NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATION

BSc THESIS

Convolutional Neural Networks and their Application in Cancer Diagnosis based on RNA-Sequencing

Maria Anna G. Kanellaki

Supervisor: **Panagiotis Stamatopoulos,** Assistant Professor

ATHENS

February 2022

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Συνελικτικά Νευρωνικά Δίκτυα και η Εφαρμογή τους στη Διάγνωση Καρκίνου βάσει της Αλληλουχίας του RNA

Μαρία Άννα Γ. Κανελλάκη

Επιβλέπων: **Παναγιώτης Σταματόπουλος**, Επίκουρος Καθηγητής

ΑΘΗΝΑ

Φεβρουάριος 2022

BSc THESIS

Convolutional Neural Networks and their Application in Cancer Diagnosis based on RNA-Sequencing

Maria Anna G. Kanellaki
S.N.: 1115201400060

SUPERVISOR: **Panagiotis Stamatopoulos,** Assistant Professor

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Συνελικτικά Νευρωνικά Δίκτυα και η Εφαρμογή τους στη Διάγνωση Καρκίνου βάσει της Αλληλουχίας του RNA

Μαρία Άννα Γ. Κανελλάκη
Α.Μ.: 1115201400060

ΕΠΙΒΛΕΠΩΝ: **Παναγιώτης Σταματόπουλος**, Επίκουρος Καθηγητής

ABSTRACT

Gene expression analysis is the study of the way genes are transcribed to synthesize functional gene products, functional RNA species, or protein products. Its study can provide insights of cellular processes, such as cellular differentiation and abnormal pathological processes.

Cancer is a genetic disease where genetic variations cause abnormally functioning genes that appear to alter expression. Proteins, being the final products of gene expression, define the phenotypes and biological processes. Therefore, detecting gene expression levels can be used for cancer diagnosis, prognosis, and even treatment prediction.

This thesis will be analyzing the theory and applications of Deep Learning. It will then apply Deep Learning (DL) and in particular a Convolutional Neural Network (CNN) as a means for the diagnosis of multiple cancer types (pan-cancer classification) using gene expression data and specifically RNA-sequencing.

The Cancer Genome Atlas (TCGA) data, which consists of RNA-sequencing, will be preprocessed and then embedded into multiple two-dimensional (2D) images. These images will then be applied to a Convolutional Neural Network which will classify them into 33 types of cancer, in an attempt to achieve the highest possible diagnosis accuracy.

SUBJECT AREA: Deep Learning
KEYWORDS: Convolutional Neural Network, Classification, Cancer Diagnosis, Gene Expression, RNA-Sequencing

ΠΕΡΙΛΗΨΗ

Η έκφραση γονιδίων αποτελεί τη μελέτη της λειτουργίας της γονιδιακής μεταγραφής, κατά την οποία συνθέτονται γονιδιακά προϊόντα, είδη RNA ή πρωτεΐνες. Η μελέτη της παρέχει την κατανόηση των κυτταρικών λειτουργιών, όπως η κυτταρική διαφοροποίηση και οι μη φυσιολογικές παθολογικές λειτουργίες.

Ο καρκίνος αποτελεί μία γενετική ασθένεια όπου γενετικές παραλλαγές προκαλούν μη φυσιολογικές λειτουργίες στα γονίδια και τροποποιούν την έκφραση τους. Οι πρωτεΐνες, οι οποίες αποτελούν το τελικό αποτέλεσμα της έκφρασης γονιδίων, καθορίζουν τους φαινοτύπους και τις βιολογικές λειτουργίες. Συνεπώς, η ανίχνευση των επιπέδων έκφρασης γονιδίων δύναται να χρησιμοποιηθεί στη διάγνωση, την πρόγνωση, ακόμα και την επιλογή της θεραπείας του καρκίνου.

Σε αυτή την πτυχιακή θα αναλυθεί η θεωρία και οι εφαρμογές της Βαθειάς Μάθησης. Στη συνέχεια, θα εφαρμοστεί η Βαθειά Μάθηση και πιο συγκεκριμένα ένα Συνελικτικό Νευρωνικό Δίκτυο, ως μέσο για τη διάγνωση πολλαπλών τύπων καρκίνου (κατηγοριοποίηση καρκίνων) χρησιμοποιώντας δεδομένα έκφρασης γονιδίων, και πιο συγκεκριμένα αλληλουχίες RNA.

Τα δεδομένα του «The Cancer Genome Atlas» (TCGA) αποτελούνται από αλληλουχίες RNA. Θα επεξεργαστούν σε πρώτο επίπεδο και μετά θα μετατραπούν σε πολλαπλές δισδιάστατες εικόνες. Οι εικόνες αυτές θα εισαχθούν σε ένα Συνελικτικό Νευρωνικό Δίκτυο, το οποίο θα τις κατηγοριοποιήσει σε 33 τύπους καρκίνου, αποσκοπώντας στην διάγνωση με τη μέγιστη δυνατή ακρίβεια.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Βαθειά Μάθηση
ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Συνελικτικό Νευρωνικό Δίκτυο, Κατηγοριοποίηση, Διάγνωση Καρκίνου, Έκφραση Γονιδίων, Αλληλουχία RNA

AKNOWLEDGMENTS

I would like to thank my supervisor, Assistant Professor Panagiotis Stamatopoulos, for his invaluable help and guidance, as well as his remarkably prompt responsiveness.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

PREFACE

This thesis was written as a part of the BSc program of studies at the Department of Informatics and Telecommunications of the National and Kapodistrian University of Athens.

1. INTRODUCTION

Artificial Intelligence (AI) has always been an ambiguous field with various definitions and interpretations. It is “a field within computer science attempting to build enhanced intelligence into computer systems” [1].

Machine Learning (ML) is a subset of AI methods. According to [2], “the field of Machine Learning is concerned with the question of how to construct computer programs that automatically improve with experience”. Its aim is the creation of machines with learning capabilities and can be used in various fields.

Deep Learning (DL) - also referred to as Representation Learning in recent literature [3] - is a branch of Machine Learning which enables computer systems to learn and improve with experience. DL studies deep Artificial Neural Networks (ANNs). The term Artificial Neural Networks refers to mathematical models that are inspired by the human brain’s neurons [4]. These models consist of multiple layers of nonlinear computational units, used to extract and process data.

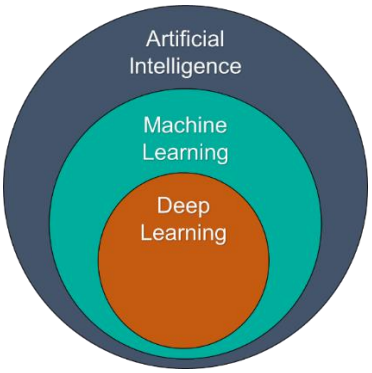

Figure 1.1: AI vs ML vs DL

AI, ML and DL are used in various fields and problems, the most important being: Classification [5], Image Identification [6], Speech Recognition [7], Economics [8], Agriculture [9] and Healthcare [10]. This paper will be focusing on the application of Deep Learning in the field of Healthcare and more specifically Biomedicine.

Healthcare is being “reshaped” by AI. AI, ML and DL enable healthcare stakeholders and medical professionals to identify healthcare needs and solutions faster and with higher accuracy with the usage of data patterns, to make informed medical decisions quickly [11].

Cancer is one of the most dangerous and life-threatening diseases for humans. It is related to abnormal and uncontrolled growth of the cell which can invade or spread to other parts of the body [12]. A fast and accurate prognosis is essential and increases the possibility of cure. Different types of cancer detection and classification, using machine assistance have opened up a new research area for early detection [13].

This thesis will be focusing on the classification (diagnosis) of cancer using Deep Learning (DL), based on genomic data, and more precisely ribonucleic acid sequencing (RNA-seq) data.

2. NEURAL NETWORKS

2.1 Structure and Function

Artificial Neural Networks (ANNs) are inspired by the human brain, both in their structure and in the way they function. They consist of processing units or nodes called neurons. Artificial neurons have the ability to receive and transmit signals to other neurons to which they are connected via edges, simulating biological neurons. Neurons are organized in groups called layers.

The edges store some values called weights which determine the inputs and outputs of the layers [14]. Weights consist of real numbers that adjust accordingly during the training process.

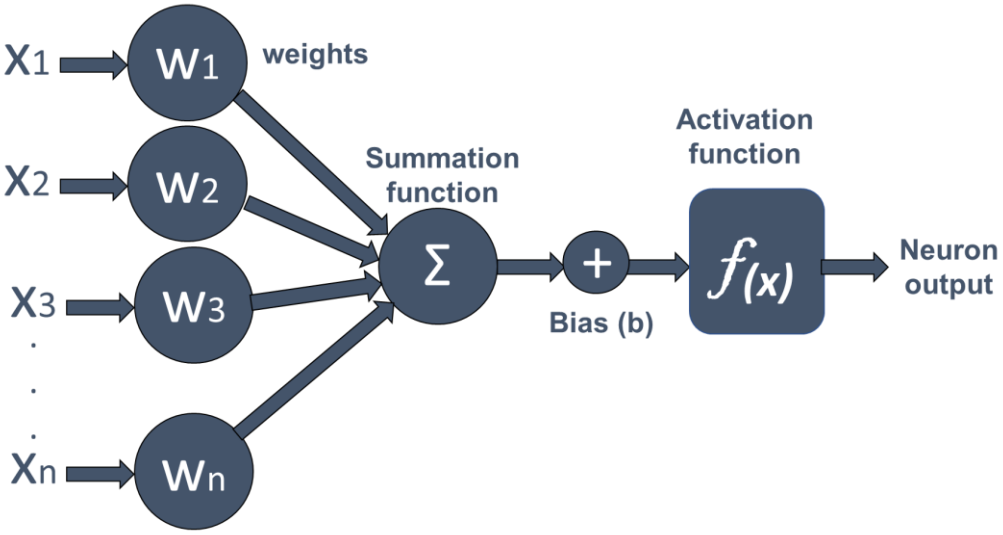

Figure 2.1: Artificial Neuron

A neuron consists of multiple inputs (x_i) and one output (y). A weight is calculated for each input and all weights are summed by a summation function F, which according to [15], is defined as:

$$F = \sum_{i}^{n} x_i w_i \tag{1}$$

According to [16], an artificial neuron produces its output when the weighted sum, calculated by equation (1), is larger than a threshold value θ:

$$\sum_{i}^{n} x_i w_i - \theta > 0 \tag{2}$$

According to [16], the final output of the neuron is produced by applying an activation function g to the weighted sum, as follows:

$$y_j = g\left(\sum_{i=0}^{n} w_{i,j}\, xa_i\right) \tag{3}$$

Neurons can also include a bias b, a constant which is added to the output in order to adjust it, by shifting the activation function to produce the best results. According to [15], in this case the final output of the neuron is calculated as follows:

$$y_j = g\left(b + \sum_{i=0}^{n} w_{i,j}\, x_i\right) \tag{4}$$

Training a neuron means choosing the suitable weights and bias to produce the best outputs for each input. To achieve that, a cost function needs to be minimized. Cost functions are used to evaluate the outputs of the network by comparing them to the desired results. They are described more extensively below (Cost Functions).

As mentioned above, neurons are divided into layers. The first one is the input layer, where the input is received and the last one is the output layer, where the output is produced. Between these, several layers can exist, which are called hidden. Each layer processes its input and advances its output to the next one, until the output layer (neuron) is reached. There, the final output of the network is produced by the chosen activation function. Since there are numerous activation functions existing, the choice of the final activation function depends on the function the model executes.

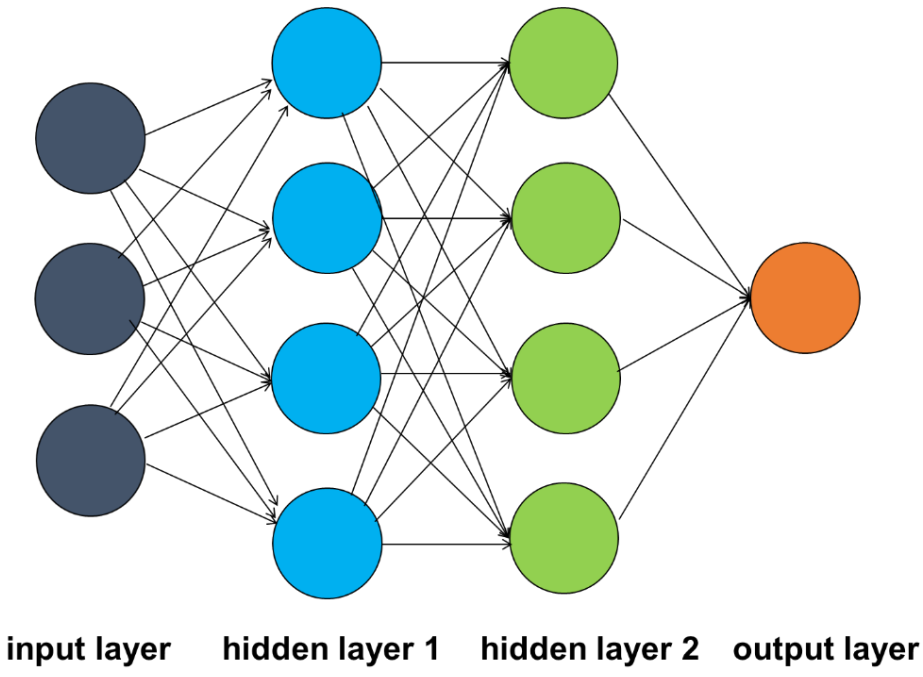

Figure 2.2: Multi-layer Feed Forward ANN

2.2 Architecture

The architecture of an ANN can be defined by the interconnections of its nodes. The most basic types of neuron connection architecture are the Single-layer feed-forward network and the Multi-layer feed-forward network.

2.2.1 Single-Layer Networks

A single-layer network - or a perceptron - consists of one node which receives the input, and outputs the result directly [17]. Perceptron was initially presented by Rosenblatt in 1957 [18]. It functions as described above and according to [19], its output for an input vector $x = (x_1, x_2, x_n)$ is calculated by an activation function g:

$$y = g\left(\sum_{i=0}^{n} w_i x_i\right) \tag{5}$$

A perceptron also includes a dummy input $x_0 = 1$ with a relative weight w_o called bias:

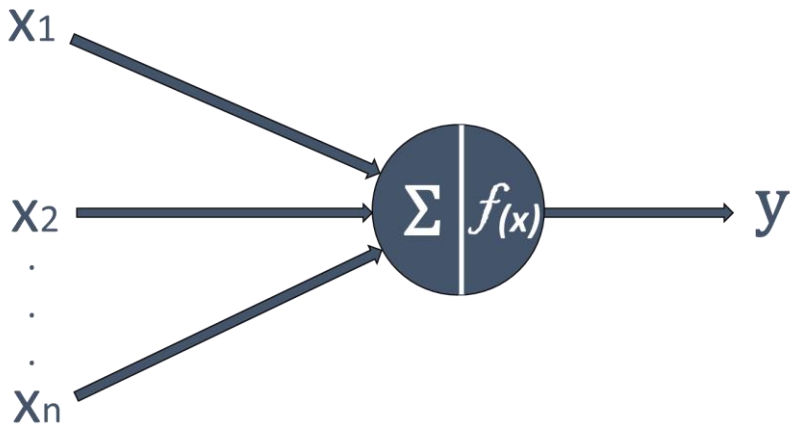

Figure 2.3: Perceptron

2.2.2 Multi-Layer Networks (MLNs)

Multi-Layer Networks (MLN) consist of at least 3 layers, which means they contain at least one hidden layer between the input and the output layers. Their nodes may be fully-connected (dense) or partially-connected. Fully-connected layers have all their nodes connected with all nodes of the next layer, whereas in partially-connected layers, not all nodes are connected with all the next layer's nodes. A simple Multi-Layer Network can be seen in Figure 2.2.

2.3 Variants of Deep Networks

There are numerous variants of neural networks with different architecture and applications. Some of the most popular Deep Networks are being described below.

2.3.1 Feed-Forward Neural Networks (FNNs)

A Feed-Forward Neural Network (FNN) is a multi-layer network that can be represented by a directed acyclic graph. All its edges point towards the same direction, meaning there is no node whose output is an input of another node belonging to the same or to a

previous layer. Therefore, this network does not have any cycles or loops and the information flows in one direction. Backpropagation Networks are also FNNs, with the difference being the direction of the information [20]. Figure 2.2 demonstrates an example of a Feed-Forward Network.

2.3.2 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are also called Feed Backward Neural Networks. As stated in [21], "they are feed forward neural networks, augmented by the inclusion of edges that span adjacent time steps, introducing a notion of time in the model". This means that their nodes are capable of passing information to any unit in any layer, forming cycles or self-connections. Its edges form a directed graph. An RNN memorizes all information ever calculated. This means that every single output takes under consideration, on the one hand its input and on the other hand all outputs that have ever been calculated in the past. This can be achieved due to the reason that the final output is calculated recursively, by applying activation functions on previous layers [22].

The input of a node at time t consists of both the input activation of the current node $x_{(t)}$ and from the hidden nodes $h_{(t-1)}$ in the previous state. The output $\hat{y}_{(t)}$ is calculated given the hidden state $h_{(t)}$ at that time step. This allows the previous input ($x_{(t-1)}$) at time t-1 to influence the current output $y_{(t)}$ at time t.

According to [21], all calculations necessary for computation at each time are shown in the following two equations:

$$h^{(t)} = \sigma\left(W^{hx}x^{(t)} + W^{hh}h^{(t-1)} + b_h\right) \quad (6)$$

$$\hat{y}^{(t)} = \mathrm{softmax}\left(W^{yh}h^{(t)} + b_y\right) \quad (7)$$

, where W_{hz} is the matrix of weights between input and hidden layers, W_{hh} is the matrix of recurrent weights between hidden layers at adjacent time steps and b_h, b_y are biases which allow each node to learn an offset.

An RNN can be represented in two ways. The first one is shown in Figure 2.4 where the timesteps are not shown distinctively and the second one in Figure 2.5 where each node corresponds to a different time step.

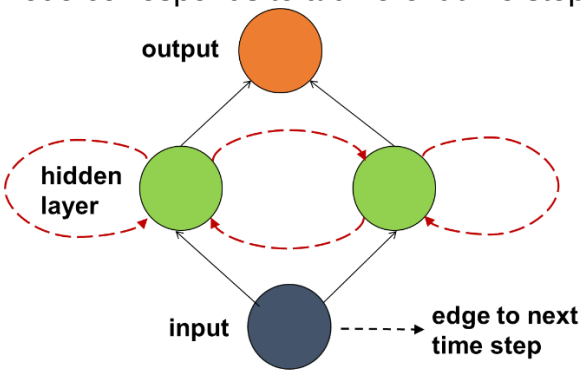

Figure 2.4: Recurrent ANN

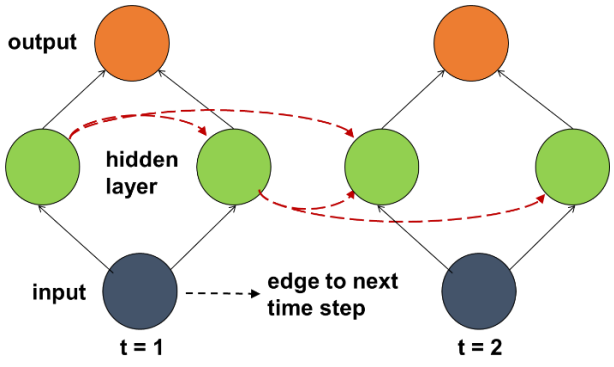

Figure 2.5: Unfolded Recurrent ANN

2.3.2.1 Recurrent Neural Network Variations

There are many variations of RNNs. The most significant ones are described below.

2.3.2.1.1 Long Short-Term Memory (LSTMs)

Long Short-Term Memory networks (LSTMs) are RNN variations created by Hochreiter and Schmidhuber in 1997 [23] and have the ability to overcome the problem of vanishing gradients. Their hidden nodes are represented by memory cells that memorize information like in a simple RNN.

An LSTM cell consists of an input gate, a forget gate and an internal state [24], as shown in Figure 2.6. The input gate controls the flow of information the cell receives in its input. If it is closed, therefore produces the value of 0, the cell receives no information at all. However, if it produces 1, the cell receives all information. It basically decides how much the new state $h[t]$ will be updated with a new candidate state $\bar{h}[t]$. The output gate controls the flow of information in the output of the cell, selecting the part of the state to be returned as the output $y[t]$ of the cell. The forget gate controls the incoming information from previous states, deciding which part of the information should be discarded from the previous state $h[t-1]$. Forget gates were not included in the model from the beginning. They were introduced for the first time in 2000 [25] and their characteristic is their ability to reset the internal state. The internal state represents a node that has a recurrent edge to itself, with a weight of 1. This edge is called Constant Error Carousel (CEC) and keeps the error flowing across timesteps without vanishing or exploding [21]. All gates depend on the current external input $x[t]$ and the output of previous cells $y[t-1]$.

When a state is updated, the input gate decides how much the new state will be updated with a new candidate state. Then, through an activation function (usually Hyperbolic Tangent - tanh), a vector C'_t with candidate values is generated and supplied to the current state. The new state C_t is calculated by multiplying the previous state C_{t-1} with the forget gate σ_f. Finally, the output gate is activated through a sigmoid function which decides which part of the cell state will be the final output.

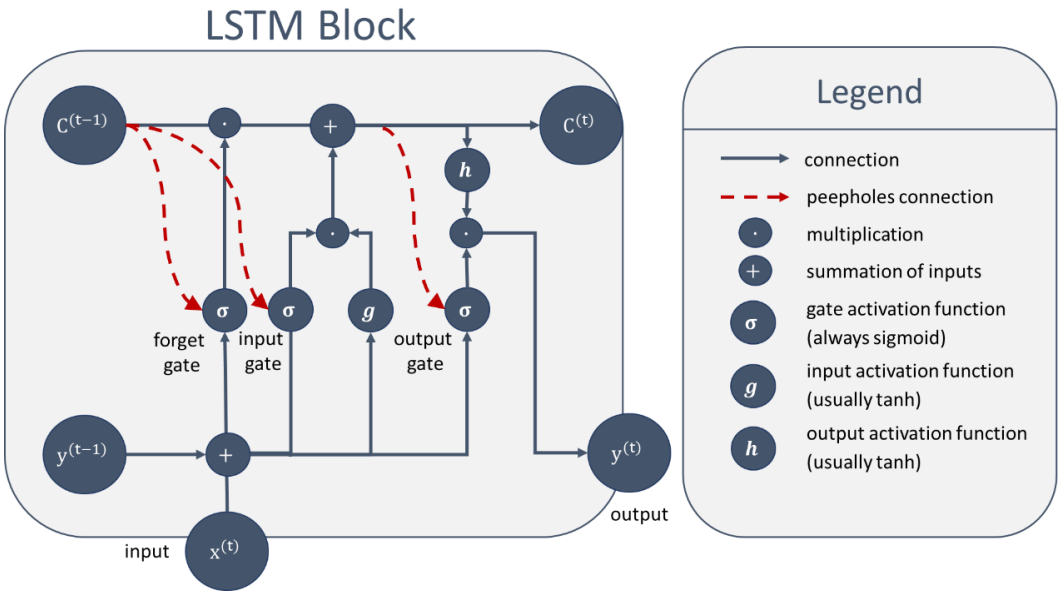

Figure 2.6: LSTM cell

According to [22], the equations defining the output of each state are the following:

$$forget\ gate: \sigma_f[t] = \sigma\big(W_f x[t] + R_f y[t-1] + b_f\big) \quad (8)$$

$$update\ gate: \sigma_u[t] = \sigma(W_u x[t] + R_u y[t-1] + b_u) \quad (9)$$

$$output\ gate: \sigma_o[t] = \sigma(W_o x[t] + R_o y[t-1] + b_o) \quad (10)$$

$$candidate\ state: \bar{h}[t] = g_1(W_h x[t] + R_h y[t-1] + b_h) \quad (11)$$

$$cell\ state: h = \sigma_u[t] \odot \mathrm{h}[t] + \sigma_f[t] \odot \mathrm{h}[t-1] \quad (12)$$

$$output: y[t] = \sigma_o[t] \odot \mathrm{g}_2(\mathrm{h}[t]) \quad (13)$$

, where $\odot$ symbolizes the element-wise multiplication, $x[t]$ is the input for each time step t, $W_f,\ W_u, W_o,\ W_h$ are the weights applied in each gate, $R_f,\ R_u, R_o,\ R_h$ are matrices that define the weights of the recurrent connections and $b_f,\ b_u, b_o,\ b_h$ are the biases in each state. Function σ represents a sigmoid function and g_1, g_2 are nonlinear activation functions (usually tanh).

2.3.2.1.2 Gated Recurrent Units (GRUs)

Gated Recurrent Units (GRUs) were introduced in 2014 by Cho et al. [26], who were inspired by the LSTMs. They contain memory cells as well.

A GRU cell consists of an update gate, which controls how much the unit updates its activation or content. On the other hand, a reset gate has the ability to reset the cell's memory. In this cell, the candidate state determines the information that needs to be removed from previous time steps.
In order to calculate the final memory value of the current unit h_t ht, the information needed from h'_t and from the previous value h_{t-1} must be determined by the update gate values z_t.

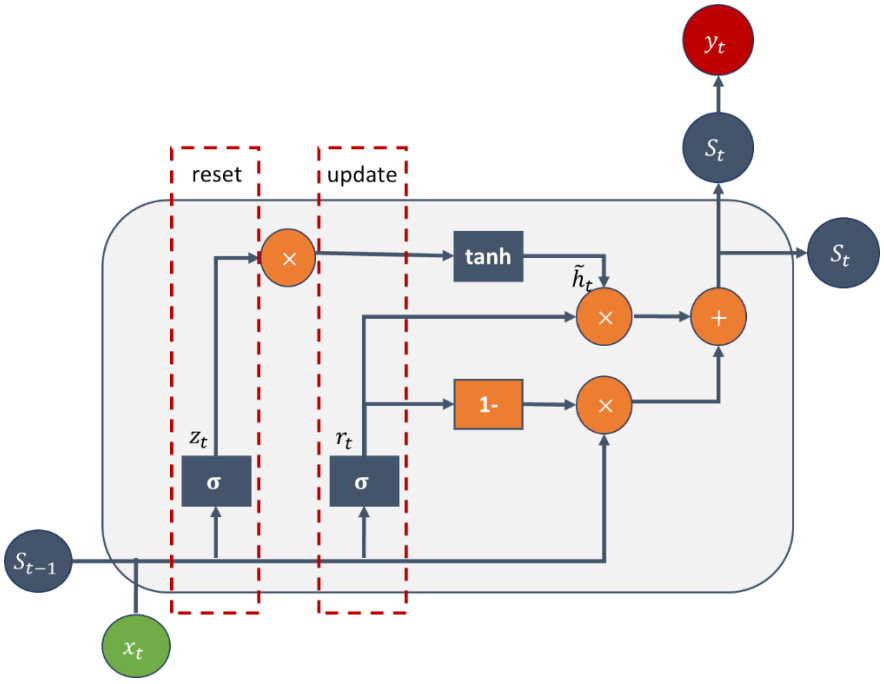

Figure 2.7: GRU cell

In accordance with [22], the equations of its states are the following:

$$reset\ gate: r[t] = \sigma(W_r h[t-1] + R_r x[t] + b_r) \quad (14)$$

$$update\ gate: u[t] = \sigma(W_u h[t-1] + R_u x[t] + b_u) \quad (15)$$

$$current\ state: h'[t] = h[t-1] \odot \mathrm{r}[t] \quad (16)$$

$$candidate\ state: \bar{z}[t] = g\left(W_z h'^{[t]} + R_z x[t] + b_z\right) \quad (17)$$

$$new\ state: h[t] = (1 - u[t]) \odot \mathrm{h}[t-1] + u[t] \odot \mathrm{h}[t-1] \quad (18)$$

, where $\odot$ symbolizes the element-wise multiplication, x[t] is the input for each time step t, W_r, W_u, W_z are the weights applied in each gate, R_r, R_u, R_z are matrices that define the weights of the recurrent connections and b_r, b_u, b_z are the biases in each state. Function σ represents a sigmoid function and g a nonlinear activation function, which usually is tanh.

2.3.2.1.3 Bidirectional Recurrent Neural Networks (BRNNs)

Bidirectional Recurrent Neural Networks (BRNNs) were firstly introduced in 1997 by Schuster and Paliwal [27]. They are described with two hidden layers that are both connected to input and output. The first layer sends information forward through time, while the second layer sends information backwards through time. This allows the neural network to have access to information from next time steps besides from past ones.

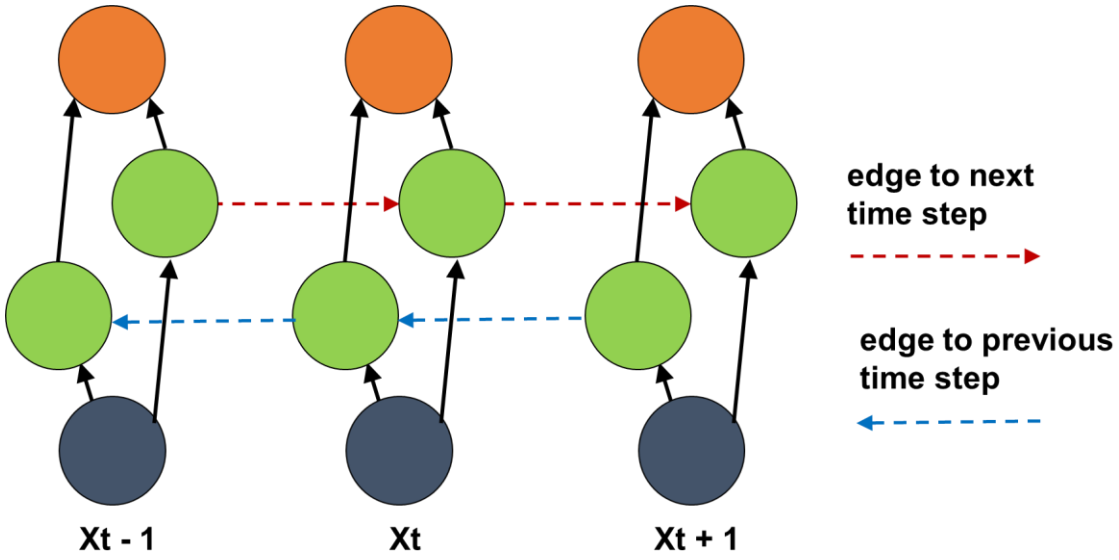

Figure 2.8: Bidirectional Neural Networks

According to [22], the equations of a BRNN are the following:

$$h_f^{(t)} = \sigma\left(W_{h_f x}\, x^{(t)} + W_{h_f h_f}\, h_f^{(t-1)} + b_{h_f}\right) \quad (19)$$

$$h_b^{(t)} = \sigma\left(W_{h_b x}\, x^{(t)} + W_{h_b h_b}\, h_b^{(t-1)} + b_{h_b}\right) \quad (20)$$

$$\hat{y}^{(t)} = softmax\left(W_{y h_f} h_f^{(t)} + W_{y h_b} h_b^{(t)} + b_y\right), \quad (21)$$

where $h_f^{(t)}, h_b^{(t)}$ correspond to the forward and backward directions, respectively.

One limitation of BRNNs is the fact that they cannot run continuously, because they require a fixed endpoint.

2.4 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) simulate the visual cortex of the brain and are based on the neocognitron [28]. They are among the most popular deep networks due to their outstanding performance with visual data and have been used in Image Recognition since 1980. Recently, they are being used in additional applications, such as in Image Classification [29], Classification of other Data [30], Computer Vision [31] and Natural Language Processing (NLP) [32]. An example of a CNN is illustrated in Figure 2.9:

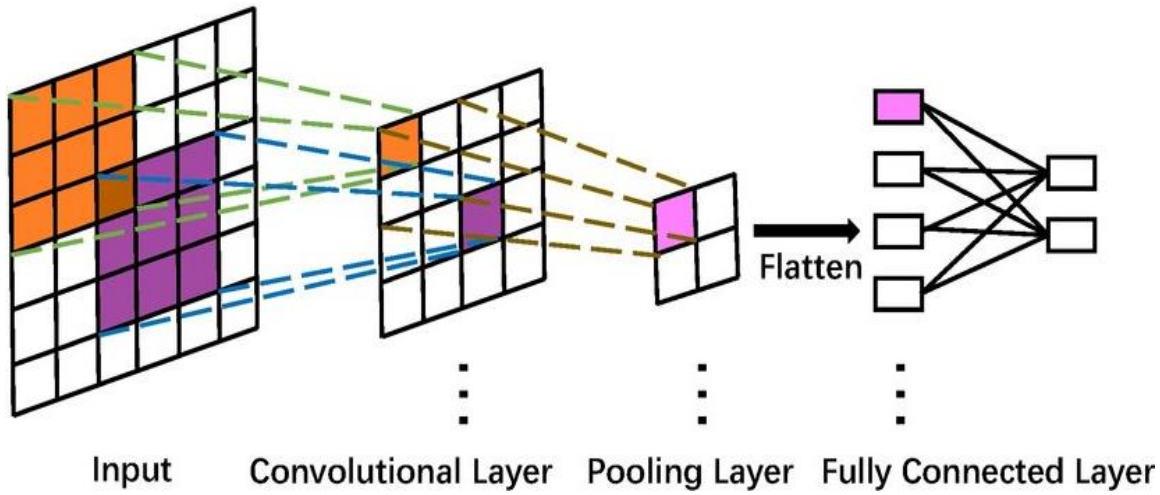

Figure 2.9: Convolutional Neural Network *[33]*

2.4.1 Convolution

Convolution is a mathematical commutative operation, applied in waveforms as a filtering mechanism. According to [34], convolution between two functions f, g is defined as:

$$(f * g)(x) = \int_{-\infty}^{+\infty} f(a)g(x-a)da \tag{22}$$

CNNs frequently manage two-dimensional data (2D). According to [35], for two-dimensional functions equation (22) is transformed as follows:

$$(f * g)(x,y) = \int_{-\infty}^{+\infty} f(a,\beta) \times g(x-a, y-\beta)\, da\, d\beta \tag{23}$$

Convolution in CNNs is conducted repeatedly and is used to extract the position and the importance of a feature. It occurs between two multidimensional matrices A and B, where A represents the input data and B the weight matrix. In CNNs, the weights are called filters or kernels. The outcome of the convolution is another 2D matrix (C), which consists of the dot product of A and B. For matrices A and B, with equal dimensions d_1, d_2, outcome C can be calculated as follows:

$$C = \sum_{i=0}^{d_1-1} \sum_{j=0}^{d_2-1} a_{i,j}\, b_{i,j} \tag{24}$$

2.4.2 Stride

Stride is a very important feature in convoluting CNNs. It defines the number of pixels the filters are shifted, each time that the next element of the output is calculated.
According to [35], for stride s, the dimension of the output can be calculated as follows:

$$\frac{n + 22p - f}{s + 1} \tag{25}$$

2.4.3 Padding

Padding is the procedure during which, information missing at the edges of the input is filled. It actually adds layers of zeros, or any other specified number, to the edges of the input images. This prevents the image of shrinking too much during convolution and preserves the information of the pixels at the border of the images.

2.4.4 Convolutional Neural Network Architecture

A Convolutional Neural Network (CNN) is a Feed-Forward Network which includes three additional types of layers. These are the Convolutional, Pooling and Dense layers. It can be seen in Figure 2.9.

2.4.4.1 Convolutional Layer

The Convolutional Layer is the most important component in a CNN and where the Convolution actually takes place. Unlike simple layers, Convolutional layers don't receive the whole input, but only receive partial input that belongs in the rectangular parts. These rectangles are the neuron's receptive fields, and this can be seen in the figure below:

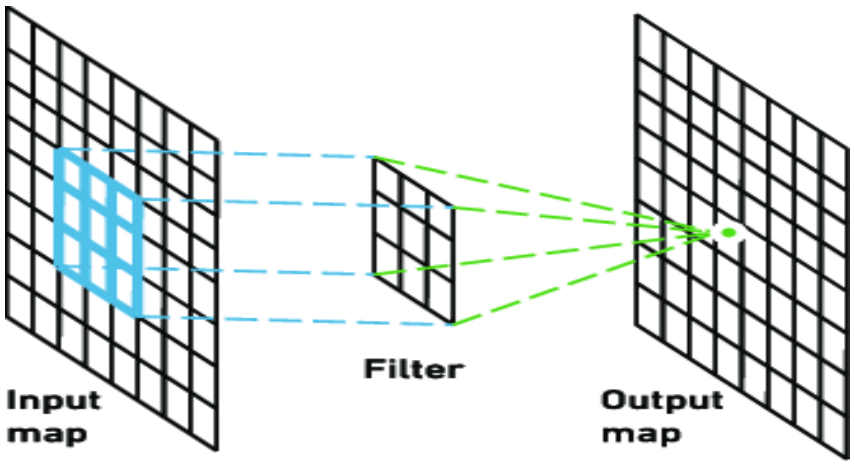

Figure 2.10: Convolutional Layer Function *[36]*

2.4.4.2 Pooling Layer

Pooling Layers are usually stacked between two Convolutional layers. They reduce the number of parameters calculated, by down-sampling the representation. This is called Spatial Pooling (or sub-sampling) and is used to reduce the information's dimensions while preserving important information in the rectified feature map.

Pooling layers don't include weights (filters). Depending on the Pooling method used, they preserve different features of their input. There are three Spatial Pooling operations:

- Max Pooling: is the pooling operation that selects the larger element of the rectified feature map.
- Average Pooling: is the pooling operation that calculates and keeps the average of the rectified feature map elements.
- Sum Pooling: is the pooling operation that calculates and keeps the sum of the rectified feature map elements.

An example of Max Pooling is illustrated in the Figure below:

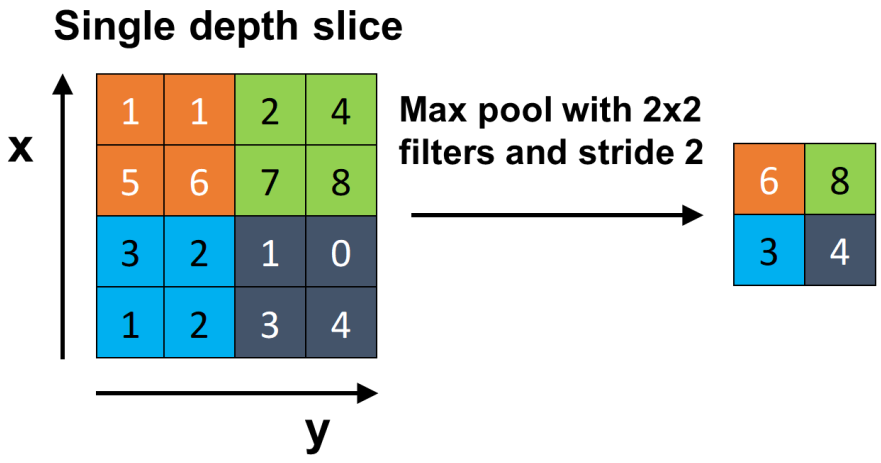

Figure 2.11: Max Pooling Example

2.4.4.3 Fully-Connected Layer:

A Fully-Connected Layer (or Dense) has its nodes connected to the nodes of the next layer. It has already been analyzed in section 2.2.2. This can be seen on the Figure below:

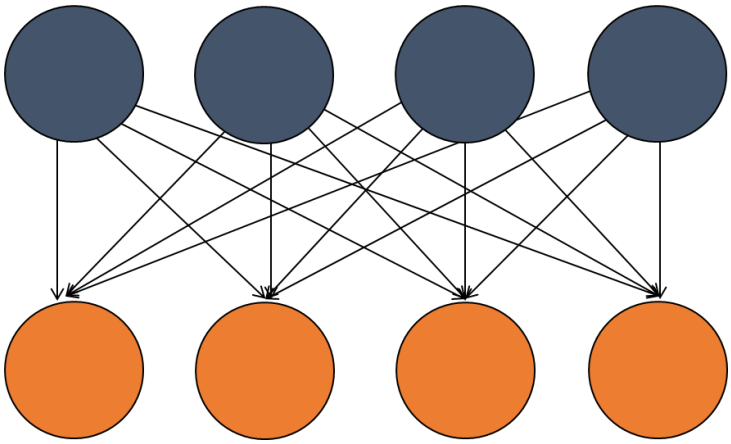

Figure 2.12: Fully-Connected Layer

2.5 More Artificial Neural Network (ANN) Components

Neural networks have more key components in order to function properly. These components are the training the network requires prior to its usage, the cost function that is used for its validation, the tuning of its hyperparameters and the choice of the activation functions for its neurons. All of these are described more extensively in the following sections.

2.5.1 Training

Training is essential in order for an ANN to produce accurate results. The first step is to input the training dataset to the Network. This dataset should include examples of the information needed to be processed. Then, the output needs to be evaluated, in comparison with the label set containing the true results of the training set's examples. For this comparison to take place, a cost function is being used and minimized for each and every input. The purpose of the training is to tweak the weights and biases in order to achieve the optimum results [37].

There are three different methods of training:

- Supervised Learning: In Supervised Learning, the model receives both the training set and the label set. Based on these sets, it creates a function that correlates them. This function is generalized and should eventually produce accurate results for inputs whose outputs are unknown. It has various applications, mainly in Classification [38] and in Prediction [39].

- Unsupervised Learning: In Unsupervised Learning, the model is adapted to the input data through observations and associations, without knowledge of the desired outputs. The label set is only used for the evaluation and the model is unaware of the true values of the training set. It is used in many applications, such as Clustering problems [40].

- Reinforcement Learning: Reinforcement Learning is the method that is most related to the theory of AI. The problem is defined through a model-agent and this model is trained through its interactions with the environment. It has multiple applications, such as in Games (board games, strategic games etc.) [41].

2.5.2 Cost Functions

Cost Functions (or Loss Functions) are used to evaluate the output of an ANN. They measure the deviation between the network's outputs versus the desired outputs. A cost function represents a real number that rates the performance of the Neural Network.

There are various cost functions that can be used. The decision of which one to use depends on the function that the network performs, or more specifically if they are used for regression or classification. In the following paragraphs the main types of Cost Functions will be analyzed.

2.5.2.1 Regression Cost Functions

Regression cost functions are used for regression models. These models aim to predict a numeric value for each of its inputs. The most commonly used regression cost functions are the following:

- Mean Squared Error (MSE) Loss - also known as L2 loss - is equal to the average of the sum of the squared differences between the model's predictions and the true values of the inputs. It is calculated as follows:

$$L(y, y') = \frac{\sum_{i=0}^{n}(y - y')^2}{n} \tag{25}$$

  , where y is the desired result (output), y' is the predicted result (output) and n is the number of values of the model's output.

- Mean Absolute Error (MAE) Loss - also known as L1 loss - is equal to the average of the sum of the absolute differences between the model's predictions and the true values of the inputs. It is calculated as follows:

$$L(y, y') = \frac{\sum_{i=0}^{n}|y - y'|}{n} \tag{26}$$

, where y is the desired result (output), y' is the predicted result (output) and n is the number of values of the model's output.

2.5.2.2 Multi-Class Classification Cost Functions

Multi-Class Classification Cost functions are used for classification models. These models intend to predict categorical variables for its inputs, where the output corresponds to one of multiple classes.

The most common Multi-Class Classification Cost function is the Multi-Class Cross-Entropy Loss (or Categorical Cross-Entropy Loss). In contrast with most Regression Cost Functions where the deviation between desired and actual output is used, the Multi-Class Cross-Entropy Loss function evaluates the outcome / classification, based on probabilities. The formula used is the following:

$$L(y, y') = \frac{1}{n} \sum_{i=1}^{n} y_i \, log y_i' \tag{27}$$

, where y_i' is a matrix containing the probabilities for the i-th value to belong to each class, y_i is a matrix with the probabilities corresponding to the target and n is the number of values of the model's output.

2.5.2.3 Binary Classification Cost Functions

Binary Classification Cost Functions are used for binary classification models. These models intend to predict categorical variables for its inputs with the characteristic that their output is restricted to two classes.

The most common Binary Classification Cost Function is the Binary Cross-Entropy Loss. It is equivalent to the Categorical Cross-Entropy Loss function, with only two possible classes. The target probabilities for the two classes are y_i and $(1 - y_i)$. The formula for this function is the following:

$$L(y, y') = \frac{1}{n} \sum_{i=1}^{n} y_i \, log y_i' + (1 - y_i) \log (1 - y_i') \tag{28}$$

, where y_i' is the i-th scalar value in the model output, y_i is the corresponding target value and n is the number of values of the model's output.

2.5.3 Hyperparameters

ANNs have learnable parameters that are adjusted during the learning process (like the weights and biases) and also non-learnable parameters called hyperparameters that need to be set prior to the learning process. Hyperparameters are very important features of an ANN. Their values significantly affect the performance of the model and tweaking them is required for getting the optimal results. It is possible to tune hyperparameters by hand, but there are also several optimizers used to automate the tuning process. The most commonly used ones are gradient descent and its variants [42], grid search and random search [43].

The core hyperparameters are the number of neurons in hidden layers, the number of the hidden layers, the activation functions and the cost functions. There are more hyperparameters used, such as the learning rate, the number of epochs, the batch size, etc.

2.5.4 Activation Functions

Activation Functions (AF) - or Transfer Functions - are used to calculate the weighted sum of the input and the biases that determine if a neuron will be activated or not. They have an important impact on the computational complexity of the network and particularly in the optimization of its parameters. Below, are described some of the most common activation functions.

All activation function graphs which are shown in this section (2.5.4) have been produced in Python and the relevant code is available here[1].

2.5.4.1 Sigmoid Functions

"A Sigmoid Function is a bounded and differentiable function that is nondecreasing and has exactly one inflection point" [44]. Sigmoid functions are mainly used in FNNs and the most common are the following: Logistic, Arctan, Tanh and Softsign.

2.5.4.1.1 Logistic Functions

It is a sigmoid function, whose derivatives are all positive. According to [45], the Logistic Function is given by the following equation:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{29}$$

The Logistic Activation graph is shown in Figure 2.13:

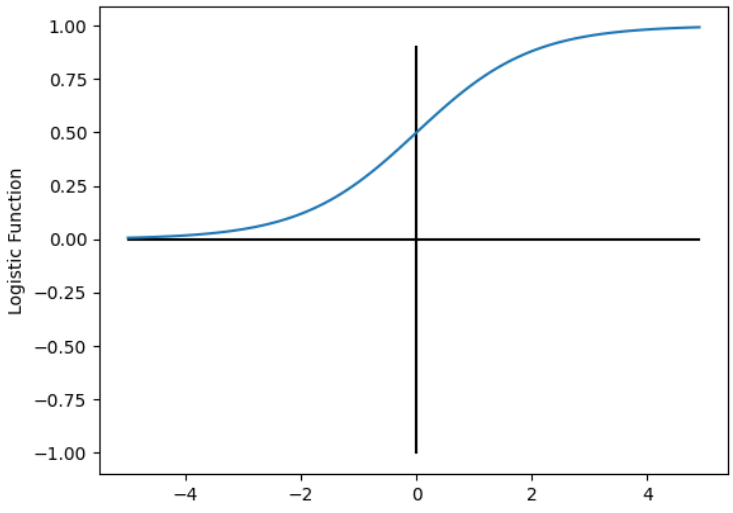

Figure 2.13: Logistic Activation Function

[1] https://github.com/mar-kan/thesis/tree/main/activation_functions_graphs

2.5.4.1.2 Hyperbolic Tangent Functions (TanH)

Hyperbolic Tangent function (Tanh) is a sigmoid function, whose range lies between -1 and +1 [-1,1]. According to [45], its equation is given by:

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{30}$$

The Tanh Activation graph is shown in Figure 2.14:

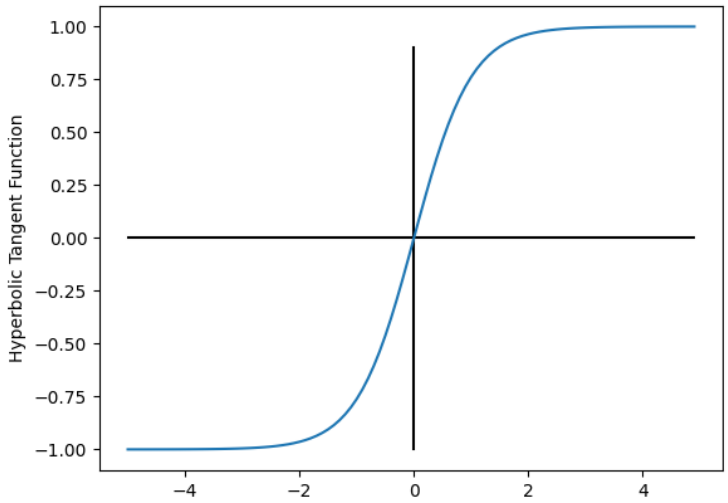

Figure 2.14: Tanh Activation Function

2.5.4.2 Rectified Linear Unit Functions (ReLU)

Rectified Linear Unit functions (ReLU) are linear functions and are also referred to as positive-part or ramp functions. For negative values, it is the constant function $y = 0$, while for positive values it is the identity function $y = x$ [45].
According to [44], its equation is:

$$f(x) = \max(0, x) = \begin{cases} x_i, x_i \geq 0 \\ 0, x_i < 0 \end{cases} \tag{31}$$

The ReLU graph is shown in Figure 2.15:

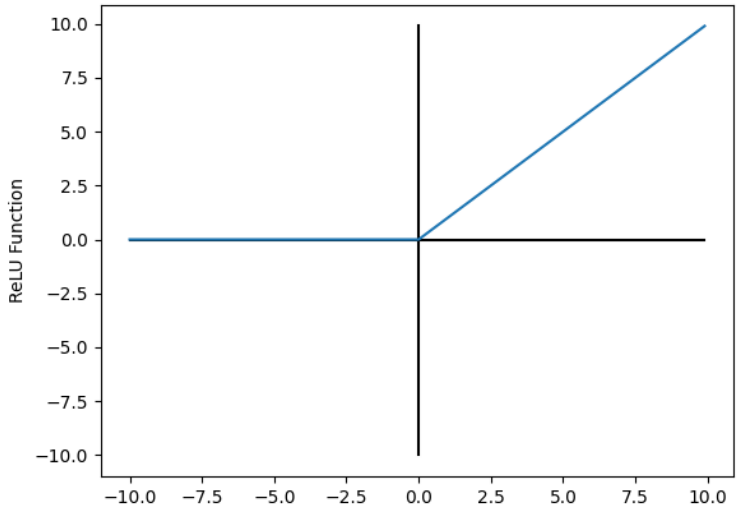

Figure 2.15: ReLU Activation Function

2.5.4.3 Leaky Rectified Linear Unit Functions (LReLU)

Leaky Rectified Linear Unit Functions (LReLU) functions were firstly introduced in 2013 [46]. A parameter α was included, to resolve the dead neuron problem of the ReLU functions. Parameter α's value lies between 0 and 1 $[\,0,1\,]$. If α equals to 0 ($a = 0$), LReLU works as the ReLU function. According to [45], it is calculated as follows:

$$f(x) = \max(ax, x) = \begin{cases} x, x > 0 \\ ax, x \leq 0 \end{cases} \tag{32}$$

The LReLU graph is shown in Figure 2.16:

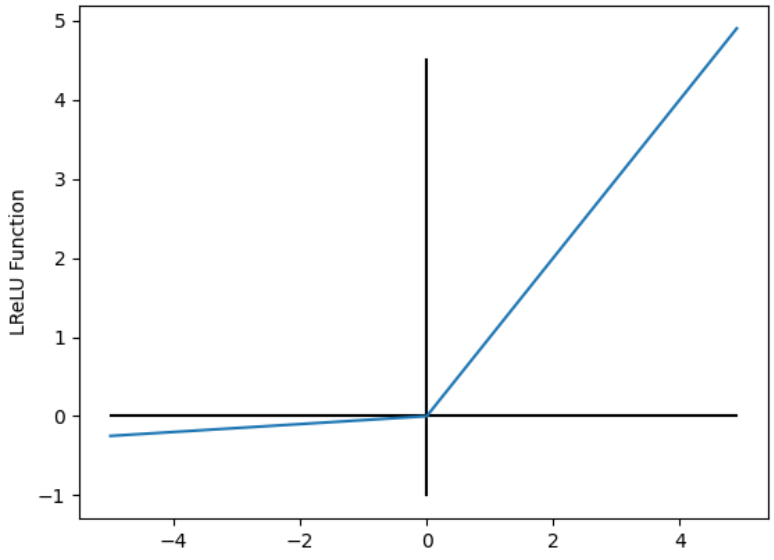

Figure 2.16: LReLU Activation Function

2.5.4.3 Exponential Linear Unit Functions (ELU)

Exponential Linear Units (ELU) were initially presented in 2015 [47]. They aim to accelerate the training process and solve the vanishing gradient problem by using only positive values. According to [44], ELU is calculated as follows:

$$f(x) = \begin{cases} x, x > 0 \\ a\,(e^x - 1), x \leq 0 \end{cases} \quad (33)$$

The ELU graph is shown in Figure 2.17:

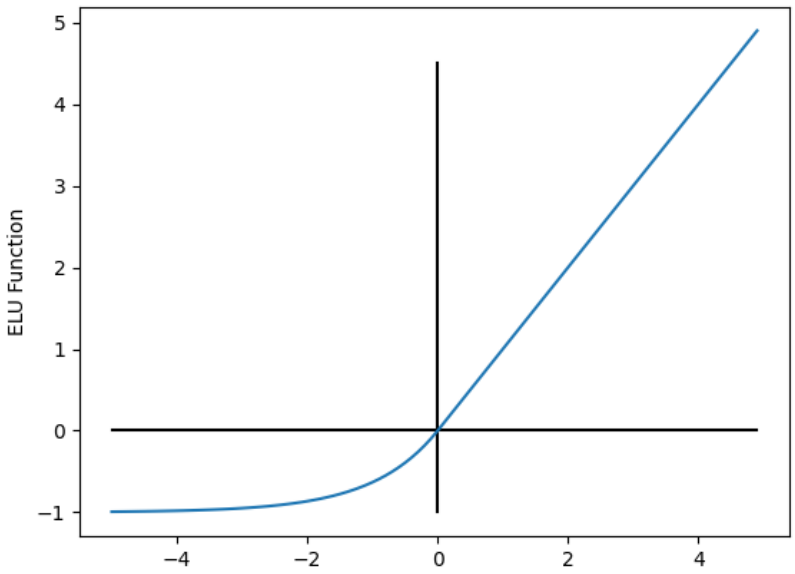

Figure 2.17: ELU Activation Function

2.5.4.4 Maxout Functions

Maxout Functions were presented in 2013 [48]. According to this, the neurons of the network inherit both the properties of LReLU and ReLU functions and overcome the problems of vanishing gradients (no dying neurons) and saturation. According to [45], the Maxout's equation is the following:

$$f(x) = \max(wT_1 x + b_1, wT_2 x + b_2) \quad (34)$$

, where w = weights, b = biases, T = transpose.

2.6 Artificial Neural Network (ANN) Issues

There are various issues that may occur while using ANNs that require to be resolved. The most important ones are the following:

2.6.1 Overfitting

Overfitting happens when a model doesn't generalize well from the training data to unknown data. The reason for this is that the model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data. This results in the model producing accurate results with the training dataset and poor results with the testing datasets.

There are various methods used to overcome overfitting, some of which are described below.

2.6.1.1 Early Stopping

This method stops the learning procedure at a previously set point, according to the theory that "the accuracy of algorithms stops improving after some point or even getting worse because of noise-learning, an idea dating back to 1970" [49]. When stopping the training, a suitable point should be used, because if the stopping occurs too early there will be underfitting (which will be analyzed in 2.5.2) and if it occurs too late the overfitting won't be resolved.

2.6.1.2 Network-Reduction

The reduction of the network's nodes is another method that can be used to resolve the problem of overfitting. It aims in reducing the complexity of the network by ignoring the least important data.

2.6.1.3 Regularization

The term regularization has many definitions. As stated in [50], "Regularization is any supplementary technique that aims at making the model generalize better, i.e. produce better results on the test set". Usually, regularization techniques are studied separately. There are many techniques, such as L1 Regularization, L2 Regularization and Dropout [49].

2.6.2 Underfitting

Underfitting is the opposite of overfitting. In this case, the model is too simplistic and proves unable to find the most important features of the input. This results in the model producing poor results for all the input datasets, including the training set.

The most common solution to alleviate overfitting is to increase the number of neurons. The problem might also take place when trying to resolve overfitting, in which case a modification on the solution used must occur.

3. DEEP LEARNING APPLICATIONS

Recently, Deep Learning is being used in a vast variety of scientific fields. ANNs can be used in any field without the researcher or scientist needing to have specialized knowledge of that specific field. The most important things needed are a validated set of data with its relative results, in order to do the proper evaluation. Therefore, Deep Learning has a variety of applications, the most important being the following:

3.1 Natural Language Processing and Generation

Natural Language Processing (NLP) deals with the understanding of the human language by the machine. The network analyzes and interprets the human language. In Natural Language Generation (NLG), it produces its own expressions. Initially, FNNs and CNNs where used for NLP. However, more recently RNNs produce better results in interpreting the meaning of idioms and expressions that might have different meaning depending on the context.

A major part of the Deep Learning papers published in NLP concern Sentiment Analysis (SA), where the model aims to identify the sentiment or opinion expressed in a text. SA is used for various purposes, such as the classification and review analysis in products, hotels, e-shops [51], in healthcare to evaluate the psychological health of a person [52], in economics [53], in extracting the topic of videos [54] and numerous other applications. In addition, NLP is widely used in Machine Translation [55], where neural networks are used to translate expressions from one human language to another. Paraphrase Identification analyzes different sentences or documents using NLP. Its applications include plagiarism detection [56] and document summarization [57]. Finally, NLP is used in Question Answering [58], where the model receives, interprets questions and answers appropriately. This technique is used in chatbots.

3.2 Processing Visual Data

Deep Learning has a wide variety of applications in processing visual data. The models used for this purpose are trying to mimic the human vision and achieve Computer vision. The most commonly used networks for analyzing visual information are CNNs and some of their variations, such as autoencoders [59].

Their main applications are Image Classification [60] - where images are classified into different categories, Object Detection and Image Segmentation [61] - where the model is trying to identify objects in images and Video Processing [62] - which does the same as the previous in videos, rather than in images. All the above are used extensively for numerous purposes, in several scientific fields.

3.3 Speech and Audio Processing

Deep Learning is also used in Speech and Audio Processing. The models used for this purpose receive and manipulate oral data and have multiple usages.

The main applications of Speech and Audio Processing are Speech Recognition [63], where the model tries to interpret the human language in oral form, Speech Emotion

Recognition [64], where the model tries to understand the emotion in oral human language, and Speech Enhancement - SE [65], where the model tries to enhance the audio and reduce the noise.

3.4 Other Applications

Deep Learning can also be used in several other fields. Some of the most important fields in which Deep Learning is used are Ecology [66], Economics [67], Information Retrieval [68] and Disaster Management Systems [69].

3.5 Healthcare

Healthcare is a very popular emergent field in Deep Learning and this paper is mainly focusing on this. DL models can discover “hidden patterns and other meaningful information from the considerable amount of health data that conventional analytics are not able to discover in a reasonable time” [70]. It has a variety of uses which can be divided in the two areas: Medical Imaging and Omics.

3.5.1 Medical Imaging

Medical Imaging or Medical Image Analysis is the main research area of Deep Learning in Biomedical applications [71]. It can analyze and interpret medical visual data for different purposes. Detection of lesion and/or abnormality is the major issue in medical image analysis [72]. Human organs are studied by analyzing images for the purpose of diagnosing or assessing various diseases. Several types of images can be used, depending on the model’s function, such as Ultrasound X-ray Radiography, Computed Tomography (CT), Magnetic Resonance imaging (MRI), Ultrasound and Digital Pathology [73].

The most common medical imaging applications in deep learning are the following [74]:

- Medical image detection [75], which is used to find the presence and location of objects that are appearing in the image.

- Medical image segmentation [76], [77] which according to [78] “refers to the process of extracting the desired object (organ) from a medical image (2D or 3D)”.

- Medical image classification, where medical images are classified as normal (healthy) and abnormal (non-healthy) and eventually this classification will result in diagnosing various medical issues, such as cancer [79], skin conditions [80] and other diseases that can be discovered in chest X-rays [80], [81].

- Medical image enhancement [82] is used to enhance the details of the image as much as possible by removing noise, sharpening, and/or increasing the density of pixels.

- Medical image registration [83], [84] is also referred to as image fusion or image matching. According to [85], it is “the process of aligning two or more images based on image appearances”.

3.5.2 Omics

The field of Omics, such as genomics, proteomics, or metabolomics [86] that is being researched in DL is divided in two areas: DNA and Proteins. Omics study DNA and RNA gene expression and sequencing, protein interactions with drugs and other proteins, and protein structure prediction. DL models that are used in Omics intend to "investigate and understand biological processes at a molecular level to predict and prevent diseases by involving patients in the development of more efficient and personalized treatment" [71].

3.5.2.1 DNA

DNA (deoxyribonucleic acid) is the genomic material in cells that contains the genetic information used in the development and functioning of all known living organisms [87]. It is a polymer of deoxyribo-nucleotides. Each nucleotide contains a pentose (5-carbon) sugar called deoxyribose, one of four nitrogenous bases (Thymine, Cytosine, Adenine and Guanine) and a phosphate group. The type of the nitrogenous base determines the type of nucleotide. A string of nucleotides that are joined together makes a DNA strand. Each DNA molecule contains two strands, which are twisted around one another to form a double helix.

Most of the DNA is located in the nucleus. However, some DNA can be found in mitochondria and chloroplasts. Within the nucleus, DNA is organized into structures called chromosomes. The complete set of chromosomes in a cell makes up its genome. Human genomes have approximately 3 billion DNA base pairs, which are arranged into 46 chromosomes. The information carried by the DNA is held in the sequence of pieces of DNA called genes.

RNA (ribonucleic acid) is also a polymer of nucleotides. RNAs are among a cell's key regulatory players, where they catalyze biochemical reactions and control and modulate gene expression [87]. An RNA nucleotide has the same main components as a DNA nucleotide. It contains a pentose sugar called ribose, one of four nitrogenous bases (Uracil instead of Thymine, Cytosine, Adenine and Guanine) and a phosphate group. A string of nucleotides that are joined together makes an RNA strand. Each RNA is single-stranded and forms a helix.

The DNA and RNA structure is illustrated in the following Figure:

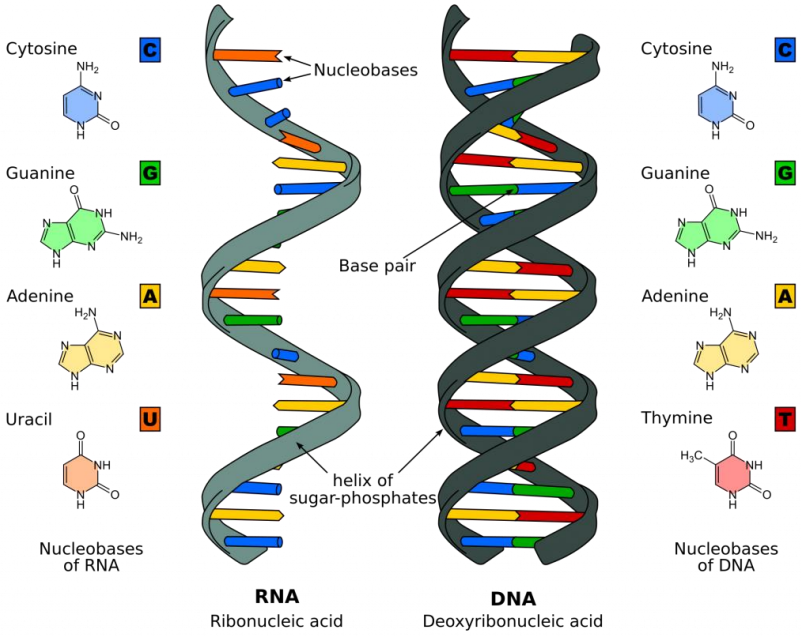

Figure 3.1: DNA and RNA structure *[88]*

3.5.2.2 DL Applications for Genomes

DL applications around the genome can be divided in three main areas, Protein Binding Prediction (PBP), Gene Expression and Genomic Sequencing.

3.5.2.2.1 Protein Binding Prediction

Protein Binding Prediction (PBP) [89] intends to predict the outcome of chemical interactions between a protein and DNA or RNA. These interactions are “essential for several molecular and cellular mechanisms, such as transcription, transcriptional regulation and DNA modifications, among others” [90]. Predicting the binding sites and finding their exact location is essential for the interpretation of the genome.

3.5.2.2.2 Gene Expression

Gene Expression is the process in which the information encoded in a gene is used to direct the assembly of a protein molecule. It is a multi-stage biological process that includes transcription, mRNA splicing, translation, and post-translational protein modification [91].

The initial step in Gene Expression is the transcription of the DNA molecule into an exact RNA copy, called precursor messenger RNA (pre-mRNA). Then, during RNA splicing, the pre-mRNA is modified into a new RNA molecule which is referred to as messenger RNA (mRNA), by removing the introns and joining the exons of the pre-mRNA. Introns and exons are nucleotide sequences within a gene. The mRNA molecule contains the exact same sequence of nucleotides as found in the DNA molecule (with Uracil substituted for Thymine). The translation process occurs in the ribosomes, which are the protein-making factories of the cell. So, the mRNA carries the transcribed message from the DNA to the ribosomes. There, the message is read three

letters (a codon) at a time, by a carrier molecule called transfer RNA (tRNA). Each codon corresponds to a particular amino acid. Each amino acid is attached specifically to its own tRNA molecule. When the mRNA sequence is read, each tRNA molecule delivers its amino acid to the ribosome and binds temporarily to the corresponding codon on the mRNA molecule. Once the tRNA is bound, it releases its amino acid, and the adjacent amino acids all join into a long chain called a polypeptide. This process continues until a protein is formed.

Deep learning applications focus on two different orientations in gene expression: the Alternative Splicing field and the Prediction of Gene Expression.

- Alternative Splicing (AS) [92] is a process whereby the exons of a primary transcript may be connected in different ways during pre-mRNA splicing. The objective is to build a model that can predict the outcome of AS from sequence information in different cellular contexts.
- Gene Expression prediction [93] aims to predicting the gene expression from histone modification. It can be formulated as a binary classification where the outputs represent the gene expression level (high or low) [71].

3.5.2.2.3 Genomic Sequencing

Genomic sequencing refers to sequencing the entire genome of an organism and is the process of determining the precise order of nucleotides within a DNA molecule [71]. It is very crucial in various applications such as basic Biological Research, Medical Diagnosis, Biotechnology, Forensic Biology, Virology, and Biological Systematics.

The applications of Deep Learning in genomic sequencing are mainly divided in two fields: learning the functional activity of DNA Sequencing and DNA Methylation.

- DNA Sequencing [94] is used in Deep Learning to map the whole genome for understanding its functional activity and identify possible genetic disorders or diseases.

- “DNA Methylation [95] is an epigenetic modification that is correlated with gene repression and is known to play an important role in gene regulation, development, and tumorigenesis.” [96]. It is a process by which methyl groups, which are organic functional groups with the formula CH3, are added to the DNA nucleotide and can modify the function of a DNA segment without changing the sequence. These methyl groups are projected into the major groove of DNA and are preventing transcription.

3.5.2.3 Proteins

A protein is a molecule that performs the necessary reactions for life sustainability of an organism [87]. Applications of Deep Learning (DL) about proteins are divided into two areas: Protein Structure Prediction (PSP) and Protein Interaction Prediction (PIP).

- Protein Structure Prediction [97], [98] is used to determine the three-dimensional shape of a protein based on its amino acid sequence. It is a problem of fundamental importance in biology, as the function of a protein is dependent on its structure [99].

- Protein Interaction Prediction studies the interactions between a protein and another protein, a protein and a drug and a protein and a compound element. Therefore, it is divided in three fields: Protein–Protein Interactions (PPIs), Drug-Target Interactions (DTIs) and Compound–Protein Interactions (CPIs) [71], which are analyzed as follows:

  - Protein–Protein Interactions [100], [101] are the physical contacts between two or more proteins and are crucial for protein function. These interactions play a critical role in many cellular biological processes. The detection and characterization of these interactions is essential towards understanding cellular biological processes in normal and disease states. [108]

  - Drug–Target Interaction [102] prediction is essential for drug research and development. It provides detailed descriptions of the biological activity, genomic features, and chemical structure to the disease treatment [71].

  - Compound–Protein Interaction [103] identification is crucial in the discovery and the development of safe and effective drugs. Revealing the effects of unknown compound–protein interactions is useful for predicting potential side effects of new drugs and finding new uses for existing drugs

4. METHODOLOGY

This section explains the research methodology used for this thesis. The RNA-sequencing data used, was downloaded, preprocessed and eventually transformed into two-dimensional (2D) images. These images were applied to a Convolutional Neural Network (CNN) in order to be classified into 33 types of cancer and the classification results were validated. The code developed and used for this thesis is available here[2].

The steps followed are extensively analyzed below:

4.1 Dataset

The data used in this paper was retrieved from The Cancer Genome Atlas (TCGA) Program [104], which is the Research Network of the National Cancer Institute's (NCI) Genomic Data Commons (GDC). Thirty-three (33) cancer types were downloaded along with their RNA-seq gene expression data. The TCGA data was downloaded using R programming language [105] for statistical computing and the "TCGAbiolinks" library [106], [107], [108]. This data consists of 9,285 samples and 19,947 genes which are presented in Table 4.1. The data was downloaded via a query that was formulated using the "TCGAbiolinks": GDCquery function for each cancer type. This function uses the GDC API for searching and downloading. The query used in this paper can be viewed in Figure 4.1, where <project> is the TCGA code of each cancer type. As far as the sample type is concerned, every available kind of primary cancer was selected.

```
query <- GDCquery(project = paste("TCGA-",project,sep=""),
                  data.category = "Gene expression",
                  data.type="Gene expression quantification",
                  experimental.strategy = "RNA-Seq",
                  platform = "Illumina HiSeq",
                  file.type = "results",
                  sample.type = c("Primary Tumor","Primary Blood Derived Cancer - Peripheral Blood",
                                  "Primary Blood Derived Cancer - Bone Marrow"),
                  legacy = TRUE
)
```

Figure 4.1: Query used for downloading the data

A dataset was exported for each cancer type, including the RNA-sequencing.

The R Packages used in this thesis are libraries MASS [109] and stringr [110] from R-base and libraries SummarizedExperiment [111] and TCGAbiolinks [106], [107], [108] from R/Bioconductor.

[2] https://github.com/mar-kan/thesis

Table 4.1: Tumor types and samples

TCGA-code	Cancer Type	Tumor Samples
ACC	Adrenocortical Carcinoma	79
BLCA	Bladder Urothelial Carcinoma	408
BRCA	Breast Invasive Carcinoma	1.095
CESC	Cervical Squamous Cell Carcinoma and Endocervical Adenocarcinoma	304
CHOL	Cholangiocarcinoma	36
COAD	Colon Adenocarcinoma	285
DLBC	Lymphoid Neoplasm Diffuse Large B-cell Lymphoma	48
ESCA	Esophageal Carcinoma	184
GBM	Glioblastoma Multiforme	156
HNSC	Head and Neck Squamous Cell Carcinoma	520
KICH	Kidney Chromophobe	66
KIRC	Kidney Renal Clear Cell Carcinoma	533
KIRP	Kidney Renal Papillary Cell Carcinoma	290
LAML	Acute Myeloid Leukemia	516
LGG	Brain Lower Grade Glioma	371
LIHC	Liver Hepatocellular Carcinoma	173
LUAD	Lung Adenocarcinoma	515
LUSC	Lung Squamous Cell Carcinoma	502
MESO	Mesothelioma	87
OV	Ovarian Serous Cystadenocarcinoma	304
PAAD	Pancreatic Adenocarcinoma	178
PCPG	Pheochromocytoma and Paraganglioma	179
PRAD	Prostate Adenocarcinoma	497
READ	Rectum Adenocarcinoma	94
SARC	Sarcoma	259
SKCM	Skin Cutaneous Melanoma	103
STAD	Stomach Adenocarcinoma	415
TGCT	Testicular Germ Cell Tumors	150
THCA	Thyroid Carcinoma	505
THYM	Thymoma	120
UCEC	Uterine Corpus Endometrial Carcinoma	176
UCS	Uterine Carcinosarcoma	57
UVM	Uveal Melanoma	80
	Total	**9.285**

4.2 Data Preparation

The datasets have been preprocessed in Python [112]. The genes were initially sorted based on their chromosome number, because adjacent genes are more likely to interact with each other [113]. Then, the gene samples were normalized and scaled, in order to attain the corresponding genes required for the sample without noise. This was achieved by using the following logarithmic transformation:

$$y = log_2(x + 1) \tag{35}$$

During the next step, all expression values of genes being less than one (<1), were set to zero (0), in order to further reduce the noise. Then, the genes whose expression levels remain almost unchanged across all samples, were filtered out, as they were most probably not associated with cancer. This was achieved by calculating the variance of each gene throughout all samples and using a threshold equal to 1.3, to filter them out, as they are probably unrelated with the cancer types. The threshold was chosen after experimentation, with an effort to achieve the highest possible accuracy of the classification. The number of genes filtered out was 8,902 from a total of 19,947, leaving 11,045 genes.

4.3 Transforming RNA sequences in two-dimensional (2D) images

The samples of RNA sequences were embedded into two-dimensional (2D) images, with the intention of inputting them in a Convolutional Neural Network. For this purpose, each sample was reshaped from an 11,045x1 array into a 106×106 image, by adding some zeros in the last line of the image. The dimension of 106 was selected, as it is the smallest number which squared produces a number larger than the total amount of genes used (11,045). Finally, all images were normalized in order to make sure that their values range between 0 and 255. Transforming the Gene Expression data into 2D images was inspired by the work of several relative papers [113], [114], [115]. An example of each cancer type of the generated images is shown in Figure 4.2:

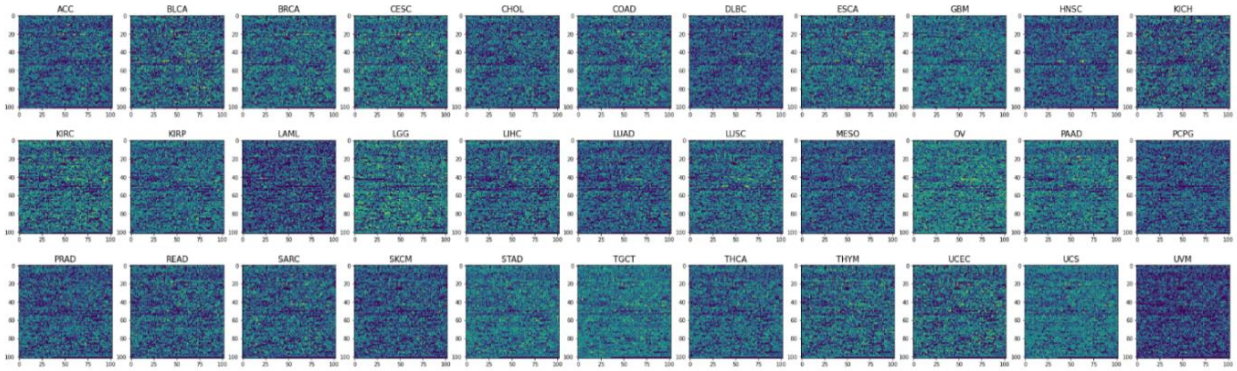

Figure 4.2: Example of embedded 2-D images

4.4 Classification

As far as the classification is concerned, a Convolutional Neural Network (CNN) was constructed, using Python [112], under the TensorFlow environment [116] and the Keras library [117]. The representation of the CNN architecture used, is depicted in Figure 4.3. It consists of three convolutional layers, each followed by a batch normalization and a max-pooling layer. Then, a flatten layer is used, to convert the multidimensional data into linear. Finally, there are three Dense layers, the latter one being the output layer. The CNN does not include many layers, as the existence of more layers caused overfitting in classes with a limited number of samples. An example of this, is the Colon Adenocarcinoma class (COAD) which only has 36 samples.

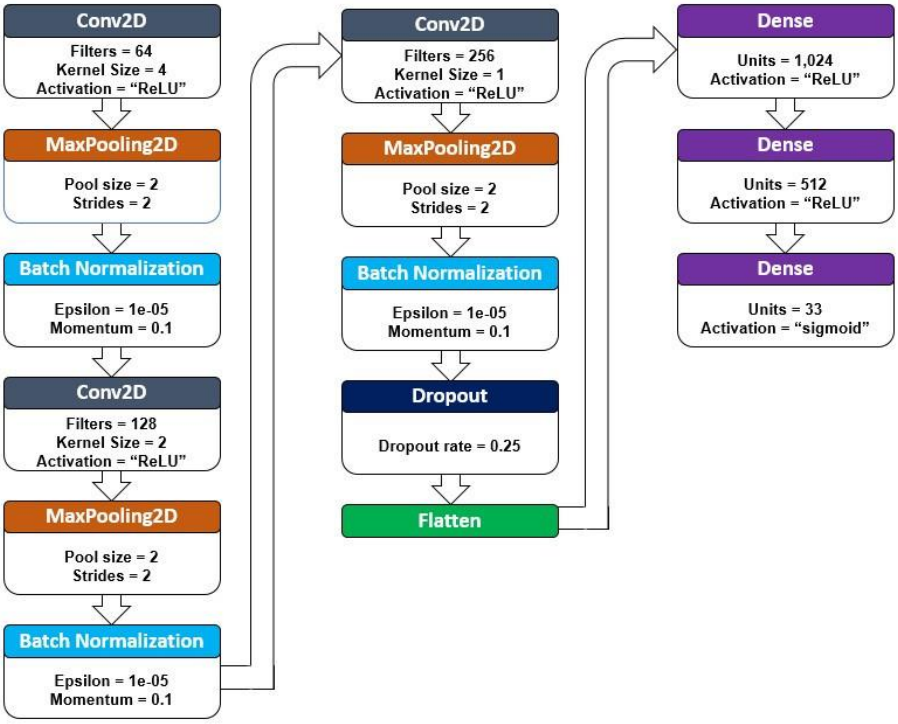

Figure 4.3: CNN Architecture

4.5 Hardware Specifications

The initial training procedures have been carried out on a personal computer with the following specifications: Intel i7 10700 CPU, 32GB RAM, 1TB SSD and Nvidia GTX 1660S (8GB) graphic card. However, the abovementioned setup proved to be marginally capable for the training as there were problems with the graphics card capacity. In order to make the system more stable and faster, further experiments were conducted in the Google Cloud Platform, using a Virtual Machine with the following specs: 1 vCPU, 3.75GB RAM, 100GB SPD and an Nvidia Tesla T4 GPU (16GB).

4.6 Training

The training was conducted using 15-fold cross validation, 200 epochs and a batch size of 128. The optimizer that was used, was Adamax with a learning rate of 0.0001 and the Categorical Cross Entropy loss function. Finally, each training session lasted approximately 8 hours in the personal computer and 7 hours in the virtual machine.

4.7 Evaluation

In this section, the evaluation metrics used are discussed briefly and the results of the model are presented.

4.7.1 Evaluation Metrics

A variety of metrics was used to evaluate the model. These metrics are Accuracy, Precision, Recall and F1 score.

4.7.1.1 Accuracy

Accuracy is defined as the ratio of the correctly predicted observations to the total observations. It is a helpful metric for the evaluation of a classification model when the dataset is balanced. However, it is not suitable for imbalanced datasets (that have different sample sizes for each class), because it does not take under consideration misclassifications. Accuracy is defined as follows:

$$\frac{True\ Negative + True\ Positive}{True\ Negative + False\ Positive + True\ Positive + False\ Negative} \tag{36}$$

4.7.1.2 Precision

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. It is also called positive predictive value and it is defined as follows:

$$\frac{True\ Positive}{True\ Negative + False\ Positive} \tag{37}$$

4.7.1.3 Recall

Recall is the ratio of correctly predicted positive observations to all observations in an actual class. It is also known as sensitivity and is defined as follows:

$$\frac{True\ Positive}{True\ Positive + False\ Negative} \tag{38}$$

4.7.1.4 F1 Score

F1 Score is the harmonic mean of Precision and Recall. It is another way of measuring the accuracy of a model. This metric takes both false positive and false negative results under consideration. It is also called F-measure and is defined as the weighted average of Precision and Recall:

$$2\ x\ \frac{Precision\ x\ Recall}{Precision + Recall} \tag{39}$$

4.7.1.5 Confusion matrix

A confusion matrix is a square matrix that can be used to summarize the results of a classification experiment. It visualizes the correlation between the predictions of the model and the true classes in which they belong to. Thus, it facilitates the detection of misclassification and confusion of classes.

4.7.2 Evaluation Results

The dataset used for the evaluation was 20% of the original dataset, a total of 1,857 samples. This 20% is selected randomly each time. Numerous experiments were conducted with different hyperparameters, in order to determine the optimal combination and achieve maximum accuracy. The accuracy metrics for each type of cancer respectively, are shown in Table 4.2.

Table 4.2: Accuracy Metrics for each cancer type

TCGA-code	Precision	Recall	F1-score	Support
ACC	1.000	1.000	1.000	15
BLCA	0.988	0.988	0.988	81
BRCA	1.000	1.000	1.000	214
CESC	0.983	0.983	0.983	58
CHOL	1.000	0.833	0.909	6
COAD	0.986	0.971	0.978	70
DLBC	1.000	1.000	1.000	10
ESCA	1.000	1.000	1.000	34
GBM	1.000	1.000	1.000	31
HNSC	1.000	1.000	1.000	106
KICH	1.000	1.000	1.000	16
KIRC	0.979	0.989	0.984	94
KIRP	1.000	0.955	0.977	44
LAML	1.000	1.000	1.000	41
LGG	1.000	1.000	1.000	104
LIHC	0.988	1.000	0.994	81
LUAD	0.981	0.981	0.981	106
LUSC	0.982	0.982	0.982	109
MESO	1.000	1.000	1.000	20
OV	1.000	1.000	1.000	65
PAAD	1.000	1.000	1.000	37
PCPG	1.000	1.000	1.000	35
PRAD	1.000	1.000	1.000	97
READ	0.923	0.960	0.941	25
SARC	1.000	1.000	1.000	54
SKCM	1.000	1.000	1.000	20
STAD	1.000	1.000	1.000	80
TGCT	1.000	1.000	1.000	31
THCA	1.000	1.000	1.000	91
THYM	1.000	1.000	1.000	24
UCEC	0.967	1.000	0.983	29
UCS	1.000	1.000	1.000	7
UVM	1.000	1.000	1.000	22

The total best average loss achieved (minimum) was 0.070 and the total maximum average accuracy was 0.993. The average accuracy metrics can be seen in Table 4.3:

Table 4.3: Average Accuracy Metrics

	Precision	Recall	F1-score	Support
Accuracy	0.993	0.993	0.993	0.993
Macro average	0.993	0.989	0.991	1,857
Weighted average	0.993	0.993	0.993	1,857

The accuracy of the classification results can also be seen in the following confusion matrix:

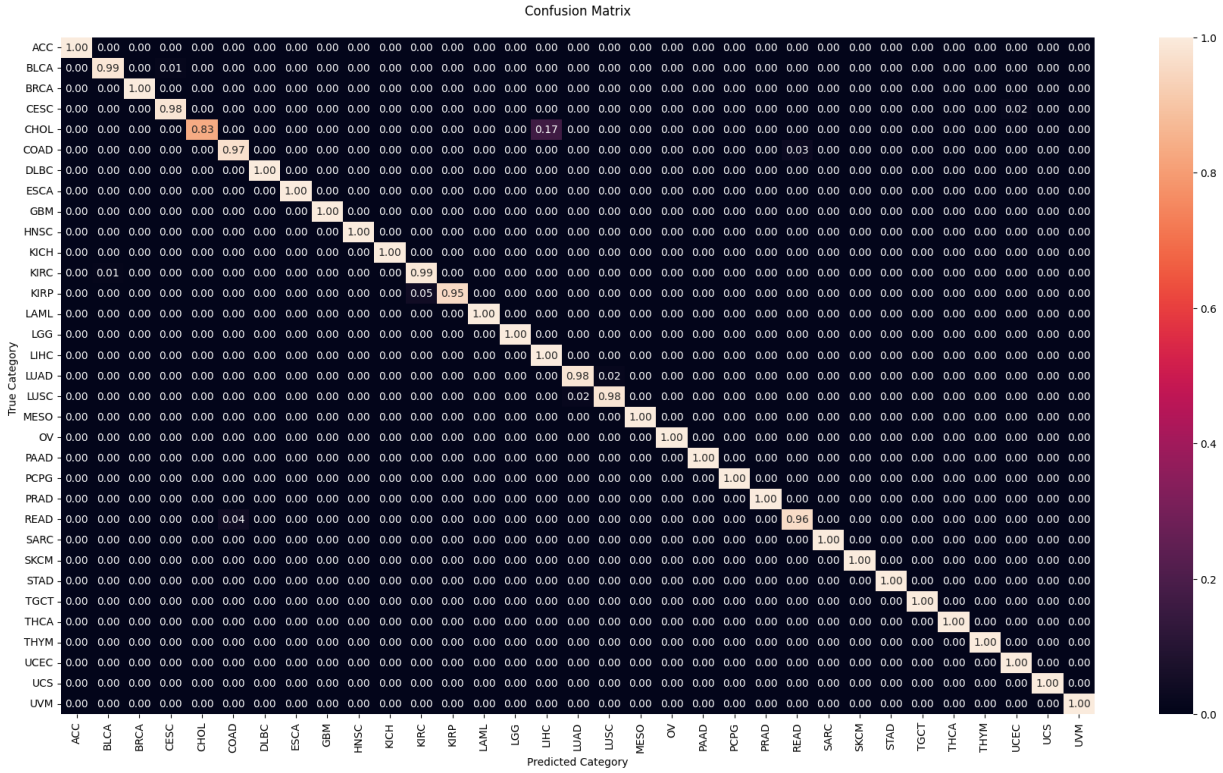

Figure 4.4: Confusion Matrix of classification

The model classifies the majority of the cancer types correctly. There is a modest percentage of samples from class Cholangiocarcinoma (CHOL) which is mislabeled. The reason for this, is that the number of samples is very small, as there are only 36 samples.

4.7.3 Graphs (Plots)

All validation sets of each fold of the training process were considered in the following 4 graphs, where the loss and accuracy of both training and validation sets are shown. The first two graphs (Figure 4.5, Figure 4.6) depict these values per epoch for each individual fold and the latter two graphs (Figure 4.7, Figure 4.8) depict the mean loss and mean accuracy per epoch for all folds.

From these graphs, we can see that, during training, the total best average loss achieved (minimum) was 0.010 and the total maximum average accuracy was 0.998. These numbers represent the averages of all validation sets of every fold.

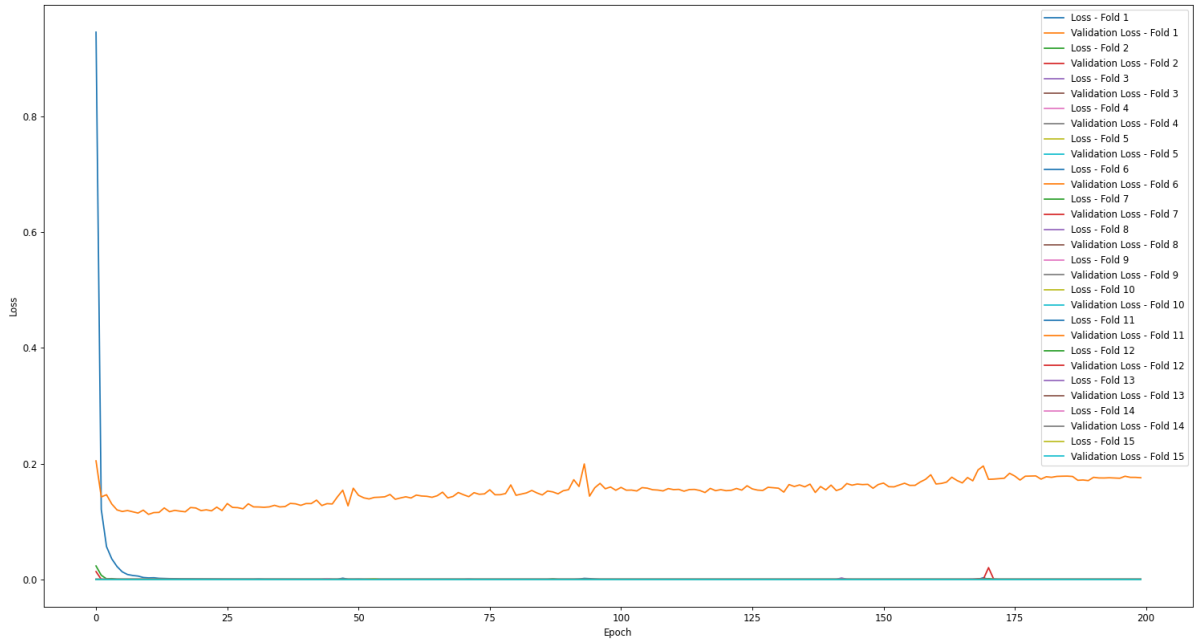

Figure 4.5: Loss per Epoch for all folds

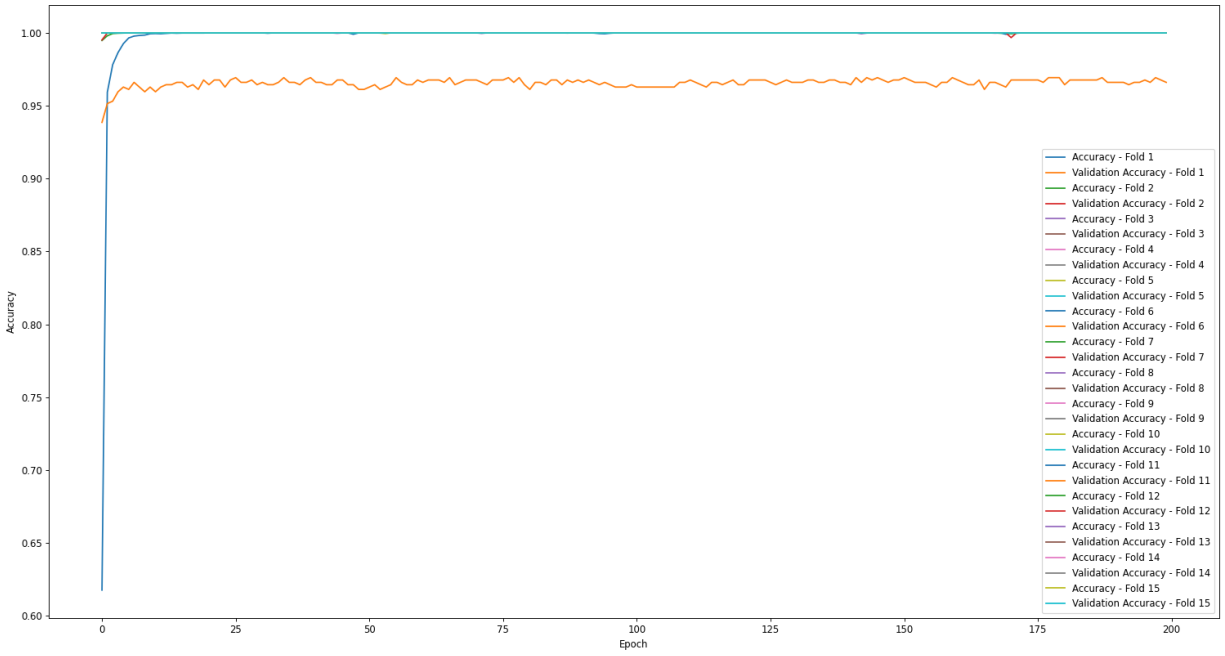

Figure 4.6: Accuracy per Epoch for all folds

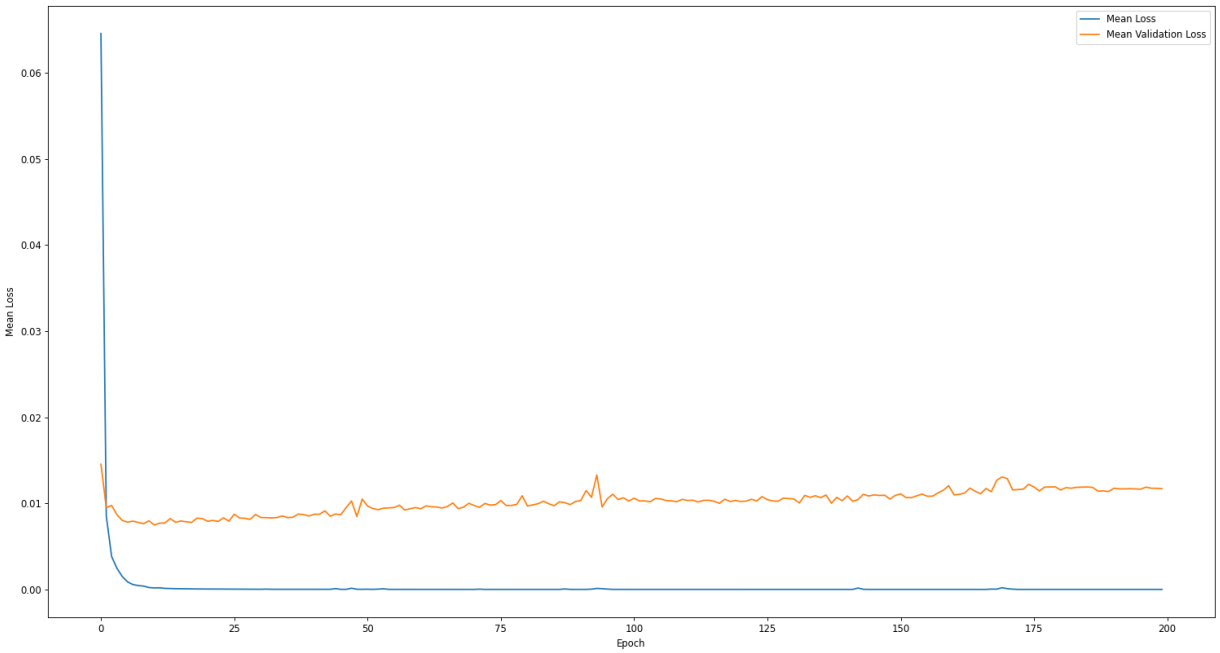

Figure 4.7: Average Loss per Epoch

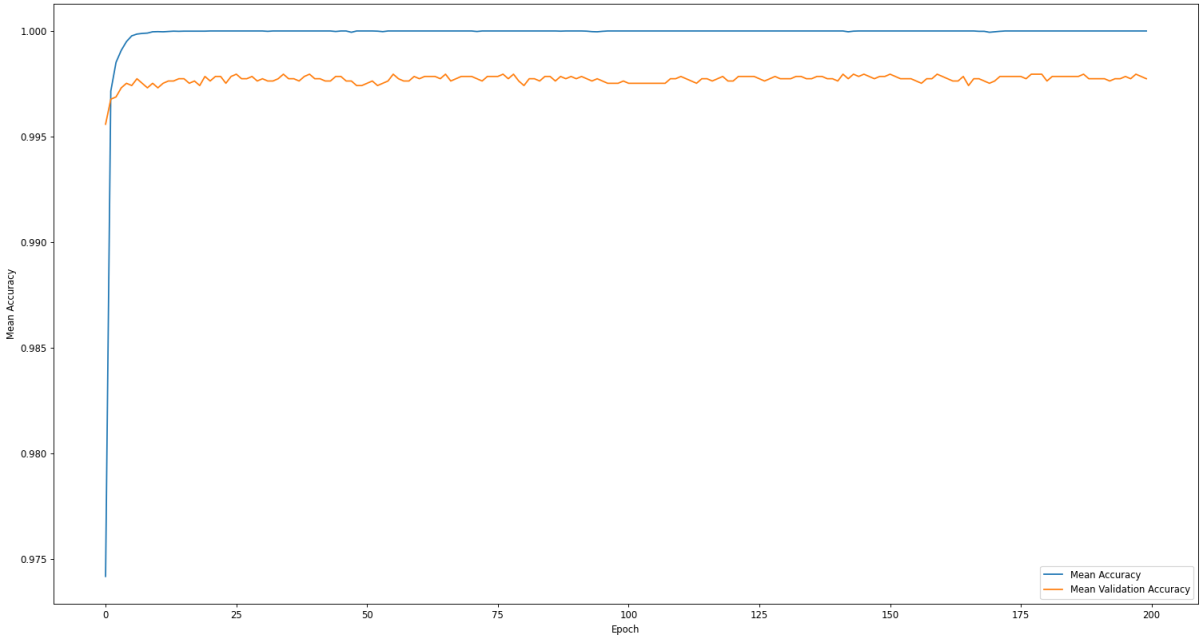

Figure 4.8: Average Accuracy per Epoch

5. CONCLUSIONS

5.1 Conclusions

The purpose of this thesis was to explore the possibility of using Deep Learning (DL) in order to serve the science of Biomedicine and particularly the diagnosis of cancer.

As stated in the World Health Organization (WHO): “Cancer is a leading cause of death worldwide, accounting for nearly 10 million deaths in 2020, nearly one in six deaths. Cancer mortality is reduced when cases are detected and treated early. A correct cancer diagnosis is essential for appropriate and effective treatment.”

Genomics and in particular Gene Expression data was used, consisting of RNA-sequencing. This data was preprocessed and transformed into multiple two-dimensional (2D) images. These images were then classified into 33 types of cancer with the application of a Convolutional Neural Network.

The result showed that cancer diagnosis for 33 different types of cancer, using the proposed method, has an accuracy of 0.993, or 99.3%. This means that 993 out of 1,000 cases will be diagnosed correctly, which shows an extremely high accuracy. This accuracy could have been increased with a more balanced dataset. More specifically, if the Cholangiocarcinoma (CHOL) would have more samples, or if it would be removed from the model.

Finally, a general comment is that this application is not panacea. Combined with the expertise from the medical community it can prove to be an excellent tool for correct cancer diagnosis.

5.2 Future Work

There are many things that could be applied in the future. The reason for not proceeding in implementing them was the lack of processing capacity and time.

It would be interesting to further experiment/proceed with a combination of a Convolutional and a Recurrent Neural Network, or with the conversion and usage of three-dimensional (3D) imaging. Furthermore, it would also be very challenging to apply this model in the prediction of cancer rather than the diagnosis, based on genomics. Finally, this method could also be applied for the diagnosis and prediction of several other genetic diseases.

ABBREVIATIONS – ACRONYMS

2D	Two-Dimensional
3D	Three-Dimensional
A	Adenine
AF	Activation Function
AI	Artificial Intelligence
ANN	Artificial Neural Networks
AS	Alternative Splicing
BRNN	Bidirectional Recurrent Neural Network
C	Cytosine
CEC	Constant Error Carousel
CNN	Convolutional Neural Networks
CPI	Compound-Protein Interaction
CT	Computed Tomography
DL	Deep Learning
DNA	Deoxyribonucleic Acid
DTI	Drug-Target Interaction
ELU	Exponential Linear Units
FNN	Feed-Forward Network
G	Guanine
GDC	Genomic Data Commons
GRU	Gated Recurrent Units
LReLU	Leaky Rectified Linear Unit
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
ML	Machine Learning
MLN	Multi-Layer Network
MRI	Magnetic Resonance Imaging

mRNA	Messenger Ribonucleic Acid
MSE	Mean Squared Error
NCI	National Cancer Institute
NLP	Natural Language Processing
PBP	Protein Binding Prediction
PIP	Protein Interaction Prediction
PPI	Protein-Protein Interaction
pre-mRNA	Precursor Messenger Ribonucleic Acid
PSP	Protein Structure Prediction
ReLU	Rectified Linear Unit
RNA	Ribonucleic Acid
RNA-seq	Ribonucleic Acid Sequencing
RNN	Recurrent ANN
SA	Sentiment Analysis
SE	Speech Enhancement
T	Thymine
TanH	Hyperbolic Tangent
TCGA	The Cancer Genome Atlas
tRNA	Transfer Ribonucleic Acid
U	Uracil

REFERENCES

[1] N. J. Nilsson, The Quest for Artificial Intelligence, Cambridge: Cambridge University Press, 2009.

[2] T. Mitchell, Machine Learning, International: McGraw-Hill International, 1997.

[3] L. Deng, "A tutorial survey of architectures, algorithms, and applications for deep learning," *APSIPA Transactions on Signal and Information Processing,* vol. 3, 2014.

[4] O. S. Eluyode and A. T. Dipo, "Comparative study of biological and artificial neural networks," *European Journal of Applied Engineering and Scientific Research,* vol. 2, no. 1, pp. 36-46, 2013.

[5] A. A. Soofi and A. Awan., "Classification techniques in machine learning: applications and issues," *Journal of Basic and Applied Sciences,* no. 13, pp. 459-465, 2017.

[6] A. Chowdhury and al., "Image driven machine learning methods for microstructure recognition," *Computational Materials Science,* no. 123, pp. 176-187, 2016.

[7] S. Agarwalla and K. K. Sarma., "Machine learning based sample extraction for automatic speech recognition using dialectal Assamese speech," *Neural Networks,* no. 78, pp. 97-111, 2016.

[8] F. Rundo and al., ""Machine learning for quantitative finance applications: A survey," *Applied Sciences,* vol. 9, no. 24, p. 5574, 2019.

[9] K. G. Liakos and al., "Machine learning in agriculture: A review," *Sensors,* vol. 18, no. 8, p. 2674, 2018.

[10] A. Dhillon and A. Singh, "Machine learning in healthcare data analysis: a survey," *Journal of Biology and Today's World,* vol. 8, no. 6, pp. 1-10, 2019.

[11] L. Kwo, "Contributed: Top 10 Use Cases for AI in Healthcare," *Mobi Health News,* vol. Global Edition, 2021.

[12] R. Shahane, Md Ismail and C. S. R. Prabhu., "A survey on deep learning techniques for prognosis and diagnosis of cancer from microarray gene expression data," *Journal of computational and theoretical Nanoscience,* vol. 16, no. 12, pp. 5078-5088, 2019.

[13] T. Saba, "Recent advancement in cancer detection using machine learning: Systematic survey of decades, comparisons and challenges," *Journal of Infection and Public Health,* vol. 13, no. 9, pp. 1274-1289, 2020.

[14] M. Zakaria, M. Al-Shebany and S. Sarhan, "Artificial neural network: a brief overview," *International Journal of Engineering Research and Applications,* vol. 4, no. 2, pp. 7-12, 2014.

[15] A. Dertat, "Applied deep learning-part 1: Artificial neural networks, 2017," URI: https://towardsdatascience.com/applied-deep-learningpart-1-artificial-neural-networks-d7834f67a4f6, 2018.

[16] A. Georgouli, Τεχνητή νοημοσύνη. [ebook], Athens: Hellenic Academic Libraries Link. Available Online at: http://hdl.handle.net/11419/3381, 2015.

[17] X. Wang and M. Benning, "Generalised Perceptron Learning," *arXiv preprint arXiv:2012.03642,* 2020.

[18] F. Rosenblatt, "The perceptron, a perceiving and recognizing automaton," *Project Para. Cornell Aeronautical Laboratory,* 1957.

[19] C. C. Aggarwal, Neural networks and deep learning, Springer 10, 2018, pp. 978-3.

[20] L. Wang, Yi Zeng and T. Chen, "Back propagation neural network with adaptive differential evolution algorithm for time series forecasting," *Expert Systems with Applications,* vol. 42, no. 2, pp. 855-863, 2015.

[21] Lipton, Zachary C., John Berkowitz and C. Elkan., "A critical review of recurrent neural networks for sequence learning," *arXiv preprint arXiv:1506.00019,* 2015.

[22] Bianchi, F. Maria and al., "An overview and comparative analysis of recurrent neural networks for short term load forecasting," *arXiv preprint arXiv:1705.04378,* 2017.

[23] H. Sepp and J. Schmidhuber, "Long short-term memory," *Neural computation,* vol. 9, no. 8, pp. 1735-1780, 1997.

[24] G. Van Houdt, C. Mosquera and G. Nápoles, "A review on the long short-term memory model," *Artif. Intell. Rev,* vol. 53, no. 8, pp. 5929-5955, 2020.

[25] F. A. Gers, J. Schmidhuber and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural computation,* vol. 12, no. 10, pp. 2451-2471, 2000.

[26] K. Cho and al., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078,* 2014.

[27] M. Schuster and K. K. Paliwal., "Bidirectional recurrent neural networks," *IEEE transactions on Signal Processing,* vol. 45, no. 11, pp. 2673-2681, 1997.

[28] K. Fukushima, "Neocognitron: A hierarchical neural network capable of visual pattern recognition," *Neural networks,* vol. 1, no. 2, pp. 119-130, 1988.

[29] S. S. Yadav and S. M. Jadhav., "Deep convolutional neural network based medical image classification for disease diagnosis," *Journal of Big Data,* vol. 6, no. 1, pp. 1-18, 2019.

[30] R. Johnson and T. Zhang, "Deep pyramid convolutional neural networks for text categorization," *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics,* vol. 1: Long Papers., 2017.

[31] D. Bhatt and al., "CNN Variants for Computer Vision: History, Architecture, Application, Challenges and Future Scope," *Electronics,* vol. 10, no. 20, p. 2470, 2021.

[32] K. Wróbel and al., "Compression of convolutional neural network for natural language processing," *Computer Science,* vol. 21, no. 1, 2020.

[33] "Automated Strabismus Detection based on Deep neural networks for Telemedicine Applications - Scientific Figure on ResearchGate," *Available from: https://www.researchgate.net/figure/The-network-architecture-composed-of-five-convolutional-layers-three-pooling-layers-and_fig4_327571320,* [accessed 30 Nov, 2021].

[34] I. Goodfellow, Yoshua Bengio and A. Courville., Deep learning., MIT press, , 2016.

[35] W. R. Sherman and A. B. Craig., "Understanding virtual reality," *San Francisco, CA: Morgan Kauffman,* 2003.

[36] "Malware Analysis of Imaged Binary Samples by Convolutional Neural Network with Attention Mechanism - Scientific Figure on ResearchGate," *Available from: https://www.researchgate.net/figure/Outline-of-the-convolutional-layer_fig1_323792694,* [accessed 30 Nov, 2021].

[37] K. O'Shea and R. Nash., "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458,* 2015.

[38] J. Zhou, Xibing Li and H. S. Mitri., "Classification of rockburst in underground projects: comparison of ten supervised learning methods," *Journal of Computing in Civil Engineering,* vol. 30, no. 5, p. 04016003, 2016.

[39] L. Cella and R. Martin., "Valid Inferential Models for Prediction in Supervised Learning Problems," *International Symposium on Imprecise Probability: Theories and Applications,* 2021.

[40] A. Dundar, J. Jin and E. Culurciello, "Convolutional clustering for unsupervised learning," *arXiv preprint arXiv:1511.06241,* 2015.

[41] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274,* 2017.

[42] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747,* 2016.

[43] P. Liashchynskyi and P. Liashchynskyi., "Grid search, random search, genetic algorithm: a big comparison for NAS," *arXiv preprint arXiv:1912.06059,* 2019.

[44] J. Lederer, "Activation functions in artificial neural networks: A systematic overview," *arXiv preprint arXiv:2101.09957,* 2021.

[45] C. Nwankpa and al., "Activation functions: Comparison of trends in practice and research for deep learning," *arXiv preprint arXiv:1811.03378,* 2018.

[46] A. L. Maas, A. Y. Hannun and A. Y. Ng., "Rectifier nonlinearities improve neural network acoustic models," *Proc. icml,* vol. 30, no. 1, 2013.

[47] D.-A. Clevert, T. Unterthiner and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289,* 2015.

[48] I. Goodfellow and al., "Maxout networks," *International conference on machine learning. PMLR,* 2013.

[49] X. Ying, "An overview of overfitting and its solutions," *Journal of Physics: Conference Series,* vol. 1168, no. 2, 2019.

[50] J. Kukačka, Vladimir Golkov and D. Cremers., "Regularization for deep learning: A taxonomy," *arXiv preprint arXiv:1710.10686,* 2017.

[51] A. Yadav and D. K. Vishwakarma., "Sentiment analysis using deep learning architectures: a review," *Artificial Intelligence Review,* vol. 53, no. 6, pp. 4335-4385, 2020.

[52] L. Abualigah and al., "Sentiment analysis in healthcare: a brief review," *Recent Advances in NLP: The Case of Arabic Language,* pp. 129-141, 2020.

[53] K. Mishev and al., "Evaluation of sentiment analysis in finance: from lexicons to transformers," *IEEE Access,* vol. 8, pp. 131662-131682, 2020.

[54] L. a. Stappen, "Sentiment analysis and topic recognition in video transcriptions," *IEEE Intelligent Systems,* vol. I36, no. 2, pp. 88-95, 2021.

[55] M. Popel and al., "Transforming machine translation: a deep learning system reaches news translation quality comparable to human professionals," *Nature communications,* vol. 11, no. 1, pp. 1-15, 2020.

[56] H. El Mostafa and F. Benabbou., "A deep learning based technique for plagiarism detection: a comparative study," *IAES International Journal of Artificial Intelligence,* vol. 9, no. 1, p. 81, 2020.

[57] D. Huang and al., "What Have We Achieved on Text Summarization?," *arXiv preprint arXiv:2010.04529,* 2020.

[58] J. H. Clark and al., "TyDi QA: A benchmark for information-seeking question answering in typologically diverse languages," *Transactions of the Association for Computational Linguistics,* vol. 8, pp. 454-470, 2020.

[59] D. N. K. Bank and R. Giryes., "Autoencoders," *arXiv preprint arXiv:2003.05991,* 2020.

[60] Q. Li and al., "Performance Evaluation of Deep Learning Classification Network for Image Features," *IEEE Access 9,* pp. 9318-9333, 2021.

[61] C. B. Murthy and al., "Investigations of object detection in images/videos using various deep learning techniques and embedded platforms—A comprehensive review," *Applied sciences,* vol. 10, no. 9, p. 3280, 2020.

[62] Q. Wang and al., "Overview of deep-learning based methods for salient object detection in videos," *Pattern Recognition,* vol. 104, p. 107340, 2020.

[63] A. Santhanavijayan, D. Naresh Kumar and G. Deepak., "A semantic-aware strategy for automatic speech recognition incorporating deep learning models," *Intelligent System Design,* pp. 247-254, 2021.

[64] D. Issa, M. F. Demirci and A. Yazici., "Speech emotion recognition with deep convolutional neural networks," *Biomedical Signal Processing and Control,* vol. 59, p. 101894, 2020.

[65] Y. Hu and al., "DCCRN: Deep complex convolution recurrent network for phase-aware speech enhancement," *arXiv preprint arXiv:2008.00264,* 2020.

[66] Q. Guo and al., "Application of deep learning in ecological resource research: Theories, methods, and challenges," *Science China Earth Sciences,* pp. 1-18, 2020.

[67] S. I. Lee, Yoo. and S. Joon, "Multimodal deep learning for finance: integrating and forecasting international stock markets," *The Journal of Supercomputing,* vol. 76, no. 10, pp. 8294-8312, 2020.

[68] R. Shraga and al., "Web table retrieval using multimodal deep learning," *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval,* 2020.

[69] H. S. Munawar and al., "UAVs in Disaster Management: Application of Integrated Aerial Imagery and Convolutional Neural Network for Flood Detection," *Sustainability,* vol. 13, no. 14, p. 7547, 2021.

[70] S. Shamshirband and al., "A review on deep learning approaches in healthcare systems: Taxonomies, challenges, and open issues," *Journal of Biomedical Informatics,* vol. 113, p. 103627, 2021.

[71] R. Zemouri, N. Zerhouni and D. R. , "Deep learning in the biomedical applications: Recent and future status," *Applied Sciences,* vol. 9, no. 8, p. 1526, 2019.

[72] C. Cao and al., "Deep learning and its applications in biomedicine," *Genomics, proteomics & bioinformatics,* vol. 16, no. 1, pp. 17-32, 2018.

[73] S. K. Zhou and al., "A review of deep learning in medical imaging: Imaging traits, technology trends, case studies with progress highlights, and future promises," *Proceedings of the IEEE,* 2021.

[74] Y.-W. Chen and L. C. Jain, Deep Learning in Healthcare, 2020.

[75] García Ocaña M.I., López-Linares Román K., Lete Urzelai N., González Ballester M.Á. and M. O. I., "Medical Image Detection Using Deep Learning," *In: Chen YW., Jain L. (eds) Deep Learning in Healthcare,* vol. 171, 2020.

[76] A. Hatamizadeh and al., "Unetr: Transformers for 3d medical image segmentation," *arXiv preprint arXiv:2103.10504,* 22021.

[77] J. Chen and al., "Transunet: Transformers make strong encoders for medical image segmentation," *arXiv preprint arXiv:2102.04306,* 2021.

[78] J. Li and al., "Medical image segmentation in oral-maxillofacial surgery," *Computer-Aided Oral and Maxillofacial Surgery,* pp. 1-27, 2021.

[79] F. F. Ting, Y. J. Tan and K. S. Sim., "Convolutional neural network improvement for breast cancer classification," *Expert Systems with Applications,* vol. 120, pp. 103-115, 2019.

[80] S. Azizi and al., "Big self-supervised models advance medical image classification," *arXiv preprint arXiv:2101.05224,* 2021.

[81] Q. Zhang and al., "A GPU-based residual network for medical image classification in smart medicine," *Information Sciences,* vol. I536, pp. 91-100, 2020.

[82] Y. Li, Y. Iwamoto and Y.-W. Chen., "Medical Image Enhancement Using Deep Learning," *Deep Learning in Healthcare,* pp. 53-76, 2020.

[83] M. Blendowski, N. Bouteldja and M. P. Heinrich., "Multimodal 3D medical image registration guided by shape encoder–decoder networks," *International journal of computer assisted radiology and surgery,* vol. 15, no. 2, pp. 269-276, 2020.

[84] Y. Li and al., "Medical image fusion method by deep learning," *International Journal of Cognitive Computing in Engineering,* vol. 2, pp. 21-29, 2021.

[85] Y. Fu and al., "Deep learning in medical image registration: a review," *Physics in Medicine & Biology,* vol. 65, no. 20: 20TR01, 2020.

[86] M. Mahmud and al., "Applications of deep learning and reinforcement learning to biological data," *IEEE transactions on neural networks and learning systems,* vol. 29, no. 6, pp. 2063-2079, 2018.

[87] our Team, Join and Team Photo Gallery, "DNA and RNA," *Big Data, RNA Matters Lecture Series, Novel Human miRNA, and Ask a Scientist.*

[88] G. Dovey., "DNA Replication," *TeachMePhysiology,* 2021.

[89] L.-C. Shen and al., "SAResNet: self-attention residual network for predicting DNA-protein binding," *Briefings in Bioinformatics,* 2021.

[90] R. A. C. Ferraz, "DNA–protein interaction studies: a historical and comparative analysis," *Plant Methods,* vol. 17, no. 1, pp. 1-21, 2021.

[91] D. V. Volgin, "Gene Expression: Analysis and Quantitation," *Animal Biotechnology,* pp. 307-325, 2014.

[92] X. Du and al., "Deepss: Exploring splice site motif through convolutional neural network directly from dna sequence," *IEEE Access,* vol. 6, 2018: 32958-32978.

[93] S. A. B. Parisapogu, C. S. R. Annavarapu and M. Elloumi., "1-Dimensional convolution neural network classification technique for gene expression data," *Deep Learning for Biomedical Data Analysis,* pp. 3-26, 2021.

[94] J. X. Zhang and al., "A deep learning model for predicting next-generation sequencing depth from DNA sequence," *Nature communications,* vol. 12, no. 1, pp. 1-10, 2021.

[95] Q. Tian and al., "MRCNN: a deep learning model for regression of genome-wide DNA methylation," *BMC genomics,* vol. 20, no. 2, pp. 1-10, 2019.

[96] G. Egger, P. Arimondo and eds, "Drug Discovery in Cancer Epigenetics," *Academic Press,* 2015.

[97] J. Jumper and al., "Highly accurate protein structure prediction with AlphaFold," *Nature 596.7873,* pp. 583-589, 2021.

[98] M. Gao, Hongyi Zhou, and and J. Skolnick., "DESTINI: A deep-learning approach to contact-driven protein structure prediction," *Scientific reports,* vol. 9, no. 1, pp. 1-13, 2019.

[99] Y.-X. Tan and al., "Induction of apoptosis by the severe acute respiratory syndrome coronavirus 7a protein is dependent on its interaction with the Bcl-XL protein," *Journal of virology,* vol. 81, no. 12, pp. 6346-6355, 2007.

[100] M. Chen and al., "Multifaceted protein–protein interaction prediction based on siamese residual rcnn," *Bioinformatics,* vol. 35, no. 14, pp. i305-i314, 2019.

[101] M. Zeng and al., "Protein–protein interaction site prediction through combining local and global features with deep neural networks," *Bioinformatics,* vol. 36, no. 4, pp. 1114-1120, 2020.

[102] L. Xie and al., "Deep learning-based transcriptome data classification for drug-target interaction prediction," *BMC genomics,* vol. 19, no. 7, pp. 93-102, 2018.

[103] M. Tsubaki, K. Tomii and J. Sese, "Compound–protein interaction prediction with end-to-end learning of neural networks for graphs and sequences," *Bioinformatics,* vol. 35, no. 2, pp. 309-318, 2019.

[104] "https://www.cancer.gov/tcga".

[105] R. C. Team, "R: A language and environment for statistical Computing," Vols. 4.1.2 (2021-11-01), 2021.

[106] A. Colaprico, T. C. Silva, C. Olsen, L. Garofano, C. Cava, D. Garolini, T. S. Sabedot, T. M. Malta, S. M. Pagnotta, I. Castiglioni, M. Ceccarelli, G. Bontempi and H. Noushmehr, "TCGAbiolinks: an R/Bioconductor package for integrative analysis of TCGA data," *Nucleic acids research,* 2015.

[107] T. Silva, A. Colaprico, C. Olsen, F. D'Angelo, G. Bontempi, M. Ceccarelli and H. Noushmehr, "TCGA Workflow: Analyze cancer genomics and epigenomics data using Bioconductor packages," *F1000Research,* 2016.

[108] M. Mounir, M. Lucchetta, C. T. Silva, C. Olsen, G. Bontempi, X. Chen, H. Noushmehr, A. Colaprico and E. Papaleo, New functionalities in the TCGAbiolinks package for the study and integration of cancer data from GDC and GTEx, vol. 15(3), PLoS computational biology, 2019.

[109] W. N. Venables and B. D. Ripley, Modern Applied Statistics with S, Fourth ed., New York: Springer, 2002.

[110] W. H., "stringr: Simple, Consistent Wrappers for Common String Operations," *http://stringr.tidyverse.org, https://github.com/tidyverse/stringr,* 2021.

[111] M. Morgan, V. Obenchain, J. Hester and H. Pagès, SummarizedExperiment: SummarizedExperiment container. R package, vol. version 1.24.0, 2021.

[112] V. Rossum, G. a. Drake and F. L., Python 3 Reference Manual, Scotts Valley, CA: CreateSpace, 2009.

[113] B. Lyu and A. Haque, ""Deep learning based tumor type classification using gene expression data," *Proceedings of the 2018 ACM international conference on bioinformatics, computational biology, and health informatics,* 2018.

[114] F. Alharbi and e. al., "Fine-Tuning Pre-Trained Convolutional Neural Networks for Women Common Cancer Classification using RNA-Seq Gene Expression," *International Journal of Advanced Computer Science and Applications,* vol. 11, no. 11, 2020.

[115] M. K. Elbashir and e. al., "Lightweight convolutional neural network for breast cancer classification using RNA-seq gene expression data," *IEEE Access 7,* pp. 185338-185348, 2019.

[116] M. Abadi, A. Agarwal, P. Barham and e. al., TensorFlow: Large-scale machine learning on heterogeneous systems, Software available from tensorflow.org, 2015.

[117] A. Gulli and S. Pal, Deep learning with Keras, Packt Publishing Ltd, 2017.

[118] Impact of Artificial Neural Networks Training Algorithms on Accurate Prediction of Property Values - Scientific Figure on ResearchGate, Available from: https://www.researchgate.net/figure/The-structure-of-the-artificial-neuron_fig2_328733599, [accessed 10 Nov, 2021].

[119] S. Albawi, Tareq Abed Mohammed and S. Al-Zawi., "Understanding of a convolutional neural network," *2017 International Conference on Engineering and Technology (ICET),* Ieee, 2017.

[120] H. Zhao and al., "Evaluation of three deep learning models for early crop classification using sentinel-1A imagery time series—A case study in Zhanjiang, China," *Remote Sensing,* vol. 11, no. 22, p. 2673, 2019.

[121] S. Robinson, "Introduction to Neural Networks with Scikit-Learn," *Tersedia: https://stackabuse. com/introduction-toneural-networks-with-scikit-learn,* 2018.