National Technical University of Athens
School of Electrical and Computer Engineering

Department of Technology, Informatics and Computers
Artificial Intelligence and Learning Systems Laboratory

Prediction of Covid-19 disease evolution as a time series using machine learning

Extended Thesis Summary

of

Admitos-Rafael Passadakis

Supervisor: Stefanos Kollias
Professor of N.T.U.A

Co-supervisor: Paraskevi Tzouveli
Laboratory and Teaching Staff of N.T.U.A

Athens, July 2021

Contents

List of Figures

List of Tables

1 Introduction

The coronovirus pandemic, commonly known as COVID-19 pandemic is a severe acute respiratory disease that is caused by the SARS-COV-2 respiratory virus. It can consequence to severe impairment of vital human organs or even death. It first appeared in the fall of 2019 (31 December) in the Chinese city of Wuhan, where the first case and death were recorded (January 2020). Up to May of 2021, around 160 million cases of COVID-19 have been recorded with more than 3.5 million deaths. The COVID-19 pandemic have had a serious impact in human life and activity and during the last 16 months the political, social and economical scene around the world have changed drastically.

Since the beginning of this pandemic our routine has been inextricably connected with the daily data of this disease which indicate its future development. More specifically, we are interested in the number of the cumulative and (mainly) daily tests, cases, deaths or admissions to Hospitals or to Intensive Care Units (ICUs). This data, since they are developed according to time, can be interpreted as a chronological sequence and thus we call them *time series*. *Time series analysis* includes methods of time series processing for the extraction of crucial statistical features. On the other hand, *time series forecasting* is the usage of models and specific tools for the future prediction of values in time series based on the values that have been observed in the past. The problems of analyzing and forecasting time series are well known in many areas of Data Science.

As a result, during the last decades, several types of mathematical models and formulas have been employed in order to analyze and predict time series. The most well-known of them are the auto-regressive (AR), integrated (I) and moving average (MA) models. In many occasions, combinations of these types of models have achieved great results. Such models are the ARMA (auto-regressive moving average) and ARIMA (auto-regressive integrated moving average). More contemporary attempts in the field of time series forecasting incorporate *machine learning* models. The present work, stepped in that axis as well. Later on, we will describe how we used such models for this specific purpose.

In order to achieve the desired result we divided the structure of this work in two main parts. First, we explore the data of this disease through mathematical formulas and then we illustrate them via statistical plots and graphs. In the second and more extensive part, we aim at the acquaintance and basic mathematical foundation of specific models of machine learning, neural networks. Out of them, we describe in detail the recurrent (mostly) and convolutional neural networks (RNNs and CNNs), their couplings and Encoder-Decoder architectures with an Attention mechanism. We deploy these models as well as specific techniques of time series forecasting so to predict the future evolution of COVID-19 disease (in terms of cases and deaths) in global and national level.

Furthermore, we conduct experiments in which our models use rolling update (feedback) mechanisms to predict the evolution of the disease beyond the time horizon of the dataset and we also construct special networks such as Autoencoders that are enabled to detect extreme values (anomalies) in the data (possible signs of sudden outbreak/shrinking of the pandemic). Lastly, we propose a method where, through the efficacy of the models, we can draw to some extent, certain geographical characteristics and inferences as to the image of the pandemic across the countries of the world.

2 Data Visualization of COVID-19

As mentioned above, by early May of 2021, approximately 160 million cases of the disease have been recorded and 3.5 million deaths have been caused by COVID-19. We can define as *time step*, the frequency of the collected data. In our case, time step coincides with every calendar day. Therefore, at each time step we can take the ratio between the total number of deaths and the total number of cases in order to take the Case Fatality Ratio (CFR), as:

$$CFR = \frac{Total \quad Deaths}{Total \quad Cases} \tag{2.1}$$

In national level United States of America is the country most severely damaged by COVID-19 recording approximately 33,000,000 cases and 600,000 deaths followed by India and Brazil which stood up at 18,000,000 and 15,000,000 cases and more than 200,000 and 400,000 deaths correspondingly. In order to be aware of what degree has the virus has been transmitted inside a closed social body is necessary to define normalization metrics referred to the total population of the whole. These metrics are called *cases and deaths per million* and can be defined as (in the context of a country):

$$\frac{c}{m} = \frac{Total \quad cases \quad in \quad the \quad country}{Country's \quad total \quad population} \times 10^6 \tag{2.2}$$

$$\frac{d}{m} = \frac{Total \quad deaths \quad in \quad the \quad country}{Country's \quad total \quad population} \times 10^6 \tag{2.3}$$

Another interesting metric in measuring COVID-19 dispersal is the so called *positivity rate* which is defined as:

$$Positivity \quad Rate = \frac{Nunber \quad of \quad cases}{Number \quad of \quad tests} \times 100\% \tag{2.4}$$

Figure 1 pie charts vividly depict the top ten countries with most confirmed cases and deaths that are attributed to COVID-19 indicating the percentage possession for each nation.

COVID-19 can be described as serious disease which can lead to hospitalization, intubation or even death. Thus, it is crucial to keep track of the possibility a patient that is contracted by the virus to have a negative outcome and be in need of ICU or intubation. Having that in mind, we can define the conditional probabilities a patient that is hospitalized to deteriorate and be in urgent need of ICU or intubation, as:

$$P_I(I|H) = \frac{N_I}{N_H} \times 100\%, \quad P_V(V|H) = \frac{N_V}{N_H} \times 100\% \tag{2.5}$$

where H, I and V are the events, a patient to be admitted to hospital, ICU or get intubated respectively and N_H, N_I and N_V are the numbers of patients in hospitalization, ICU and intubation units correspondingly.

Another intriguing question arises when we want to find out the geographical distribution of COVID-19 in terms of cases, deaths or other metrics such the aforementioned CFR or the "per million metrics". Data Science is a field that gives an answer to that question by providing specific tables or even maps that illustrate the staggered distribution of specific data among the world's nations. More specifically, this work utilized, the more commonly known *geographical heatmaps* in order to represent such information. Throughout the construction of those maps,

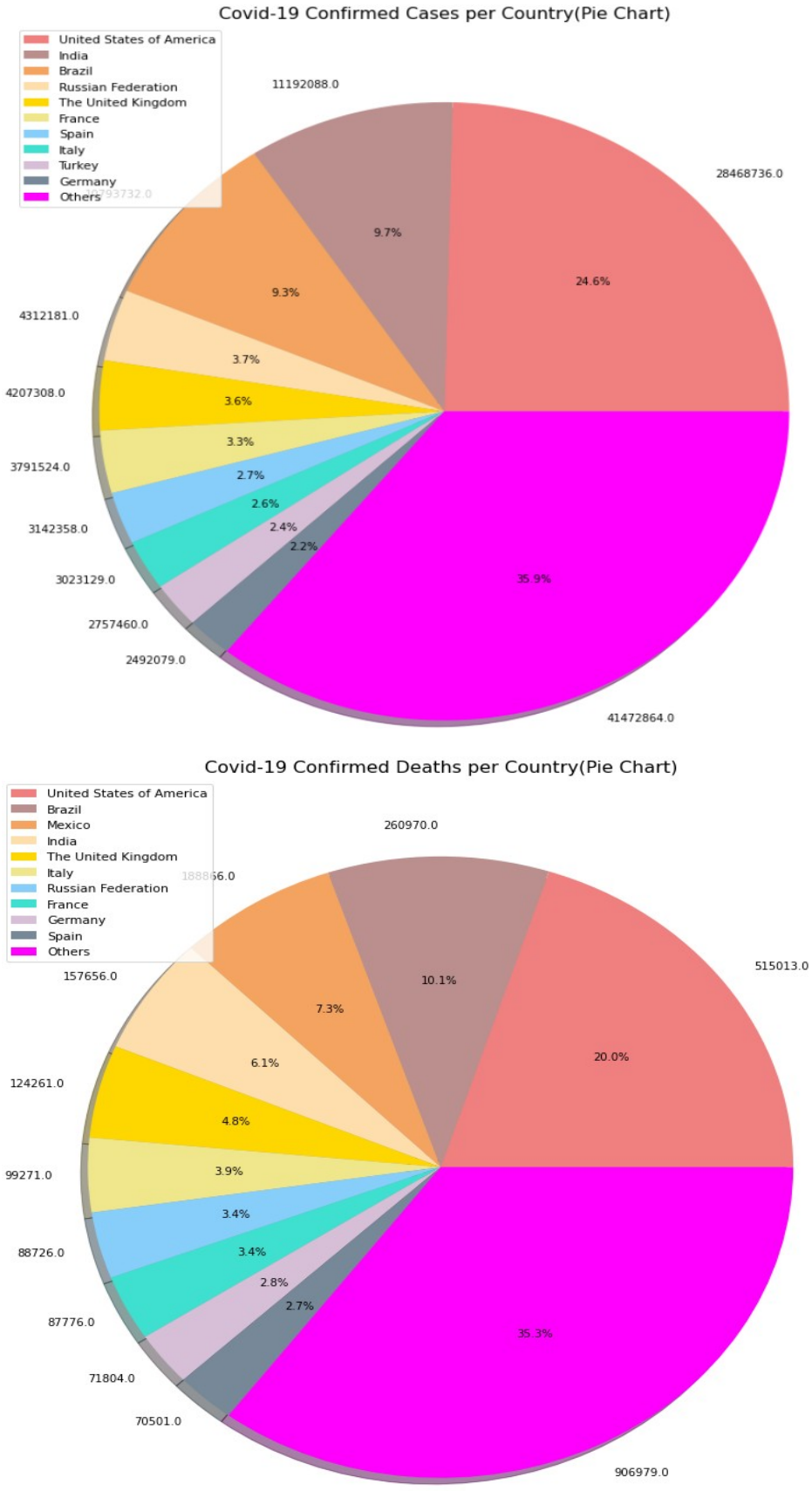


Figure 1: Top ten (10) countries most affected by the pandemic

linearization of the data (due to their vast arithmetic divergence) had to take place, and so we used the natural logarithm, as (only for the actual number of cases and deaths):

$$\text{number of cases}_{lin} = \ln(\text{number of cases}) \tag{2.6}$$

$$\text{number of deaths}_{lin} = \ln(\text{number of deaths}) \tag{2.7}$$

Figures 2 and 3 show the formed geographical heatmaps of COVID-19 cases and deaths per million around the world.

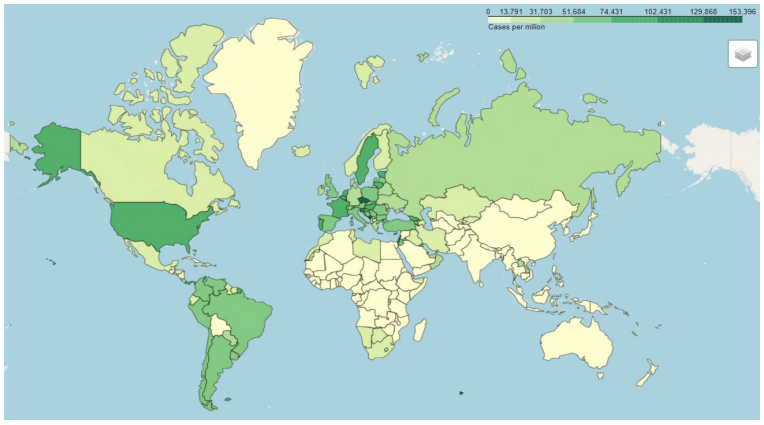


Figure 2: Geographical heatmap of cases per million

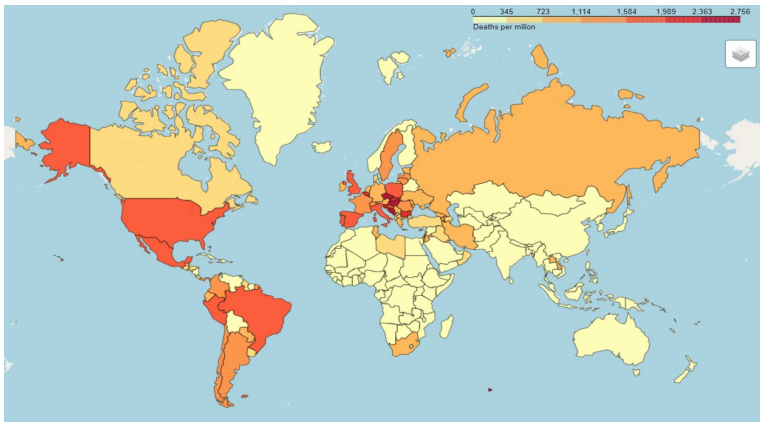


Figure 3: Geographical heatmap of deaths per million

3 Neural Networks

3.1 Simple Neural Networks

As mentioned before the main goal of this project is to use machine learning models in order to deduce inferences about the evolution of COVID-19 worldwide and predict its future development. These machine learning models are widely known to the scientific community as *neural networks*.

The simplest neural network can be characterised as a plain *regression model* which operates under the following vector equation:

$$\mathbf{o} = W\mathbf{x} + \mathbf{b} \tag{3.1}$$

In equation (3.1) $\mathbf{x}, \mathbf{o}$ and $\mathbf{b}$ are the input, output and bias vectors respectively. We can generalise equation (3.1) by obtaining more than one inputs and thus resulting in:

$$O = XW + \mathbf{b} \tag{3.2a}$$

$$\hat{Y} = \text{softmax}(O) \tag{3.2b}$$

where $X \in \mathbb{R}^{n\times d}$ (input matrix), $W \in \mathbb{R}^{d\times q}$ (Weight matrix), $\mathbf{b} \in \mathbb{R}^{1\times q}$ (bias vector) and $\hat{Y} \in \mathbb{R}^{n\times q}$ (output probability matrix) whilst n is the number of input batches, d the number of input features and q the number of outputs. The *softmax* function is used so to express the output of the model as a probability$\in [0, 1]$. Nevertheless, networks described by the set of matrix equations (3.2) are linear and cannot model any non-linear procedure. Therefore, we need an embedded extra layer (other than the input and output layer) that can process the inbound information implicitly. This layer is dubbed as *hidden layer* and this kind of networks are called *Multi Layer Perceptrons (MLPs)*. The describing set of equations of such models is given as showed below:

$$H = \phi(XW_{xh} + b_h) \tag{3.3a}$$

$$O = HW_{hq} + b_q \tag{3.3b}$$

where $X \in \mathbb{R}^{n\times d}$, $H \in \mathbb{R}^{n\times h}$ (Hidden matrix), $b_h \in \mathbb{R}^{1\times h}$ and $W_{xh} \in \mathbb{R}^{h\times h}$ while ϕ is the activation function, h the plurality of neurons and q the number of outputs.

3.2 Recurrent Neural Networks

The *Recurrent Neural Network (RNN)* is a type of Neural Network where connections between its nodes can form a directed cyclic or acyclic graph along a temporal sequence. The acyclic type of RNNs can be unrolled and be represented as feedforward neural networks while cyclic cannot. This type of networks contain a specific state called *Hidden State* which allows us to store crucial information that has been met in the past while processing temporal sequential data. The basic mathematical formula that models RNN's operation can be given by the following equations:

$$H_t = \phi(X_t W_{xh} + H_{t-1} W_{hh} + b_h) \tag{3.4}$$

$$O_t = H_t W_{hq} + b_q \tag{3.5}$$

In equations (3.4) and (3.5) $H_t \in \mathbb{R}^{n\times h}$ denotes the hidden state of the model, $X_t \in \mathbb{R}^{n\times d}$ the input and $O_t \in \mathbb{R}^{n\times q}$ the output at each time step. Moreover, $W_{xh},\ W_{hh},\ W_{hq},\ b_h$ and b_q are all trainable parameters of the model.

The *Long-Short Term Memory (LSTM)* model is a generalization of the classic RNN model and it is proved to be exceptionally effective in the processing and prediction of temporal sequential data such as the COVID-19 data case. It was introduced in order to deal with the vanishing/exploding gradient of the basic RNNs and to be capable of handling sequences of large

length. The equations that describe the function of the LSTM are the following:

$$F_t = \sigma(X_t W_{xf} + H_{t-1} W_{hf} + b_f) \tag{3.6}$$
$$I_t = \sigma(X_t W_{xi} + H_{t-1} W_{hi} + b_i) \tag{3.7}$$
$$C_i = tanh(H_{t-1} W_{hc} + X_t W_{xc} + b_c) \tag{3.8}$$
$$O_t = \sigma(X_t W_{xo} + H_{t-1} W_{ho} + b_o) \tag{3.9}$$
$$C_t = F_t \odot C_{t-1} + I_t \odot C_i \tag{3.10}$$
$$H_t = O_t tanh(C_t) \tag{3.11}$$

Sequence X_t is the input of the model, H_t symbolises the *hidden state* and $C_i \in \mathbb{R}^{n \times h}$ and $C_t \in \mathbb{R}^{n \times h}$ are the *candidate cell state* and the end *cell state* respectively whilst I_t, F_t and O_t are called *input*, *forget* and *output* gates correspondingly. All of W_{xf}, W_{hf}, W_{xi}, W_{hi}, W_{hc}, W_{xc}, W_{xo}, W_{ho}, b_f, b_c, b_i, b_o are trainable parameters of the model and σ denotes the sigmoid function.

The *Gated Recurrent Unit (GRU)* was proposed in order simplify the to complexity in the structure of LSTM. At the same time it retained the gated formation which deals with the problem of vanishing/exploding gradient and the handling of lengthy sequences. The equations that describe the GRU operation can be seen below:

$$R_t = \sigma(X_t W_{xr} + H_{t-1} W_{hr} + b_r) \tag{3.12}$$
$$Z_t = \sigma(X_t W_{xz} + H_{t-1} W_{hz} + b_z) \tag{3.13}$$
$$\tilde{H}_t = \tanh(X_t W_{xh} + (R_t \odot H_{t-1}) W_{hh} + b_h) \tag{3.14}$$
$$H_t = Z_t \odot H_{t-1} + (1 - Z_t) \odot \tilde{H}_t \tag{3.15}$$

We call R_t as *reset* gate and Z_t as *update* gate while H_t and $\tilde{H}_t$ are the *Hidden state* and the *candidate Hidden state* respectively. The weights W_{xr}, W_{xz}, b_r, b_z, W_{xh}, W_{hh} and b_h are all trainable parameters of the model.

All types of RNNs described till now can process information in a single direction ignoring possible crucial data from the future. In order to pay heed to future information we can handle our data in the opposite direction. We can achieve this by stacking two recurrent layers: the first processes information from the beginning to the end of the temporal sequence (forward layer) and the second implements an opposite handling (backward layer). We call these models as *Bidirectional RNNs* and their standard mathematical formulas for can be stated as follows:

$$\overrightarrow{H}_t = \phi(X_t W_{xh}^{(f)} + \overrightarrow{H}_{t-1} W_{hh}^{(f)} + b_h^{(f)}) \tag{3.16}$$
$$\overleftarrow{H}_t = \phi(X_t W_{xh}^{(b)} + \overleftarrow{H}_{t+1} W_{hh}^{(b)} + b_h^{(b)}) \tag{3.17}$$
$$O_t = H_t W_{hq} + b_q \tag{3.18}$$

In the above equations $\overrightarrow{H}_t \in \mathbb{R}^{n \times h}$ and $\overleftarrow{H}_t \in \mathbb{R}^{n \times h}$ indicate the forward and backward pass of the information respectively, while $\overrightarrow{H}_{t-1}$ is the information from the previous (past) time step and $\overleftarrow{H}_{t+1}$ is the information from the next (future) time step. The total hidden state $H_t \in \mathbb{R}^{n \times 2h}$ is the concatenated $\overrightarrow{H}_t$ and $\overleftarrow{H}_t$ at every time step. Once again, $W_{xh}^{(f)}$, $W_{hh}^{(f)}$, $b_h^{(f)}$, $W_{xh}^{(b)}$, $W_{xh}^{(b)}$, $b_h^{(b)}$ are all trainable parameters of the model related to the forward and backward layer correspondingly. The same stands for the output weight matrix $W_{hq} \in \mathbb{R}^{2h \times q}$ and the output bias vector $b_q \in \mathbb{R}^{1 \times q}$.

3.3 Convolutional Neural Networks

Convolutional neural networks, unlike recurrent networks, are particularly powerful at retaining spatial information by extracting the important features from the set of data they analyze. The output of a convolutional layer between a 3-dimensional input image $[X]_{i,j,k}$ and a rolling 3-dimensional convolutional filter (or kernel) $[W]_{a,b,c,d}$ inside a bounded region of area $4\Delta^2$ of our input can be described by the following equation:

$$[H]_{i,j,d} = \sum_{a=-\Delta}^{\Delta} \sum_{b=-\Delta}^{\Delta} \sum_{c} [W]_{a,b,c,d}[X]_{i+a,j+b,c} \tag{3.19}$$

Generalising, the output size $[H]_{i,j,d}$ is given by the input size $n_h \times n_w$ minus the size of the convolutional kernel $k_h \times k_w$ via the subsequent equation: $(n_h - k_h + 1) \times (n_w - k_w + 1)$. A common method in convolutional operation is the *padding* in which we enlarge the size of our input in order not to lose crucial information during the convolution and thus we add extra pixel to the input image. For such an operation in 2 dimensions, where we increase the input size by (p_h, p_w) the final dimension of our output will be given by: $(n_h - k_h + p_h + 1) \times (n_w - k_w + p_w + 1)$.

In plenty of occasions the rolling window can be crawling on the input with step greater than one. That step is called *stride* and it affects the final size of the output. In fact, for a convolutional layer which pads the input and the kernel has stride $(s_h, s_w) > (1, 1)$ the size of the output is given by:

$$\lfloor \frac{n_h - k_h + p_h + s_h}{s_h} \rfloor \times \lfloor \frac{n_w - k_w + p_w + s_w}{s_w} \rfloor \tag{3.20}$$

where the $\lfloor \quad \rfloor$ indicate the rounding down. Lastly, many convolutional networks perform the *pooling* function as well. This mechanism contributes so to mitigate the the sensitivity of deep convolutional layers in locality and to avoid the excessive spatial reduction of data dimensionality. A *polling window* is rolling through the input data and accumulates adjacent information by carrying out either *maximum pooling* or *average pooling*. The entire convolutional operation and pooling operation can be seen in the images of Figure 4.

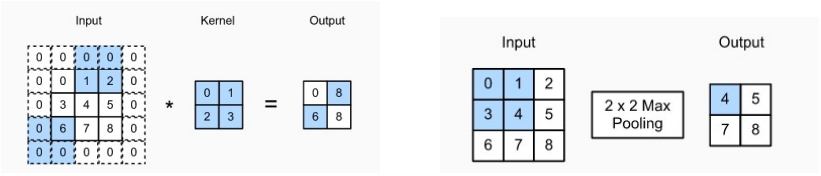


Figure 4: Convolutional (left) and Pooling (right) operations

3.4 Encoder-Decoder Structure/Attention Mechanism

In many machine learning problems we want to predict an output sequence, given an input sequence of different length, with no exact correspondence between each input and each output feature. This technique is named *sequence to sequence mapping*. *Encoder-Decoder* is a model that performs this kind of technique by receiving an input sequence and producing an other sequence as an output (possibly of different length). It's main 2 components can be detailed as follows:

- **Encoder**: It is responsible for passing and encoding an input sequence through successive time steps in a constant length vector called *context vector*. The encoder can be reconstructed by any of the previously presented RNN models. Assuming a chronological input

sample x_t where $t \in [1, T]$ the encoder converts the corresponding vector $\mathbf{x_t}$ of the input sample and the hidden state $\mathbf{h_{t-1}}$ from the previous time step to the current hidden state $\mathbf{h_t}$ through a nonlinear transformation f, as:

$$\mathbf{h}_t = f(\mathbf{x}_t, \mathbf{h}_{t-1}) \tag{3.21}$$

In general, the encoder converts hidden states at any time step in the context variable ($\mathbf{c}$) using an adapted nonlinear function q, as:

$$\mathbf{c} = q(\mathbf{h}_1, \ldots, \mathbf{h}_T) \tag{3.22}$$

- **Decoder**: It is responsible for passing information through successive output time steps while processing the context vector. Given the output sequence $y_1, y_2, \ldots, y_{T'}$ from the training data set, for each step t' (note, the symbol differs from the time step t of the encoder input sequences due to possible difference in sequence length), the probability of decoder output being $y_{t'}$ depends on the previous outputs $y_1, \ldots, y_{t'-1}$ and the context variable $\mathbf{c}$ is: $P(y_{t'} \mid y_1, \ldots, y_{t'-1}, \mathbf{c})$. We can model this conditional probability by assuming another recurrent network operating as a decoder with it's hidden state vector being represented with $\mathbf{s_{t'}}$. For each time step t' of the decoder, this hidden state depends on the previous hidden state $\mathbf{s_{t'-1}}$, the context vector $\mathbf{c}$ and the previous output $y_{t'-1}$. We use also, another nonlinear function g to express the transformation of the hidden layer of the decoder, as:

$$\mathbf{s}_{t'} = g(y_{t'-1}, \mathbf{c}, \mathbf{s}_{t'-1}) \tag{3.23}$$

The final conditional probability $P(y_{t'} \mid y_1, \ldots, y_{t'-1}, \mathbf{c})$ can be obtained easily by using a *softmax* function in the latter output layer of the network and an appropriate weight matrix V, as:

$$\hat{y} = softmax(Vs(t')) \tag{3.24}$$

Even with the Encoder-Decoder extension, it is very difficult to summarize long information in a single vector as the length of the sequence increases, resulting the model often to forget the previous parts of the input sequence while processing the latest parts. In order to tackle this issue, we can use an *Attention mechanism*, which tries to give more emphasis to the most important pieces of information of the input sequence. There are 3 different types of attention mechanisms:

- **Self Attention**: The idea is to correlate all the different elements of the hidden state, which have come from the input sequence, to each other.
- **Global Attention**: The idea is to draw a context vector c_t, based on all the hidden states of the encoder. Therefore, this kind of mechanism monitors the entire entrance batch.
- **Local Attention**: The idea is to eliminate the cost of Global Attention by focusing only on a small subset of the input sequences. Considering a specific alignment position, we create a window $[p_t - D, p_t + D]$ on the input sequence, where D is half the length of the window and we ignore the information that exceeds these limits.

This work approached and utilized most of all the Global Attention form. Let us consider that in an Encoder-Decoder model that implements sequence to sequence mapping we symbolize

with $\mathbf{h}_k$ the hidden states of the Encoder and with $\mathbf{s}_t$ the hidden states of the Decoder (both t, k refer as time variables). The main two global attention equations can be described as follows:

$$\mathbf{e}_{t,k} = \mathbf{V}^T tanh(\mathbf{W_2 s}_{t-1}, +\mathbf{W_1 h}_k + \mathbf{b}) \tag{3.25}$$

$$\boldsymbol{\alpha}_{k,t} = softmax(\mathbf{e}_{t,k}) = \frac{exp(\mathbf{e}_{t,k})}{\sum_{k=1}^{N} exp(\mathbf{e}_{t,k})} \tag{3.26}$$

In these equations $\mathbf{e}_{t,k}$ is also referred as the *alignment score or vector*, $\alpha_{t,k}$ as *attention weights or probabilities* and contain the score of how much attention should be put on the k^{th} feature of the sequence by the decoder. N is the encoder's time steps. In (3.25) $\mathbf{W_1}$ and $\mathbf{W_2}$ are trainable weights associated to the hidden states of the encoder and the decoder respectively while $\mathbf{V}$ are weights for the alignment score and $\mathbf{b}$ is a bias vector initialized at zero. The attention mechanism functions as a layer of the network that is adapted and consequently is trained like the rest.

Now the context vector (output of the attention layer) can be expressed as the weighted sum of all the hidden values of the encoder according to the attention weights:

$$\mathbf{c_t} = \sum_{k=1}^{T} \boldsymbol{\alpha}_{k,t} \mathbf{h}_k \tag{3.27}$$

where T is the time steps of the decoder. Hence, the current hidden state $\mathbf{s_t}$ is calculated according to the nonlinear output activation function g, taking as input the context vector $\mathbf{c_t}$, the hidden state $\mathbf{s_{t-1}}$ and the output $\hat{y}_{t-1}$ of the previous time step, like below:

$$\mathbf{s_t} = g(\mathbf{c_t}, \mathbf{s_{t-1}}, \hat{y}_{t-1}) \tag{3.28}$$

Another possible way of inserting the attention mechanism in the network is to align the hidden and cell states given by the Encoder $\mathbf{h}_t$ and $\mathbf{C}_t$ with a certain mini batch of the input vector $\mathbf{x_k} = ({x_1}^k, {x_2}^k, ...{x_m}^k) \in \mathbb{R}^m$, as:

$$\mathbf{e_{k,t}} = \mathbf{V}^T tanh(\mathbf{W_1}[\mathbf{h}_{t-1}, \mathbf{C}_{t-1}] + \mathbf{W_2 x}_k + \mathbf{b}) \tag{3.29}$$

Finally, we can get the output of the attention as the dot product between the attention weights and the initial inputs by the following equation:

$$\mathbf{c_t} = (\boldsymbol{\alpha}_{j,t} \cdot \mathbf{x}_k)^T = ({\alpha_t}^1 {x_t}^1, {\alpha_t}^2 {x_t}^2, ..., {\alpha_t}^n {x_t}^n)^T \tag{3.30}$$

4 Techniques Analysis

As mentioned earlier the aim of this work is to predict the upcoming cases and deaths in a focused region of the world or the entire globe itself. In order to achieve that, we collected data from the official website of the World Health Organization (WHO) and other worldwide known websites such the Our World in Data website. The dataset consisted of a Variety of features such as daily and cumulative tests, cases and deaths, number of patients in hospitals (single beds), in ICU or intubation.

As designated from the title of this work the prediction was conducted on a time series data. These time series where composed (mainly) by the daily cases and deaths of each country. In time series forecasting can be distinguished in 2 different methods:

- **Univariate Forecasting**: Usage only the past of the observation variable to predict it.
- **Multivariate Forecasting**: Usage of additional features to predict the observation variable (e.g. For cases → usage of tests OR for deaths → Ussage of the patients in Hospitals & ICUs).

In order to decide whether which features are interrelated we use a *Correlation Matrix*. This matrix is symmetrical, positive semi-defined in which the elements of the main diagonal are all 1 and indicate the autocorrelation values. All values in this matrix vary in $[-1, 1]$, where -1 signs negative correlation, 0 signs no correlation and 1 full (positive) correlation. Let's consider an observation $\mathbf{X} = [X_i, ..., X_n]$ then the correlation matrix will be of $n \times n$ dimension and will contain the normalized mean values of the random variables X_i divided by the standard deviation σ of X_i, that is, $X_i/\sigma(X_i)$ for $i = 1, ..., n$, as seen in equation (4.1) (with μ_i we symbolize the mean value of random variables X_i).

As it is already probably known the COVID-19 pandemic didn't initiated in all countries at the same date. It was first appeared in China and later in all other nations.

$$corr(\mathbf{X}) = \begin{bmatrix} 1 & \frac{E[(X_1-\mu_1)(X_2-\mu_2)]}{\sigma(X_1)\sigma(X_2)} & ... & \frac{E[(X_1-\mu_1)(X_2-\mu_2)]}{\sigma(X_1)\sigma(X_2)} \\ \frac{E[(X_2-\mu_2)(X_1-\mu_1)]}{\sigma(X_2)\sigma(X_1)} & 1 & ... & \frac{E[(X_2-\mu_2)(X_n-\mu_n)]}{\sigma(X_2)\sigma(X_n)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{E[(X_n-\mu_n)(X_1-\mu_1)]}{\sigma(X_n)\sigma(X_1)} & \frac{E[(X_n-\mu_n)(X_2-\mu_2)]}{\sigma(X_n)\sigma(X_2)} & ... & 1 \end{bmatrix} \tag{4.1}$$

Therefore, the data received for each country had different lengths. In order to avoid such a problem in model training we used the *padding* method so to bring all kind of data at a common length, in fact equivalent to that of China's data. The padding was achieved by adding zeros before the initiate date of the pandemic for each country so it's data to obtain the desired length.

Furthermore, there were plenty of occasions where the data recording was incorrect. For instance, observed steep increases in the daily cases curve due to recording of past "forgotten" confirmed cases or simply soar rise owing to a public holiday the very previous day. In such cases the specific part of data was substituted by the 7-day rolling average $\pm c$ in order to retain somewhat the observed surge (or plunge). We recall that the 7-day rolling average for an observation X of length N is given by:

$$M_i = \frac{X_{i-3} + X_{i-2} + X_{i-1} + X_i + X_{i+1} + X_{i+2} + X_{i+3}}{7} \quad \text{for} \quad 3 \leq i \leq N-3 \tag{4.2}$$

For $i = 0, 1, 2$ and for $i = N-2, N-1, N$ we get $M_i = 0$.

Neural networks are very specialized models in their functionality. Their input must be a structure of 3 dimensions (we call it *tensor*) while our collected data is a 2-dimensional array ($days \times features$). In order to transform the data to $3D$ we used the *sliding window* method. According to this method, we choose a window of length l_{window} that slides through the entire dataset and creates sub-matrices ($l_{window} \times features$) in size. Eventually, the tensor that will have been formed will be of dimension ($days - l_{window}, l_{window}, features$). At the same time, we used as label next day's observation.

Moreover Neural networks are capable of handling data that do not show large deviation. So it was necessary to normalize the data (which often was on the scale of 10^6) down to far smaller values. To that end, we used the *MinMax scaler* with which all data values are brought down to $[-1, 1]$. Mathematically, for collected data X with minimum and maximum values X_{min} and X_{max} respectively the formula is:

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}}, \tag{4.3}$$

It is worth mentioning that 90% of data was for training, 10% for testing and 10% for validation (from the training part).

Throughout the project, except from Univariate and Multivariate Forecasting we used 2 other methods of forecasting as well. These were the *Labelling method* and the *method of Differencing*. The former is a method that was proposed to deal with possible extremes outliers in the time series during forecasting. An extra feature in boolean form (true/false or 0/1) was added in the data, if the current observation exhibits greater than or equal divergence by a constant value, $thres$, from some observation k time steps in the past. Usually, we choose $k = 1$. In this way the networks, possibly, hold up better the information in the existence of an extreme outlier. We model this procedure by the following formula:

$$F_{label}(t) = \begin{cases} 1, & |x(t) - x(t-k)| \geq thres \\ 0, & |x(t) - x(t-k)| < thres \end{cases} \tag{4.4}$$

The second method, that of Differencing, is applied subsequently well at non-stationary curves that showcase apparent trend and period (T). We aim to obtain a new curve, let it be Z_{diff}, by conducting constant subtractions between the data that abstain T time steps. The Z_{diff} curve shows greater stagnation but we "sacrifice" a period T from the initial observation X_{ob}. The mathematical expression is as follows:

$$Z_{diff}(t) = X_{ob}(t) - X_{ob}(t-T), \quad t \geq T. \tag{4.5}$$

For the evaluation of the results, a reverse transformation is needed in order to return the data in their original scale. This can be described as below ($Z_{diff-pred}(t)$ is the prediction of the model on Z_{diff} data):

$$Z_{inv}(t) = Z_{diff-pred}(t) + X_{ob}(t-T), \quad t \geq T. \tag{4.6}$$

In case where the data are strongly non-stationary we can apply the Differencing method once more as:

$$\begin{aligned} Z_{2-diff}(t) &= Z_{diff}(t) - Z_{diff}(t-T) \\ &= X_{ob}(t) - X_{ob}(t-T) - X_{ob}(t-T) + X_{ob}(t-2T) \\ &= X_{ob}(t) - 2X_{ob}(t-T) + X_{ob}(t-2T) \end{aligned} \tag{4.7}$$

The selection of the period T for applying the Differencing method is based on the maximization (with respect to τ) of the function autocorrelation, $\mathbf{E}[X_t, X_{t+\tau}]$ of the initial time series X_t, with the exception of $\tau = 0$. For the COVID-19 data, specifically, due to the periodically conducted tests we used mostly $T = 7$. In Figure 5 we can see an example of the autocorrelation of an observation X_t and the much lower autocorrelation of the resulting curve Z_{diff} after Differencing.

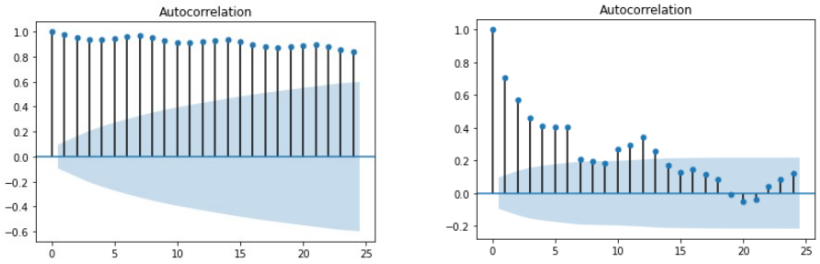


Figure 5: Autocorrelation of X_t curve (left) and Z_{diff} curve (right) after Differencing

Coming to the models and parameters choice, we selected as length of time step, $l_{window} \in \{7, 21\}$ and thus we allowed our model to "look back to the past" 1 to 3 weeks during training.

The entire dataset was of length 429 days and in keeping with the analogies of 90%-10% as we mentioned, we resulted in 386 days for training and 43 for testing (approximately 6 weeks). As shown previously the final form of data was $3D$ in shape: $(days, timestep, features)$. The models employed for these experiments were 9 in total: 1) Simple RNN, 2) Simple GRU, 3) Simple LSTM, 4) Stacked LSTM (2 layers), 5) Bidirectional GRU, 6) CNN-RNN, 7) Convolutional LSTM[1], 8) TCN[1] and 9) Attention based. At each network there was a great variety of hyperparameters to choose, such as: Number of neurons, Dropout probability, Kernel size, Number of epochs etc.

5 Experiments and Evaluation

5.1 Evaluation Metrics

We evaluated our models according to the error they made in each prediction. There are two kinds of errors: Root Mean Squared Percentage Error (RMSPE - %normalization of RMSE) and Mean Absolute Percentage Error (MAPE - %normalization of MAE). The definition of those are given by the following equations:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_{pred}^{(i)} - y_{obs}^{(i)}\right)^2} \Rightarrow RMSPE = \left(\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_{pred}^{(i)} - y_{obs}^{(i)}}{y_{obs}^{(i)}}\right)^2}\right) \times 100\% \quad (5.1)$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_{pred}^{(i)} - y_{obs}^{(i)}\right| \Rightarrow MAPE = \left(\frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_{pred}^{(i)} - y_{obs}^{(i)}}{y_{obs}^{(i)}}\right|\right) \times 100\% \quad (5.2)$$

Subtracting these errors from 100%, two respective scores of efficacy (A_{ab}, A_{sq}) can be derived as:

$$A_{ab} = Absolute\ Accuracy = 100\% - MAPE, \quad (5.3)$$

$$A_{sq} = Squared\ Accuracy = 100\% - RMSPE \quad (5.4)$$

Another accuracy metric is the so called *determination coefficient* or *"R-squared"*, which indicates us the cross-correlation of regression model's forecasts with independent variable (observations), on a scale of 0-100%. Valuer close to 1 show perfect prediction, whereas values close to 0 accent weak forecasting. We formulate this as:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \quad (5.5)$$

where,

$$SS_{tot} = \sum_i \left(y_{obs}^{(i)} - \bar{y_{obs}}\right)^2 \quad (5.6)$$

$$SS_{res} = \sum_i \left(y_{obs}^{(i)} - y_{pred}^{(i)}\right)^2 \quad (5.7)$$

In all of equations (5.1), (5.2), (5.6) and (5.7), y_{obs} and y_{pred} indicate the vectors of observations and predictions respectively whilst n is their common length. In equation (5.6), $\bar{y_{obs}}$ is the mean value of observation vector.

[1]Search the official Thesis for detailed description

5.2 Experiments

5.2.1 General Experimentation

Every model's score eventuated by averaging 5 evaluations ±standard deviation in each of 4 following different methods of prediction: 1)Univariate Prediction, 2)Multivariate Prediction, 3) Labelling Method and 4) Differencing Method.

In our first experimentation we tried to predict the upcoming COVID-19 cases on global scale using the four aforementioned methods and the nine previously reported models. It should be noted that during the Multivariate Prediction we used as an ancillary feature the daily conducted tests while during the Differencing Method we used a period of $T = 7$ due to the strongly non-stationary curve. In Figure 6 we depict the best score that each model achieved at any of the four methods[2]. In Figure 7 we show the head-to-head result of the most accurate model, Attention with Differencing, against the actual cases of the predicted period. In that figure the x-axis represents the respective calendar day.

In our second experimentation we tested our models in a curve that is related to deaths and more precisely at Italy's daily deaths curve. This curve is more stationary than the one in cases experimentation, nevertheless in Differencing method we uses a period of $T = 7$ once again. Moreover, in Multivariate Forecasting the daily numbers of patients in Italy's hospitals and ICUs were used. The best score that each model accomplished at any of the four methods can be seen in Figure 8, while in Figure 9 the head-to-head result of the most accurate model, TCN with Univariate, is depicted[3]. Again in x-axis we can see the calendar day.

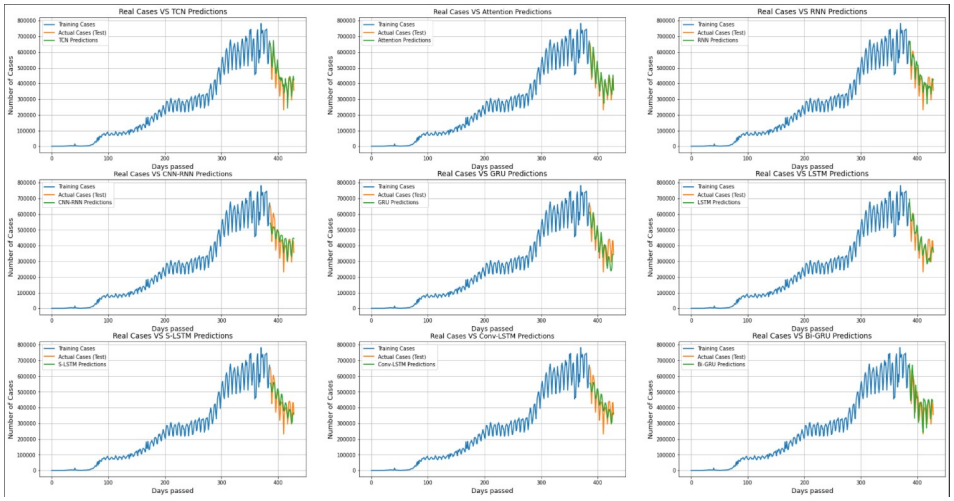


Figure 6: Graphical representation of best results of all models on global cases curve

[2]For the analytical tables of results of this experiment search the official Thesis
[3]For the analytical tables of results of this experiment search the official Thesis

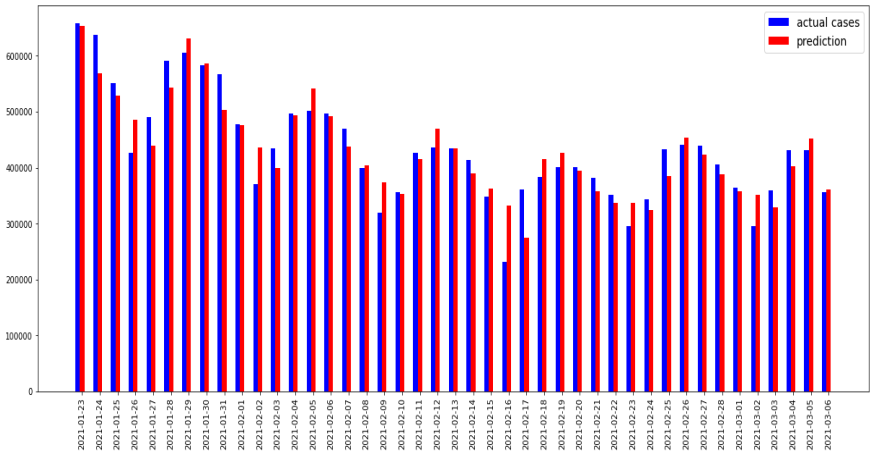


Figure 7: Comparison of actual global cases with the best model, Attention with Differencing

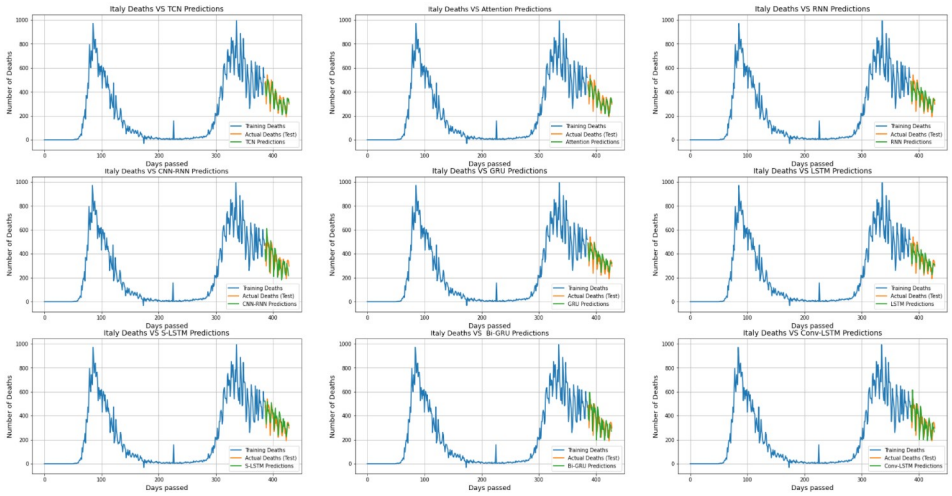


Figure 8: Graphical representation of best results of all models on Italy's deaths curve

5.2.2 Rolling Update Technique

A particularly distinctive technique for achieving prediction beyond the time horizon of dataset is the *Rolling Update Technique* or *Feedback Technique*. We can describe this method with the following steps:

- The entire dataset is used as training data in order to provide as much information as possible on the model.
- The last sub-table of training data used for the 1^{st} forecast beyond the time horizon of the dataset $\Rightarrow$ Concatenate and slide the model's output with the last sub-table of training data $\Rightarrow$ Feedback at the model input $\Rightarrow$ Use of the new sub-table to predict the 2^{nd} value beyond the limit and so on.
- In multivariate prediction we are forced in predicting all features.

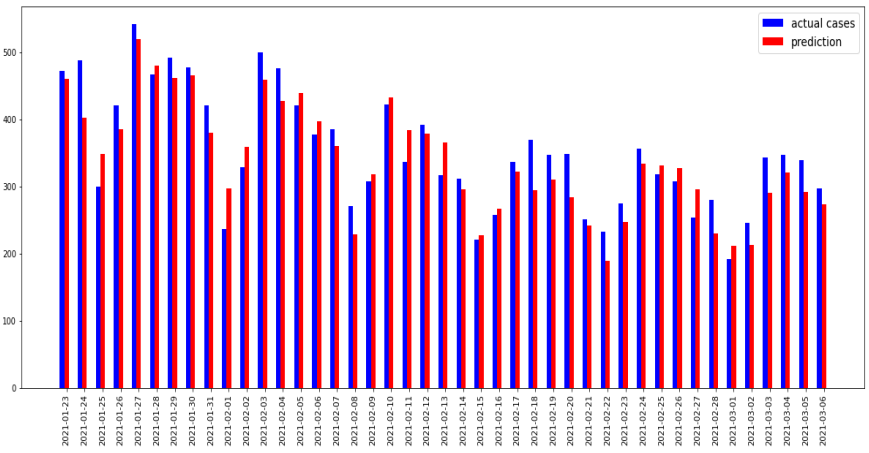


Figure 9: Comparison of actual Italy's deaths with the best model, TCN with Univariate

- As we move forward in the long run, forecasts will tend to stabilize.
- Avoid using metrics of evaluation due to large deviations and absence of real testing data. However, we collected data from later dates for comparison.
- We used only 6 models: LSTM, Attention based, Bidirectional GRU, Convolutional LSTM, GRU and TCN and at the same time we proceeded for a time horizon $T_f = 30$ days.

Figure 10 presents a model that uses the Rolling Update Technique. It's about an LSTM but in fact, it could be any of the previously described models. We tested that experimentation in two daily cases curves, those of USA and Greece. In Figures 11a and 11b we illustrate the results that were closest to the actual evolution of the pandemic in the 2 examined countries for the next 30 days. In both figures these results evolve from the Attention based model, while in x-axis we can see the calendar day once more[4].

[4]For graphical comparison search the official Thesis

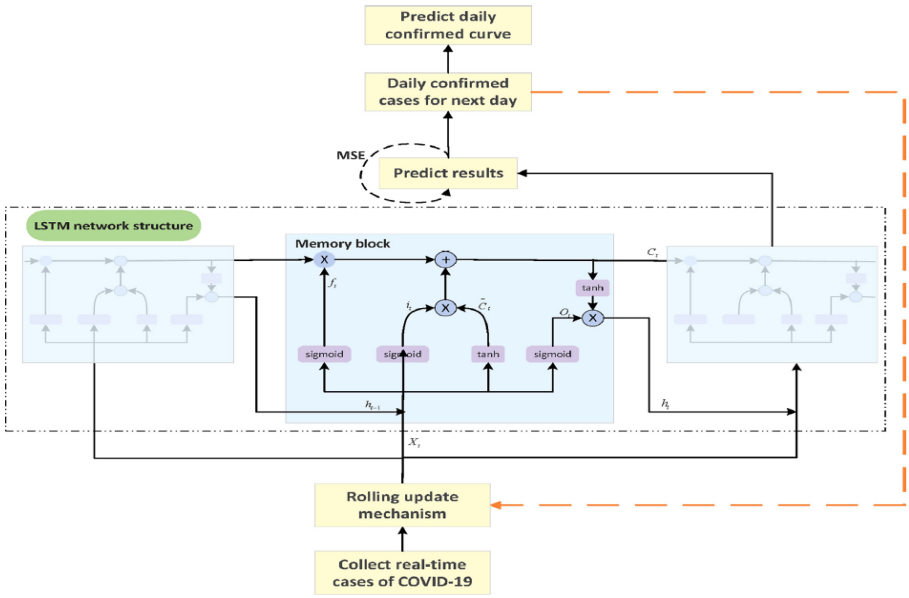
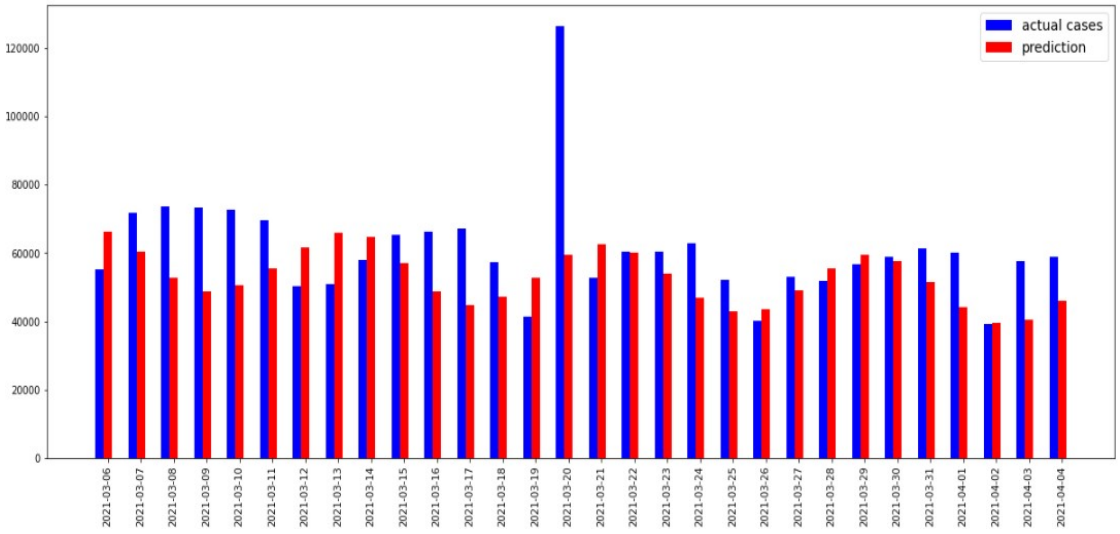
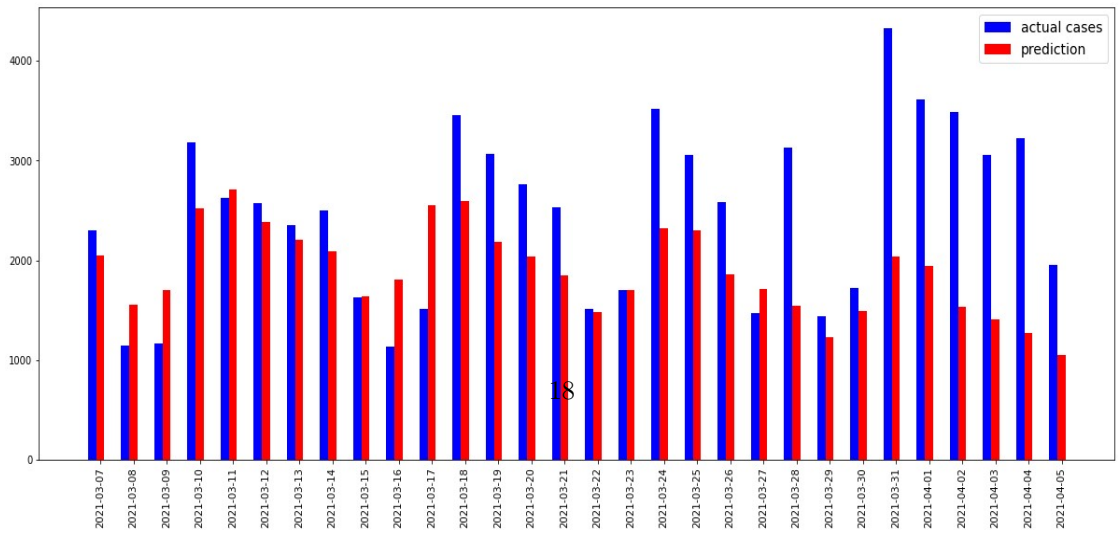


Figure 10: Comparison of actual global cases with the best model, Attention with Differencing



(a) In United States of America



(b) In Greece

5.2.3 Anomaly Detection

Sometimes, in data processing, it is important to search where our data seem to have abnormal behaviour. This behaviour can be described as an abrupt rise or decrease of the observed variable. In order to catch these steep variations in Data Science we used *Anomaly Detection*, a method in which we indicate that a specific part of our data appears to be extreme. In machine learning and in this work we use *Autoencoders* so to detect any anomalies in COVID-19 data. Autoencoders are models that aim to retread the initial information through encoding and decoding. Mathematically, given an input X, an Encoder's action ϕ and a Decoder's action ψ, an autoencoder can be delineated as:

$$\phi : X \to F(X) \tag{5.8}$$

$$\psi : F(X) \to X \tag{5.9}$$

$$\phi, \psi = \underset{\phi,\psi}{\operatorname{argmin}} ||X - \psi(\phi(X)|| \tag{5.10}$$

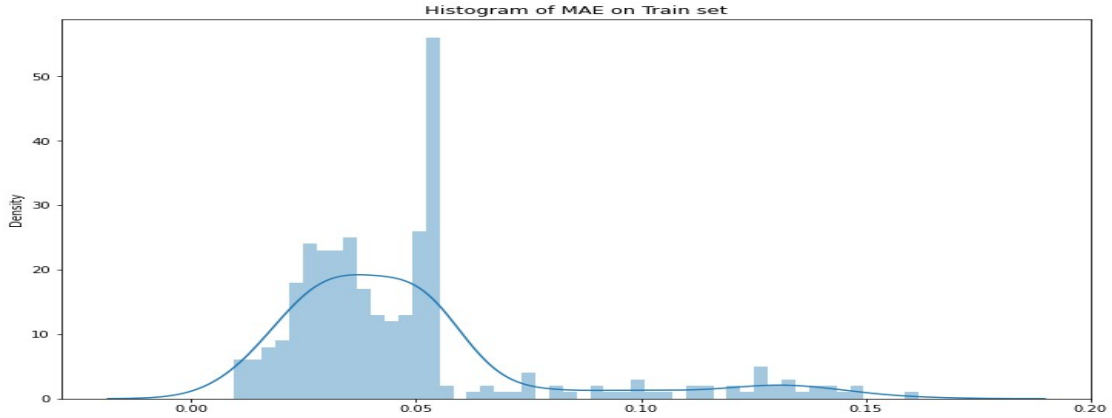
We used 3 kind of Autoencoders: LSTM, GRU and TCN. The following method was used in order to track any anomalies:

- Suppose a train set X_{train} and a test set X_{test} in proportion 80%-20% of the total data.
- Train the model on X_{train} data and evaluate it in known data X_{train}.
- Receive the Absolute Error between prediction ($X_{train-pred}$) and real observation (X_{train}) as: $E_{train} = X|_{train} - X_{train-pred}|$.
- Receive the maximum value of E_{train} ($max\{E_{train}\}$) as a threshold (Th), over which, we consider anomaly any prediction of the model.
- Evaluation of the model in unknown data (X_{test}).
- Receive, once again, the Absolute Error between prediction ($X_{test-pred}$) and real test set (X_{test}), as: $E_{test} = X|_{test} - X_{test-pred}|$.
- For whatever value of E_{test} is valid: $E^i_{test} > Th$, then data i is characterised as anomaly.

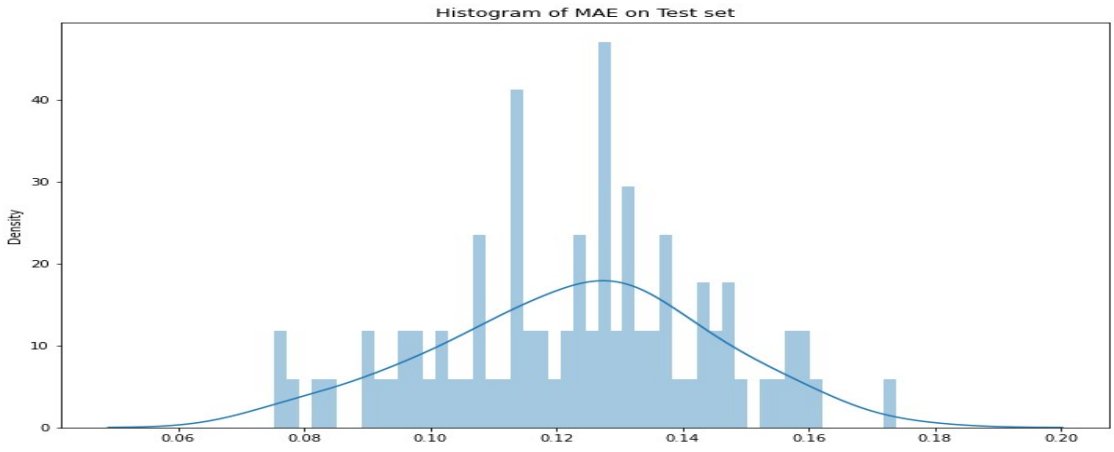
We conducted 3 experiments: on global and Italy's daily cases and on Greece's deaths with thresholds that emerged as (respectively): $Th = 0.15, Th = 0.25$ and $Th = 0.3$. We depict the above described method graphically in Figures 12a, 12b and 12c[5].

5.2.4 Extracting Geographical Features

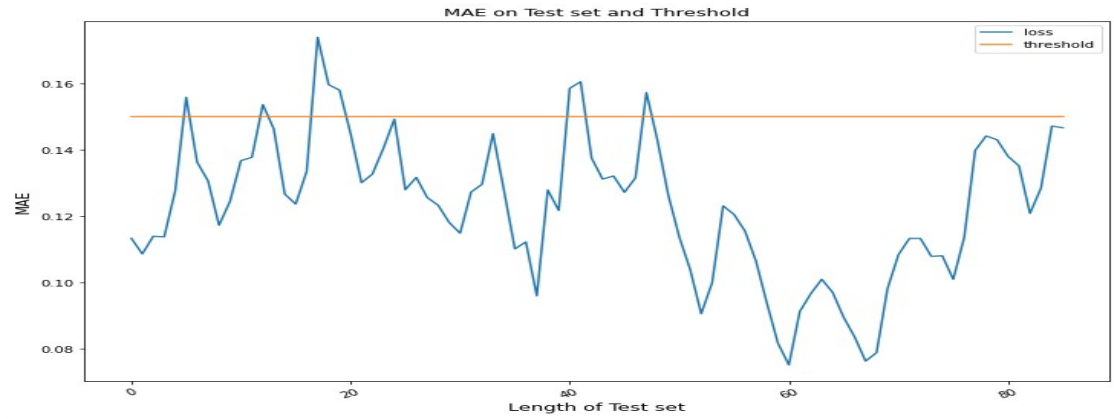
In the last section of this project we tried extract similarities in the evolution of COVID-19 disease. We investigated if these similarities are linked with the geographic and socioeconomic position of the countries. In order to achieve such a purpose we decided to use a set of countries, **A** large in number and to train different type of models in the epidemiological data (daily cases) of those. Subsequently, we chose another set of countries, **B**, with fewer countries than **A**, so to test the models in their corresponding epidemiological data. Observing the geographical heatmaps of Section 2, we inferred that Europe along with USA suffered more from the pandemic in contrast to Africa and Asia (assuming that all countries provided equally reliable data). Furthermore, we saw that countries of Europe had similar ways of confronting the pandemic (testing civilians for the virus, impose of restriction measures etc.) unlike nations of Asia like India or China. Based on these annotations the 2 conducted experiments had the following structure:



(a) Histogram of the absolute error from Train set



(b) Histogram of the absolute error from Test set



(c) Absolute error with threshold $Th = 0.15$

Figure 12: Anomaly detection with autoencoders

1^{st} Experiment	
Set **A** *(Train set)*	*Set* **B** *(Test set)*
Austria, Belgium, Bulgaria, Canada, Denmark, France, Iceland, Ireland, Italy, Luxembourg, Netherlands, Portugal, Slovenia, Spain, United Kingdom, United States	Greece, Cyprus

Table 1: Selection of countries for training and evaluation with (relatively) common characteristics

2^{nd} Experiment	
Set **A** *(Train set)*	*Set* **B** *(Test set)*
Algeria, Angola, Bangladesh, China, Democratic Republic of Congo, Egypt, Ethiopia, India, Iran, Kazakhstan, Mongolia, Pakistan, Russia, Saudi Arabia, Sudan, Turkey	Greece, Cyprus

Table 2: Selection of countries for training and evaluation with (theoretically) dissimilar characteristics

As seen above in Tables 1 and 2 the two experiments had the same Test countries: Greece and Cyprus. As far as the Train sets is concerned it is obvious that in 1^{st} experiment we have a group of countries from Europe & North America while in 2^{nd} experiment we nations that originate from Africa & Asia. Our goal is models to learn and depend on the development of daily cases curve of these 2 clusters in order to predict the respective curve of Test set states. Common logic indicates that, since Greece and Cyprus are European countries, more similarities will arise during the training 1^{st} Experiment. The models we used during these experimentations were LSTM, GRU, TCN and Attention based, while the metrics of evaluation were A_{ab} and A_{sq} given respectively from equations (5.3) and (5.4). The results from those experiments are demonstrated in Tables 3 and 4[6]. Note that due to the following equation:

$$MAE \leq RMSE \leq \sqrt{n}MAE \tag{5.11}$$

the $RMSPE$ values can be far greater than $MAPE$ values, especially when the latter gets values big enough (i.e. 50%) the $RMSPE$'s values can exceed 100%. As a result, we observe some negative percentages in the respective metric of efficacy (A_{sq}).

[5]For the analytical results search the official Thesis
[6]For graphical comparison search the official Thesis

Country	Model	$\mathbf{A_{ab}}$ (%)	$\mathbf{A_{sq}}$ (%)
Cyprus	Attention	**56.42**	**-3.47**
	TCN	52.68	-6.83
	LSTM	48.31	-12.97
	GRU	49.02	-18.62
Greece	Attention	**61.24**	**20.13**
	TCN	48.99	-11.92
	LSTM	56.68	8.49
	GRU	54.89	-3.99

Table 3: Evaluation of the models in the experiment of Table 1

Country	Model	$\mathbf{A_{ab}}$ (%)	$\mathbf{A_{sq}}$ (%)
Cyprus	Attention	**50.16**	**0.49**
	TCN	45.32	-9.73
	LSTM	42.94	-17.75
	GRU	42.44	-15.7
Greece	Attention	**56.14**	4.97
	TCN	51.79	**5.71**
	LSTM	49.15	-8.58
	GRU	45.31	-23.82

Table 4: Evaluation of the models in the experiment of Table 2

Coming to the evaluation of the results from Tables 3 and 4 we can highlight that:

- The Attention model is by far the most reliable in delivering the total epidemiological curve of countries having been trained in data from other countries.
- On average, the accuracy rates of A_{ab} from values $\geq$50% in Experiment 1 dropped to percentages of $\leq$50% in Experiment 2, thus having a rough decrease of 5% to 10%.
- Most importantly, through this process we verified, through the effectiveness of the models, our initial hypothesis that in some parts of the world the picture of pandemic evolution is similar to other parts of the world, while for other parts of the globe is different.

6 Conclusions and Future expansion

Through our general experimentations we inferred the following:

- The usage of TCN model for Univariate forecasting.
- The usage of the Attention based model for non-stationary curves.
- The use of the Differencing method also, for curves that showcase a specific trend and period.
- The use of "classic" RNNs (any kind) in more stationary curves.
- Usage of Multivariate technique only when the features are strongly correlated.

- The use of the proposed Labeling technique after investigation.

We deem that the current work can be expanded in two directions. The first direction, is affiliated with extensions related to variety of data used. For example, during the last weeks of the project the emerging vaccinations provided a reliable variable that is strongly (and reversely) correlated to the cases and deaths. Furthermore, from our own experience we know that the dispersal of the virus depends on metrics such as the temperature of a place or the motility of population. Variables that model these non-arithmetic data could be used in order to improve our prediction. In the same way, for deaths we could use indicators that show the aging of general population or the capabilities of the National Healt System of a country.

The second direction concerns the deployment of more complex models. We have to suggest alternative ways of using the Attention mechanism as an extension to our employed Attention based model. Moreover, the construction of the so-called *Transformers* models and the incorporation of them in our problem of prediction time series is a true challenge. These models are extremely powerful and use in an appropriate way the Fully Connected Layers, the Residual Connections and the Attention Mechanism. During the last years the are dominating in Natural Language Processing (NLP) and Neural Machine Translation (NMT) fields. A view of such a model is given in Figure 13 from NLP sector.

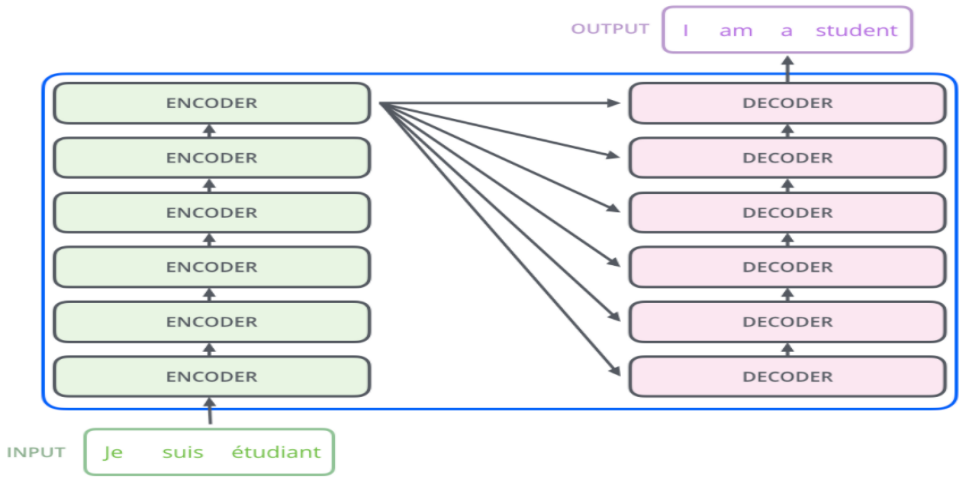


Figure 13: Representation of a Transformer's architecture