COVID 19: Forecasting Project

Team members: Guillem Amat, Sebastián Soriano Pérez

Abstract

Covid-19 has affected over 170 countries in the world, changing the ways in which we interact with each other and having a profound and long-lasting societal and economic impact. It has become increasingly important to be able to forecast cases and fatalities in order to fend off the pandemic. This report looks at applications of well known deep learning models to predict Covid-19 cases at a global and local level. This study also discusses challenges in the application of these models and proposes ideas to enhance their predictive capacity.

1 Introduction

The Covid-19 pandemic is the main global health crisis of the 21st century and the biggest challenge humanity has faced since the end of the 2nd World War.[1]

The virus comes from a family of single-stranded RNA organisms called coronaviruses that measure 120 nanometers in diameter and that have a characteristic circular shape with "crown-like" spikes in their surface. These viruses are susceptible to mutations and recombinations, resulting in a highly diverse pool. They mostly reside in bats but can easily be transmitted to other mammals like humans.[2]

It is thought that the virus originated in a seafood wholesale market in the Wuhan province of China. The virus quickly spread across China and further expanded across the world. Countries like Italy, Spain, Mexico or the United States were particularly impacted soon after the beginning of the pandemic and continue to be so as of the date of writing.[5] The World Health Organization officially declared the virus a pandemic when 120.000 confirmed cases were reached worldwide, bringing the world into a halt and causing an immense loss of life.[4] The virus has also caused the most severe recession since the Great Depression and had an immense societal impact.[3]

Most people infected with the virus will experience mild symptoms, mainly, fever, chills and dry throat, and they will recover without the need of medical assistance. Older people and groups with chronic diseases such as diabetes are at an increased risk of having complications. Nevertheless the virus is very unpredictable and people that were apparently healthy have died.[6]

1.1 Motivation and importance of the problem

In this environment it has become increasingly critical to accurately detect and forecast Covid-19 cases. Specifically Covid-19 forecasting is useful to:

- Inform public health decision making in the implementation of measures such as the mandatory use of masks or regional and country-wide lockdowns. It can help policy makers decide between mitigation or containment strategies.[7]
- Help anticipate medical supply needs, including ventilators, hospital beds and intensive care units (ICU). It can also help forecast the need of qualified medical personnel and the timing when patient surges will happen.[8]
- Help the business community better predict their talent, budget and equipment needs.

Covid-19 is a notoriously difficult virus to predict given the lack of population tracking datasets, the relatively large incubation period and the lack of enough samples to accurately infer trends. The

following section gives context to the issue at hand and discusses studies that have contributed to the body of knowledge.[9]

2 Related works

The problem of Covid-19 forecasting belongs to the field of time-series. A time-series is a sequence of n data points indexed by time t.

Traditional methods such as AR models and ARIMA can incorporate information about trends and seasonality in the data. However, they suffer from many shortcomings, such as assuming linearity, the need for a complete dataset, the inability to deal with more than one variable (without performing significant modifications), and they suffer when making long-term forecasts.[10]

More recent and powerful Neural Network models can help overcome these weaknesses. In particular Recurrent Neural Networks and their variants are a specific instance of this class of models that uses past information to make future predictions. This makes them particularly well suited for time-series analysis. Neural Network models have been used across fields consistently outperforming the previous existing benchmarks:

- In Finance, Neural Networks have been extensively used in stock price prediction as well as to forecast other financial variables, such as Volatility.[11]
- In the Social Sciences, they have been used to predict population indicators such as the birth rate or population growth.[12]
- In Economics, these models have been used to predict and forecast targets like GDP growth, Consumer Price Indexes, Unemployment Rates or Inflation.[13]

The field of Epidemiology deals with the incidence, distribution and possible control of diseases. Since the beginning of the pandemic, there have been many attempts to use Deep Learning to forecast global and local Covid-19 cases and fatalities. Two notable examples are included below:

- A study between the United States and India, comparing different LSTM architectures, found that the best performing model was a Convolutional LSTM. Not only was the Convolutional LSTM trained faster, but it also offered consistently higher results when measuring Loss with MAPE.[14]
- A study of RNNs, including 10 different types of LSTMs with simplifications, was carried in the United States. The results found that the simplified LSTMs were giving similar results to a regular LSTM. The models were able to be trained faster, but they did not solve the prevalent case underestimation problems shown by other models.[15]

In this report we aim to try different Neural Network architectures to predict global and local Covid-19 cases. The following sections deal with the process, models used and results.

3 Details of the project

3.1 Data

We used the Covid-19 Global Forecasting dataset provided by Kaggle. The dataset contains 7 features: County, Region, Country, Population, Weight, Confirmed Cases and Fatalities.

We developed two different pipelines to predict Covid-19 cases at a global and local level. The global pipeline consisted of pivoting the data, reshaping it, scaling it to values between 0 and 1 and consolidating it by adding up all Confirmed Cases and Fatalities by Date. A final step is required when processing time-series: The data must be divided into sequences of fixed length. We tested sequences of length 5, 10, 15 and 25. We settled down for the latter given superior predictions for longer sequences, even at a cost of fewer data points.

The local pipeline had many similarities with the global one, but it also contained a few important differences. The data was initially filtered to only include five European countries that had similar Confirmed Cases and Fatalities curves. Locations were geocoded by adding Latitude and Longitude.

Population was added as an extra value. The data was then reshaped and scaled to have values between 0 and 1. Finally the dataset was divided into sequences of length 25.

Note: The data contained many irregularities, such as instances where Confirmed Cases were negative when a country made a reporting mistake. We considered applying a smoothing factor or capping values to the data. At the end we decided to leave the data as it was and let the Neural Network uncover hidden trends. In a future analysis we would strongly consider adding the discussed methods to our pipeline.

3.2 Models

We tested and fitted the data to an instance of the following model architectures: Artificial Neural Networks, Convolutional Neural Networks and two different Recurrent Neural Networks; the Gated Recurrent Unit and the Long-Short Term Memory. The models are specified in detail below:

3.2.1 Artificial Neural Network

The first model we tested was an Artificial Neural Network (ANN) so we could have a baseline model to compare the rest of our results to. An ANN consists of an input layer which connects to a series of interconnected hidden layers and a final output layer. Each of the layers has a certain number of nodes or 'neurons' that connect to the nodes in the next layer. During the process of forward propagation, each node receives a signal (one of the input variables $\mathbf{x}$ for the input layer or the result of the multiplication of some weights $\mathbf{W}^{(k)}$ and the previous layer's outputs $\mathbf{z}^{(k)}$: $\mathbf{a}^{(k)} = \mathbf{W}^{(k)}\mathbf{z}^{(k)}$ for the hidden layers) and process the signal through an activation function to create an output $\mathbf{z}^{(k)} = f(\mathbf{a}^{(k)})$. The output layer's output is the model's prediction for the target variable $\hat{\mathbf{y}}$.

The model's weights $\mathbf{W}^{(k)}$ at every layer k are the parameters that are optimized during training during the backpropagation process using gradient descent to achieve better predictions on the training data while minimizing the loss function being used. The loss function is a method of measuring how close a model's predictions are to the actual target values in the training dataset. An Artificial Neural Network can be very flexible in predicting or modeling target variables, especially when their architecture has enough depth and width on its hidden layers.

For our first model we used an architecture of an input layer with a number of nodes equal to the number of entries in a sequence (25) times the number of variables being considered (either 2 for the aggregated global data or 5 for the regional or local data), 10 hidden layers with 500 nodes each, and one output layer with 2 nodes for predicting both 'confirmed cases' and 'fatalities'. Each of the hidden layers uses a rectified linear unit activation function (ReLU). In order to process the time series data with either 2 (global case) or 5 (regional or local case) predictors, we flattened all the predictors' time series into a single vector of length 25 times 2 (global) or 25 times 5 (regional or local), so it could be processed into the ANN.

3.2.2 Convolutional Neural Network

A Convolutional Neural Network (CNN) also consists of an input and an output layers, and a number of hidden layers in between. However, the hidden layers include a series of convolutional layers in which the layer's input values (the previous layer's outputs) are processed with a kernel that multiplies its weights to a set number of input values (the kernel size) during a series of convolutions that run through all of the input values generating the layer's activations. This process is useful to identify patterns and 'shapes' in the data in local regions, and the deeper layers of the CNN are capable of identifying more complex patterns.

CNN's are commonly used in the field of computer vision and image classification where two-dimensional convolutional layers and kernels are used to identify patterns in pixels that are close together. For time series models, one-dimensional convolutional layers and kernels can 'convolve' through the data in a similar fashion, but in only one dimension. When dealing with multivariate problems as ours, each predictor has its own set of 25 values in a time series sequence. A 1-D convolutional layer can process each predictor as a separate channel in a manner that is analogous to how a 2-D convolutional layer processes different channels in an image that has all three RGB channels.

Our CNN model's architecture consist of an input layer with 2 (global case) or 5 (regional or local case) channels and 25 nodes each, a hidden one-dimensional convolutional layer with 64 channels

and a kernel size of 1, a hidden linear layer with 50 nodes and a ReLU activation function, and an output layer with 2 nodes for predicting both 'confirmed cases' and 'fatalities'.

3.2.3 Gated Recurrent Unit

One of the main problems surrounding plain-vanilla Neural Networks is that long matrix products tend to result in vanishing and exploding gradients. The presence of important elements at the beginning of a sequence, the fact that some inputs to the network do not carry any information or the possibility that there is a logic breakdown in a sequence, are elements that a Recurrent Neural Network is not designed to handle well.

The Gated Recurrent Unit is a special type of RNN that supports gating hidden states, which help mitigate the aforementioned problems.

There are two critical parts to a GRU. The first one is a reset gate, which regulates the amount of information a previous state gives to the current one. The second one is an update gate that gives information of how much of the new state is a copy of the old one. These two components are combined in different ways to create a final Hidden State and an Output.The formulas to calculate the different parts of a GRU are given below in order of computation:[16]

$$
\begin{aligned}
Z_t &= \sigma(X_t W_{xz} + H_{t-1} W_{hz} + b_z) \\
R_t &= \sigma(X_t W_{xr} + H_{t-1} W_{hr} + b_r) \\
\tilde{H}_t &= tanh(X_t W_{xh} + (R_t \circ H_{t-1}) W_{hh} + b_h) \\
H_t &= Z_t \circ H_{t-1} + (1 - Z_t) \circ \tilde{H}_t
\end{aligned}
$$

Where R_t is the Reset Gate, Z_t is the Update Gate, $\tilde{H}$ is the Candidate Hidden State and H_t is the Hidden State.

3.2.4 Long-Short Term Memory

An LSTM can be seen as an enhanced version of a GRU. LSTMs are composed of a memory cell that is governed by a system of gates: The output gate, the forget gate and the input gate. The output gate reads the entries from a cell. The input gate discerns whether to read data into the cell or not. The forget gate resets the content of a cell. This system of gates allows the LSTM to choose whether to remember or forget data. In a similar fashion to a GRU, the output of these gates is combined to create a Hidden State and an Output. The formulas to calculate the different parts of the memory cell are outlined below:[17]

$$
\begin{aligned}
I_t &= \sigma(X_t W_{xi} + H_{t-1} W_{hi} + b_i) \\
F_t &= \sigma(X_t W_{xf} + H_{t-1} W_{hf} + b_f) \\
O_t &= \sigma(X_t W_{xo} + H_{t-1} W_{ho} + b_o) \\
\tilde{C}_t &= tanh(X_t W_{xc} + H_{t-1} W_{hc} + b_c) \\
C_t &= F_t \circ C_{t-1} + I_t \circ \tilde{C}_t \\
H_t &= O_t \circ tanh(C_t)
\end{aligned}
$$

Where I_t is the Input Gate, F_t is the Forget Gate, O_t is the Output Gate and $\tilde{C}_t$ is the Candidate Memory Cell. These 4 gates can all be calculated in parallel. The Candidate Memory Cell is then combined with the Input and Forget Gates to produce C_t, the Memory Cell. Finally the Memory Cell is combined with the Output Gate via the Hadamard Product to produce H_t, the Hidden State.

3.3 Contribution of each member of the team

Guillem Amat: Contributed by creating the preprocessing pipeline which included geocoding, sequence creation and data wrangling. Created the Training Pipeline which was used by the different models. Created and visualized predictions of Gated Recurrent Units and Long-Short Term Memory. Developed and worked on all sections of the report and the final presentation.

Sebastián Soriano Pérez: Contributed to cleaning and generating high quality reusable code. Improved and optimized the preprocessing and training Pipelines. Researched on time-series analysis and contributed to all sections of the report. Created and visualized predictions of a Convolutional Neural Network for time-series analysis and an Artificial Neural Network.

4 Experimental results

The results are very surprising. We expected the GRU and the LSTM to outperform the CNN and the ANN. The latter were able to correctly identify trends, train significantly faster and give a much lower RMSE loss value. A summary of the results can be found in the table below:

Model Results				
Neural Network	Global Train Loss	Global Test Loss	Local Train Loss	Local Test Loss
ANN	0.01	1.01	0.20	0.13
CNN	0.11	1.29	0.20	0.13
GRU	11.56	4.51	0.98	0.07
LSTM	0.26	6.45	0.96	0.13

We are not able to pinpoint the underlying reasons driving the superior performance of the ANN and CNN. We tried different LSTM and GRUs hyperparameter configurations by gradually augmenting the hidden layer sizes, providing longer sequence lengths and lowering weight decay and dropout values. While their loss dropped and they found a local minimum, the GRU and the LSTM were never able to outperform the other two. The LSTM was able to overfit the train data given enough iterations, but it performed poorly on the test set.

5 Concluding remarks

The results were very surprising, we did not expect the non-recurrent Neural Networks to perform as well as they did. Given that almost all the literature we reviewed included LSTMs, we believed this would be the best performing model. Nevertheless, we would like to explore training the two special RNNs with a larger number of hidden units or implement a more complex architecture. There are multiple advanced LSTM and GRU architectures such as the following:

- Time-Aware LSTM: A particular implementation of LSTM designed to handle irregular elapsed times that is able to capture temporal dynamics well. We believe this is a component that was lacking for our LSTM and that the data we are dealing with requires.[18]
- Encoder-Decoder GRU: An enhanced version of the GRU that gives better results in seq2seq and Natural Language Processing tasks. It has been used to successfully identify trends across different instances of a time-series, which is similar to what we want to achieve with the local pipeline.[19]
- GRU-ODE-Bayes: A GRU fitted by using Ordinary Differential Equations from a Bayesian perspective. This would allow us to introduce a prior.[20]

Another area of improvement could come in the preprocessing stage of the data. We believe that, by clipping values and smoothing them, we would be able to help the special RNNs achieve better predictions. There are also many resources online that we could use to give extra information to our models:

- The New York Times has a public repository with data from the pandemic at the state and county level for the US. Among the many things in the repository, they publish the results of public opinion polls on the use of face masks.[21]
- The Center for Disease Control has data on the number of tests administered through time as well as ventilator capacity by date in the United States.[22]
- The European Center for Disease Prevention and Control has an extended dataset of all the countries in the world which includes 160 more days than the the dataset that we used.[23]

Perhaps this does not belong on a report of this kind, but we really enjoyed this project. It was the first time working with a time-series for us, so everything was new. If given the opportunity, we would like to extend and improve the analysis with the ideas above. Thanks for reading!

6 References

1. `https://www.undp.org/content/undp/en/home/coronavirus.html`

2. https://www.cebm.net/covid-19/coronaviruses-a-general-introduction/
3. https://www.worldbank.org/en/news/press-release/2020/06/08/covid-19-to-plunge-global-economy-into-worst-recession-since-world-war-ii
4. https://www.frontiersin.org/articles/10.3389/fphy.2020.00127/full
5. https://www.worldometers.info/coronavirus/
6. https://www.who.int/health-topics/coronavirus#tab=tab_1
7. https://www.cdc.gov/coronavirus/2019-ncov/cases-updates/forecasting.html#~:text=Why%20COVID%2D19%20Forecasting%20Is%20Important&text=Forecasts%20of%20deaths%2C%20hospitalizations%2C%20and,19%20pandemic%20in%20coming%20weeks
8. https://www.aha.org/system/files/media/file/2020/04/COVID-19-Modeling-Forecasting-the-Pandemics-Spread.pdf
9. Forecasting Models for Coronavirus Disease (COVID-19): A Survey of the State-of-the-Art
10. https://towardsdatascience.com/deep-learning-for-time-series-and-why-deep-learning-a6120b
11. MDPI: The application of Stock Index Price Prediction with Neural Networks.
12. https://arxiv.org/ftp/nlin/papers/0607/0607058.pdf
13. https://www.bankofcanada.ca/wp-content/uploads/2010/05/wp99-3.pdf
14. Time series forecasting of Covid-19 using deep learning models: India-USA comparative case study
15. How well can we forecast the COVID-19 pandemic with curve fitting and recurrent neural networks?
16. https://d2l.ai/chapter_recurrent-modern/gru.html
17. https://d2l.ai/chapter_recurrent-modern/lstm.html
18. https://github.com/illidanlab/T-LSTM
19. https://towardsdatascience.com/encoder-decoder-model-for-multistep-time-series-forecastin
20. https://github.com/edebrouwer/gru_ode_bayes/blob/master/gru_ode_bayes/models.py
21. https://github.com/nytimes/covid-19-data
22. https://covid.cdc.gov/covid-data-tracker/#cases_casesper100klast7days
23. https://www.ecdc.europa.eu/en