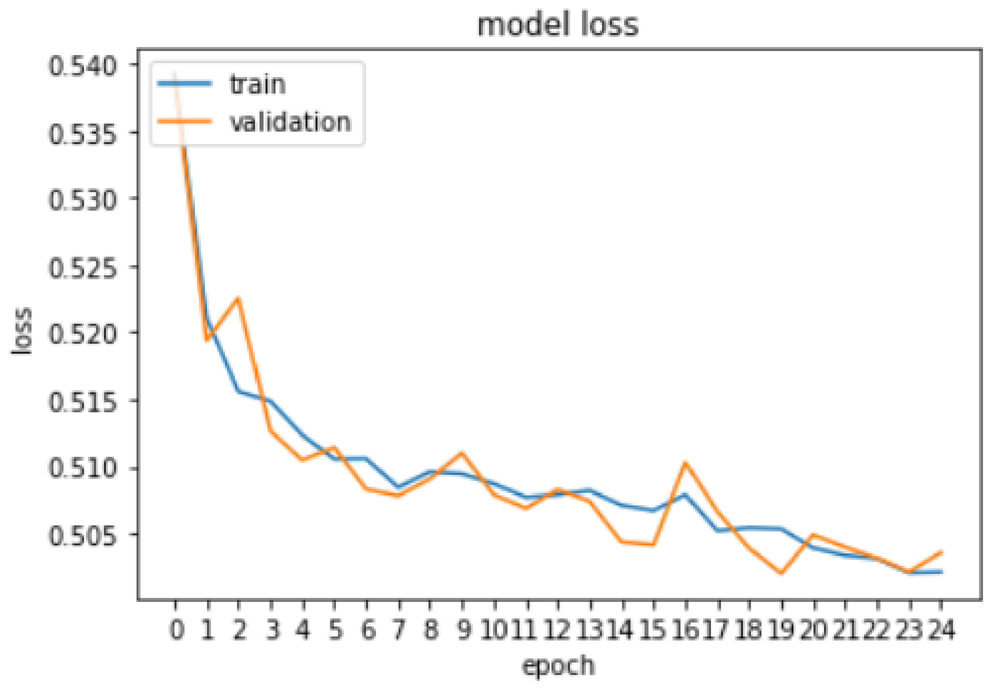


Figure 18: Model Loss

5.2 Model Accuracy

Accuracy is the most frequently used metric to measure the performance of a model. Accuracy is calculated as the number of correct predictions divided by the number of all prediction made:

$$\frac{\textit{Correct Predictions}}{\textit{All Predictions}}$$

Obviously, the higher the accuracy, the better the model is at making correct predictions. Our model managed to achieve an accuracy close to 78% on the test set versus 76% on the training set (figure 18), an accuracy which is above the average but still not a great one. Probably with more epochs we could achieve better results, but as explained above in paragraph 4.7, more epochs mean more training time. Though accuracy is the most common metric, in paragraph 4.4.1 we mentioned that it is not always a representative one and that's the reason why we should not rely only on accuracy to evaluate our model's

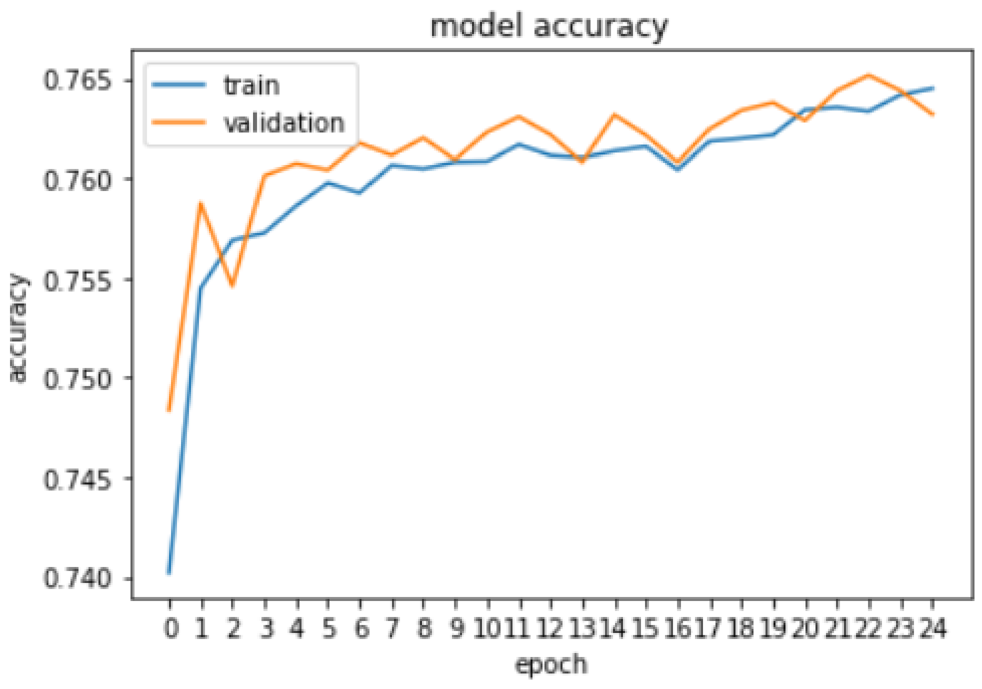


Figure 19: Model Accuracy

performance. In the following paragraph we display the results of a more suitable metric to our classification problem, AUC.

5.3 ROC - AUC curve

AUC - ROC curve, also known as AUROC, is an important performance metric for classification problems. ROC is a probability curve and AUC symbolises the level of separability of the model. In other words, AUC measures the model's ability to distinguish between classes and classify posts correctly. Accordingly, the higher the AUC, the better is the model at separating posts between suicidal and non - suicidal. AUC is plotted with False Positives against True Positives, where True Positives are the samples of suicidal posts that were predicted as such and False Positives are the samples that were predicted as suicidal though they are not. In order to achieve its best performance a model should aim at an AUC of 100%. The developed model reaches an AUC close to 85% indicating a sufficiently good separability. Bellow, the ROC - AUC curve is plotted in figure 19 with data obtained from the test set and in figure 20 we can see AUC changes during the training of the model.

5.4 Comparison to similar models

A model developed within the context of an undergraduate thesis surely cannot be compared to state-of-the-art models like those of Chaitanya Bhargava and Michael Mesfin analysed in paragraph 3.3. Needless to say, it would not be fair to compare them since they are trained and tested on different datasets than the one we used to train our model.

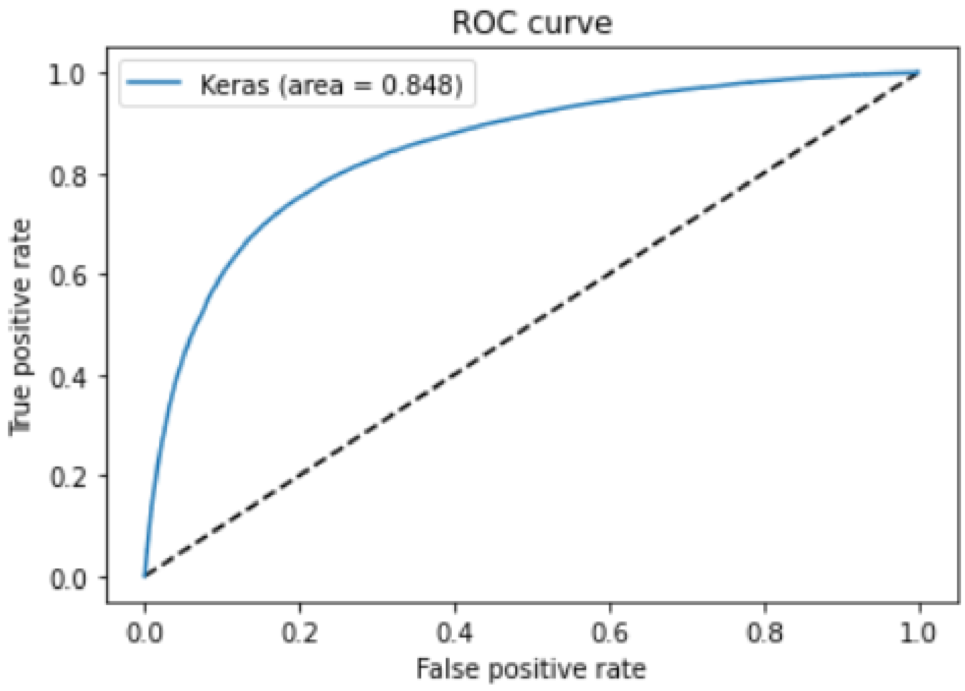


Figure 20: Model ROC - AUC curve

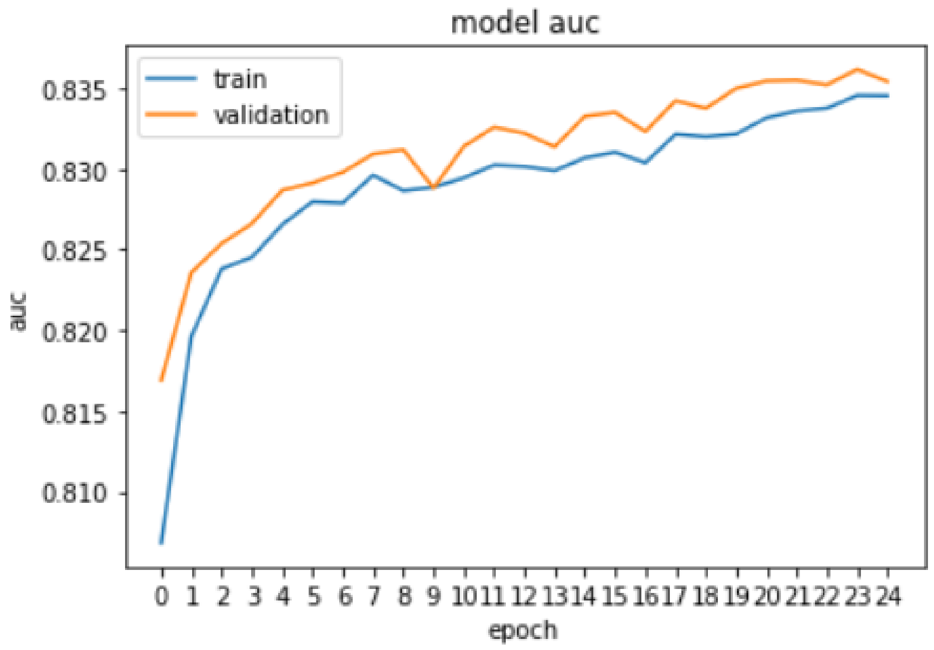


Figure 21: Model AUC

6. ETHICS IN DEPRESSION DETECTION ON SOCIAL MEDIA PLATFORMS

The rising popularity of social media during the last decades has offered the opportunity of conducting easy, massive studies on people's psychology and behavior. Many studies have taken place on social media or have obtained datasets by them, whether the participating users know it or not. However, to what extent is it morally acceptable to perform studies on people who have not, most of the times, given consent, or the consent they gave was blind because they skipped reading the extensive data usage policy of a social media platform or because the terms of service were written in an incomprehensible way? Many studies refer to social media, but very few of them are found to be actually concerned about the ethical and legal implications. A well known example of a study that was later considered ethically unacceptable [8] is Facebook's emotional contagion study [10] in 2012, where no informed consent was clearly obtained by the participants. Moreover, when the users involved learnt about the experiment they were taking part into involuntarily, they could not withdraw from the study. Especially when the study contains potential medical data, like a study about depression and suicide ideation detection does, it is extremely important that participants' consent is ensured and data privacy is protected. The most important subject to discuss is how the balance between the study's integrity and accuracy on one hand, and the feeling of security of the people participating on the other, can be maintained.

Data privacy is dependent on each social media platform, meaning that different social media have different ways to express their policies about their data privacy practices and offer different settings available to the user. This variety of settings and data privacy policies can be very confusing, since they change regularly. Also, the language used in these texts is usually very strange and difficult to comprehend for the average user in order to create backdoors according to which the platform can use the user's data for multiple, not always described, purposes. For example, Twitter mentions the following in their terms of service: “You retain your rights to any Content you submit...What's yours is yours — you own your Content...” and “...you grant us a worldwide, non-exclusive, royalty-free license (with the right to sublicense) to use, copy, reproduce, process, adapt, modify, publish, transmit, display and distribute such Content in any and all media or distribution methods now known or later developed...” [39]. Overall, social media data policies are not user-friendly at all.

In the case of a research or of an actual service about predicting suicidal behaviors, users should be well-informed about the purposes, the outcomes and the possible implications of such an undertaking before consenting [14]. Consent should never be implied in such cases and data should not be used for purposes other than those described in the terms of service of each platform. Of course, it should be mentioned that medical data are specifically protected by the Declaration of Helsinki [44] which describes the ethical principles for medical research.

To conclude, in the trade off between helping people in need and potentially saving lives on one side and risking the security of participants' data on the other [12], we should not pick a side. Research about legal and ethical issues that arise in the technological sphere is still in a rather embryonic state, but progress has been made through the years. The technology to predict mental health issues exists and the real question is not when it will be applied but how, so that it does not violate people's personal data.

7. CONCLUSION

7.1 Conclusions

To conclude, we could say that this thesis proves that the technology to predict suicidal ideation exists, as mentioned above. With this project we managed to successfully create a model that will be able to automatically forecast if a user's post implies suicidal thoughts, a tool that could be used to help people in need and especially those who are in denial of their condition refusing to seek professional help.

Certainly, it would be insufficient to rely exclusively on this project to develop a service that will go to production, since the purpose of this thesis is to be engaged with the academic research of a topic of our interest and to use our knowledge to create something from scratch and test its performance through experiments, rather than creating a commercial product. The development of a profitable service would need more time and resources, but, if these things were given, the implementation would be feasible.

7.2 Future work

This thesis dealt only with a small fraction of what could be implemented in order to detect suicidal ideation and was limited to examine exclusively textual data, since the purpose was to study the subject from the perspective of an NLP problem. Following this approach, it is almost certain that minor changes to the implementation, the value of a parameter or a change in the model architecture, could produce much better results despite our best efforts to optimize the model as much as possible. Also, in future modifications it would be useful if we used data obtained by different social media platforms used by people of different age in order to create a dataset of posts from all kinds of demographic groups.

Beyond our implementation, in order to achieve better results, a more spherical approach should be considered, namely an approach that combines data of different types like images, audio and video. We should not forget that text is only one of all means of communication people use every day. After all, two of the most popular social media platforms (Instagram and TikTok) are based entirely on audiovisual content. People using these platforms should not lack the opportunity to be helped if in need.

Last but not least, future work includes research on how this technology could be integrated in social media platforms without violating users' personal data. Of course, a user who doesn't wish for his/her posts to be examined could avoid subscribing in a platform that uses this technology, but this would mean that this person would miss a big part of the modern social life. The solution is not to make people fit into technology, but to make technology fit people's needs.

ABBREVIATIONS

AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
AUC	Area Under the Curve
BERT	Bidirectional Encoder Representations from Transformer
Bi-LSTM	Bidirectional Long Short Term Memory
CNN	Convolutional Neural Network
DL	Deep Learning
ELMo	Embeddings from Language Model
GloVe	Global Vectors
GPU	Graphics Processing Unit
LSA	Latent Semantic Analysis
LSTM	Long Short Term Memory
ML	Machine Learning
MSE	Mean Square Error
MTL	Multi Task Learning
MTM	Medium Term Memory
NER	Named Entity Recognition
NLP	Natural Language Problem
NN	Neural Network
RCNN	Region Based Convolutional Neural Net-works
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristics
STM	Short Term Memory
SVM	Support Vector Machine
VGG	Visual Geometry Group

REFERENCES

[1] Thomas K Landauer, Peter W. Foltz, and Darrell Laham. “An introduction to latent semantic analysis”. In: *Discourse Processes* 25 (2-3 Jan. 1998), pp. 259–284.

[2] “Data Mining: Concepts, Models, Methods, and Algorithms, Second Edition”. In: *Data Mining: Concepts, Models, Methods, and Algorithms: Second Edition* (Jan. 2002). DOI: 10.1109/9780470544341.

[3] David D. Luxton, Jennifer D. June, and Jonathan M. Fairall. “Social media and suicide: A public health perspective”. In: *American Journal of Public Health* 102.SUPPL. 2 (May 2012). ISSN: 00900036. DOI: 10.2105/AJPH.2011.300608.

[4] Victoria López et al. “An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics”. In: *Information Sciences* 250 (Nov. 2013), pp. 113–141. ISSN: 00200255. DOI: 10.1016/J.INS.2013.07.007. URL: http://dx.doi.org/10.1016/j.ins.2013.07.007.

[5] Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. “Neural Machine Translation by Jointly Learning to Align and Translate”. In: *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings* (Sept. 2014). DOI: 10.48550/arxiv.1409.0473. arXiv: 1409.0473. URL: https://arxiv.org/abs/1409.0473v7.

[6] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. “GloVe: Global Vectors for Word Representatio In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

[7] *GloVe: Global Vectors for Word Representation*. 2015. URL: https://github.com/stanfordnlp/GloVe.

[8] Jukka Jouhki et al. “Facebook’s Emotional Contagion Experiment as a Challenge to Research Ethics”. In: *Media and Communication* 4.4 (2016), pp. 75–85. ISSN: 21832439. DOI: 10.17645/MAC.V4I4.579. URL: https://jyx.jyu.fi/handle/123456789/51619.

[9] Chris Olah and Shan Carter. “Attention and Augmented Recurrent Neural Networks”. In: *Distill* 1 (9 Sept. 2016), e1. ISSN: 2476-0757. DOI: 10.23915/DISTILL.00001. URL: http://distill.pub/2016/augmented-rnns.

[10] Evan Selinger and Woodrow Hartzog. “Facebook’s emotional contagion study and the ethical problem of co-opted identity in mediated environments where users lack control”. In: *Research Ethics* 12 (1 2016), pp. 35–43. DOI: 10.1177/1747016115579531.

[11] Mandar Deshpande and Vignesh Rao. “Depression detection using emotion artificial intelligence”. In: *2017 International Conference on Intelligent Sustai-nable Systems (ICISS)* (2017), pp. 858–862.

[12] Glen Coppersmith et al. “Natural Language Processing of Social Media as Screening for Suicide Risk.” In: *Biomedical informatics insights* 10 (Aug. 2018), p. 1178222618792860. ISSN: 1178-2226. DOI: 10.1177/1178222618792860. URL: http://www.ncbi.nlm.nih.gov/pubmed/30158822%20http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC6111391.

[13] Jingcheng Du et al. “Extracting psychiatric stressors for suicide from social media using deep learning”. In: *BMC medical informatics and decision making* 18.Suppl 2 (July 2018), p. 43. ISSN: 1472-6947. DOI: 10.1186/s12911-018-0632-8. URL: https://europepmc.org/articles/PMC6069295.

[14] Ruth F Hunter et al. “Ethical Issues in Social Media Research for Public Health”. In: *American Journal of Public Health* 108.3 (2018), pp. 343–348. DOI: 10.2105/AJPH.2017.304249. URL: https://doi.org/10.2105/AJPH.2017.304249.

[15] Ahmed Husseini Orabi et al. “Deep Learning for Depression Detection of Twitter Users”. In: *CLPsych@NAACL-H* 2018.

[16] Tao Gui et al. “Cooperative Multimodal Approach to Depression Detection in Twitter”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (July 2019), pp. 110–117. DOI: 10.1609/aaai.v33i01.3301110. URL: https://ojs.aaai.org/index.php/AAAI/article/view/3775.

[17] Beakcheol Jang et al. “Bi-LSTM Model to Increase Accuracy in Text Classification: Combining Word2vec CNN and Attention Mechanism”. In: *Applied Sciences* 10.17 (2020). ISSN: 2076-3417. DOI: 10.3390/app10175841. URL: https://www.mdpi.com/2076-3417/10/17/5841.

[18] Chenhao Lin et al. “SenseMood: Depression Detection on Social Media”. In: *Proceedings of the 2020 International Conference on Multimedia Retrieval*. ICMR ’20. Dublin, Ireland: Association for Computing Machinery, 2020, pp. 407–411. ISBN: 9781450370875. DOI: 10.1145/3372278.3391932. URL: https://doi.org/10.1145/3372278.3391932.

[19] Anshu Malhotra and Rajni Jindal. “Multimodal Deep Learning based Framework for Detecting Depression and Suicidal Behaviour by Affective Analysis of Social Media Posts”. In: *EAI Endorsed Transactions on Pervasive Health and Technology* 6.21 (Jan. 2020). DOI: 10.4108/eai.13-7-2018.164259.

[20] Yaakov Ophir et al. “Deep Neural Networks Detect Suicide Risk from Textual Facebook Posts”. In: (June 2020). DOI: 10.31234/osf.io/k47hr.

[21] Michael Mesfin Tadesse et al. “Detection of Suicide Ideation in Social Media Forums Using Deep Learning”. In: *Algorithms* 13.1 (2020). ISSN: 1999-4893. DOI: 10.3390/a13010007. URL: https://www.mdpi.com/1999-4893/13/1/7.

[22] Chaitanya Bhargava et al. “Depression Detection Using Sentiment Analysis of Tweets”. In: *Turkish Journal of Computer and Mathematics Education (TURCOMAT)* 12.11 (2021), pp. 5411–5418.

[23] “Suicide and Depression Detection”. In: (2021). URL: https://www.kaggle.com/datasets/nikhileswarkomati/suicide-watch.

[24] *AI-Definitions-HAI*. URL: https://hai.stanford.edu/sites/default/files/2020-09/AI-Definitions-HAI.pdf.

[25] *Bi-LSTM*. URL: https://medium.com/@raghavaggarwal0089/bi-lstm-bc3d68da8bd0.

[26] *Binary Cross Entropy/Log Loss for Binary Classification*. URL: https://www.analyticsvidhya.com/blog/2021/03/binary-cross-entropy-log-loss-for-binary-classification/.

[27] *Conv Nets: A Modular Perspective - colah’s blog*. URL: http://colah.github.io/posts/2014-07-Conv-Nets-Modular/.

[28] *Depression*. URL: https://www.who.int/news-room/fact-sheets/detail/depression.

[29] *Effect of batch size on training dynamics*. URL: https://medium.com/mini-distill/effect-of-batch-size-on-training-dynamics-21c14f7a716e.

[30] *GBD Results Tool | GHDx*. URL: https://ghdx.healthdata.org/gbd-results-tool?params=gbd-api-2019-permalink/d780dffbe8a381b25e1416884959e88b.

[31] *GloVe: Global Vectors for Word Representation*. URL: https://nlp.stanford.edu/projects/glove/.

[32] *Gradient Descent: A Quick, Simple Introduction | Built In*. URL: https://builtin.com/data-science/gradient-descent.

[33] *How to Break GPU Memory Boundaries Even with Large Batch Sizes*. URL: https://towardsdatascience.com/how-to-break-gpu-memory-boundaries-even-with-large-batch-sizes-7a9c27a400ce.

[34] *How to Configure the Learning Rate When Training Deep Learning Neural Networks*. URL: https://machinelearnin com/learning-rate-for-deep-learning-neural-networks/.

[35] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. URL: http://ronan.collobert.com/senna/.

[36] *Neural Networks & Word Embeddings | by Nwamaka Imasogie | Nwamaka Imasogie’s Machine Learning and Artificial Intelligence Projects | Medium*. URL: https://medium.com/nwamaka-imasogie/neural-networks-word-embeddings-8ec8b3845b2e.

[37] *Suicide Prevention | Online Well-being*. URL: https://www.facebook.com/safety/wellbeing/suicideprevention.

[38] *The Basics of Recurrent Neural Networks (RNNs) | by Ben Khuong | Towards AI*. URL: https://pub.towardsai.net/whirlwind-tour-of-rnns-a11effb7808f.

[39] *Twitter Terms of Service*. URL: https://twitter.com/en/tos.

[40] *Understanding LSTM Networks – colah’s blog*. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[41] *Understanding the Mathematics behind Gradient Descent.* | *by Parul Pandey* | *Towards Data Science*. URL: https://towardsdatascience.com/understanding-the-mathematics-behind-gradient-descent-dde5dc9be06e.

[42] *What is Artificial Intelligence (AI)?* | *IBM*. URL: https://www.ibm.com/cloud/learn/what-is-artificial-intelligence.

[43] *What is the suicide rate among persons with depressive disorder (clinical depression)?* URL: https://www.medscape.com/answers/286759-14675/what-is-the-suicide-rate-among-persons-with-depressive-disorder-clinical-depression?reg=1 (visited on 05/14/2022).

[44] *WMA Declaration of Helsinki – Ethical Principles for Medical Research Involving Human Subjects – WMA – The World Medical Association*. URL: https://www.wma.net/policies-post/wma-declaration-of-helsinki-ethical-principles-for-medical-research-involving-human-subjects/.