Auto-regressive Multi-variable Auto-encoder

Emerson V. Oliveira
Grad. Prog. in Elec. and Comp. Eng.
Fed. Univ. of Rio Grande do Norte
Natal, Brazil
evilaroliveira@gmail.com

Davi H. do Santos
Grad. Prog. in Elec. and Comp. Eng.
Fed. Univ. of Rio Grande do Norte
Natal, Brazil
davihenriqueds@gmail.com

Luiz M. G. Goncalves
Grad. Prog. in Elec. and Comp. Eng.
Fed. Univ. of Rio Grande do Norte
Natal, Brazil
lmarcos@dca.ufrn.br

***Abstract*—Due to the global pandemic disclaimer caused by the SARS-COV-2 virus propagation, also called COVID-19, governments, institutions, and researchers have mobilized intending to try to mitigate the effects caused by the virus on society. Some approaches were proposed and applied to try to make predictions of the behavior of possible pandemics indicators. Among those methodologies, some models are data orientated, also known as data-driven, which had considerable prominence over the others. Artificial Neural Networks are a widely used model among data-driven models. In this work, we propose a novel Auto-Encoder RNA architecture. This architecture aims to forecast time series related to the COVID-19 pandemic, particularly the number of deaths. The model uses as inputs possible associated time series with the desired forecasting. In the experiments, we used the representation in time series from the number of COVID-19 cases, deaths, temperature, humidity, and the Air Quality Index (AQI) of São Paulo city in Brazil. The results show that the model has a prominent forecasting accuracy for the COVID-19 deaths time series.**

***Index Terms*—machine learning, artificial neural network, auto-encoder, pandemic, COVID-19.**

I. Introduction

In early December 2019, an outbreak of unknown pneumonia turned into an epidemic that caused considerable damage, threatening to reach pandemic status [1], [2]. The agent responsible for this outbreak is a novel beta-coronavirus related to Middle East Respiratory Syndrome and Severe Acute Respiratory Syndrome virus (MERS-CoV and SARS-CoV). This new disease, the SARS-CoV-2 coronavirus, was named "COVID-19" by the World Health Organization (WHO). On January 30, 2020, the COVID-19 outbreak was declared a Public Health Emergency of International Concern by the Director-General of the WHO. Some places have implemented lockdown and suspension of all public transport, flights, and trains after January 23, 2020. Despite this, the authorities recorded approximately 40,000 confirmed cases at that time. Therefore, the National Health Commission reported more than 6 thousand cases of the disease with severe symptoms and about 900 deaths until February 10, 2020. After a short time, more the 20 different countries reported another 319 cases and one death [1], [3].

Nowadays, the data-sets recorded more than 570 million confirmed cases worldwide and more than 6.3 million deaths, showing a mortality rate of more than 1%. Attempting to predict these values or even to model how the spread of viruses can occur, researchers used several modeling techniques. Some of those techniques are the models SIR, SEIR, and SEIRS, among others [4]–[6]. These models are known as traditional or epidemiological models. In general, they use population information divided into subgroups, estimating the speed and probability of individuals in a population moving from one subgroup to another. These subgroups vary from model to model, generally observing the number of infected and recovered, among other indicators. At first, these models are valuable and are applied directly, in addition to being relatively easy to understand in theory. However, the global pandemic problem may have several aggravating factors not recognized intuitively, such as climatic conditions and social and political factors. Therefore, these methodologies may not be enough to understand the COVID-19 pandemic through and through.

As a real-world problem, several factors can affect the behavior of the pandemic, directly and indirectly. There are current studies that try to find elements that can influence the spread of the virus. Aspects such as air quality, social distancing, and use of specific equipment can influence. It is not yet sure which of these factors impact to a greater degree, despite there is already good guidance on which ones can stimulate the spread of the virus [7]–[10].

To understand the consequences and behavior of the COVID-19 pandemic, some studies used forecast methods known as data-driven. These methods try to predict the number of infected cases, deaths, and also the virus spread rate. Data-driven approaches can use artificial intelligence techniques to operate on data and better understand it. These techniques can provide possible to generate functional tools for a given action. It is also the goal of these methodologies to make future estimates from data from a time series. Finally, data-driven models are capable of assimilating complex information, which can aid in possible human decision-making. Among the most common techniques used in data-driven models are artificial neural networks. These networks prove to be strong allies in the use of data as sources of information. Due to the ability of these networks to approximate complex functions, which are model problems considered to be real-world [11].

In this context, this work proposes a new data-driven model to promote time series predictions related to the COVID-19 pandemic. More specifically, the proposed model is an auto-regressive artificial neural network of the auto-encoder type,

This work is sponsored by CAPES under grant 88887.506889/2020-00.

with punctual modifications in its functioning. The model presented here must be able to predict the number of deaths for a given data-set. As a case study, the model test receives data with values from the state of São Paulo - Brazil.

II. Theory

This section will address the theoretical knowledge necessary to understand our work. We start with a short definition of time series, the most common mathematical methods used in the endemics forecast, and end with the mathematical concepts about auto-encoders, used in this work for time series prediction.

A. Time-series

We can describe time series as a sequence of values ordered chronologically and observed over time. Although time is a continuous (or analog) measurement, a time series usually have it is values sampled at constant intervals of time (sampling frequency) [12]. The authors usually divide a time series into three components. The trend, the seasonality, and residuals (or noise) [12], [13]. The trend is the general direction or movement of the signal (series) along the observed time or long-term variation. It does not take into account seasonality or residual components. Some types of orientation are more common in time series as linear, exponential, and parabolic. Seasonality is the component that represents variations occurring in a given regular time interval. In real signs, this effect is present in situations such as weather changes, economic cycles, seasons, or even festivities. Residuals are the leftover signals with the removal of trend and seasonality components. These values can turn out large enough to mask the real orientation and seasonality of the series. There are several forms of origins for this type of behavior, which can make it difficult to predict the desired signal.

B. Auto-encoders

An auto-encoder is an artificial neural network trained to be able to copy its input into its output [14], [15]. The auto-encoder structure is divided into two parts, the encoder and the decoder. These, in turn, can be seen as two functions $Z = h(X)$ and $\hat{X} = g(Z)$. The first one is responsible for mapping the input data x to the latent space (feature/latent space). While the second produces the data reconstruction, mapping Z from the latent space back to the input data space x [15], [16]. Figure 1 shows a high-level representation of an auto-encoder.

Auto-encoders have generalized the idea of a encoder and a decoder in addition to the deterministic functions shown above, for mapping into stochastic functions $p_{encoder}(Z \mid X)$ and $p_{decoder}(\hat{X} \mid Z)$. Where $\hat{X}$ is the reconstruction of the input signal X. Given real applications, it is not entirely of interest that the auto-encoder only learns how to copy the input X. Then restrictions are made so that auto-encoders learn to copy the inputs roughly [16]. Furthermore, researchers widely used these structures for input dimensionality reduction and extraction/learning of features [15].

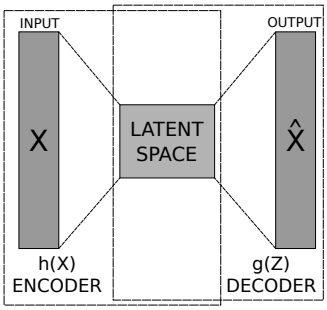

Fig. 1. High-level representation of an auto-encoder.

In the application of a *auto-encoder* on a given training set $S = \{x_i | x_i \in R^d\}$, where $1 < i < n$, this can be modeled as:

$$EA = \begin{cases} Z = h(w_e, b_e; X) \\ \hat{X} = g(w_d, b_d; Z) \end{cases} \tag{1}$$

Having the equation (1), $h(\cdot)$ is the encoder and $g(\cdot)$ is the decoder, which are usually artificial neural networks. w_e and b_e are encoder parameters and w_d and b_d are decoder parameters. In the case of neural networks, these parameters are the sets of weights and biases of the networks encoder and decoder, respectively.

Training an auto-encoder is optimizing, in this case minimizing, the error of the following loss function (loss function). So:

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{n} \|X_i - \hat{X}_i\|_2^2 \tag{2}$$

Where $\theta = (w_e, b_e; w_d, b_d)$. For the optimization solution present in the equation (2), the Gradient Descending or Stochastic Descending algorithm is normally used.

Other examples of auto-encoders types found in the literature are SparseAE, DenoisingAE, ConvolutionalAE, ContractiveAE, and VariationalAE, among others.

III. Related Works

In the literature, it is possible to find a range of auto-encoders used in different areas and study cases. For time series prediction, we found a deep learning framework that used Wavelet Transforms (WT) combined with Stacked auto-encoders (SAE) and LSTM layers [17]. The framework is applied to predict the stock price. Initially, the methodology uses WT to decompose the time series and eliminate noise, followed by the SAE layers application to generate high-level features. Finally, the framework presents the auto-encoders feature extraction from the LSTM layers for prediction. The authors consider the auto-encoders the central part of the study, as it is at this stage that the financial characteristics are acquired. They evaluated the methodology based on the quality of prediction and profitability [17].

We also found a proposed Deep LSTM-based Stacked Auto-encoder (LSTM-SAE) for multi-variable time series prediction

[18]. According to the authors, the difference in this proposal is how they initialize the network weights. They use a methodology that divides the training into two phases. The first is an unsupervised pre-training, called Greedy Layer-wise Pre-training Phase, for acquiring a set of initial weights for the next phase. The second phase is called the Fine-tuning Phase. The pre-training phase is composed of three LSTM-AE blocks. So, the framework trains the first network block to reconstruct the original signal. Then, the central layer of the first block receives the inputs and returns the latent space of signal features. Thus, having the first block encoded values, the next block of the LSTM-AE is trained to get close to the original input. The last step is repeated for the third block or the remaining blocks that may exist. When the pre-training phase ends, the second phase begins, where the authors applied a DLSTM [19] previously proposed by themselves. They train the network to predict a single sample of the desired time series [18].

In direct application to COVID-19, a proposed mixing of Variational Auto-Encoder (VAE) with LSTM networks try to solve the COVID-19 spread prediction problem [20]. The authors divided the proposed model into two branches. The first is a self-attention auto-encoder LSTM, fed by virus propagation data by day and country. Along with that, government policies by day and by each country, and urban characteristics by day and country. They split this first branch into self-attention LSTM sequence encoder and LSTM sequence decoder. According to the authors' references, the self-attention mechanism makes the LSTM capable of understanding the representation of its inputs relating to the positioning of each sequence. The VAE is the second branch of the model and works in parallel with the self-attention LSTM mechanism. The VAE is fed by a spatial matrix of dimensions and repeated throughout the training duration, with timestamps referring to the date and time. The two model outputs branches are concatenated in the feature dimension and sent to the LSTM sequence decoder, which returns the prediction values. This study attempts to predict the spread (accumulated number of cases) of COVID-19 worldwide and in each country separately [20].

As more close work, our study group proposed a Long-short Term Memory model to forecast the number of COVID-19 deaths in the São Paulo city in brazil [21]. The authors used a Montecarlo test, training 50 trails from their model and acquiring a statistic distribution concerning the model Root Mean Square Error (RMSE) from the entire and test data-set prediction. In the study, the authors used five input features. They used the number of cases, the number of deaths, the temperature, humidity, and air quality index AQI from March 27, 2020, to June 03, 2021.

The model presented has the differential of predicting directly the time series related to the number of deaths of COVID-19, as in the last cited work, and is also integrated into a simple architecture of multilayer-perceptrons. This simplicity can reduce the computational cost in training concerning complex ANN as LSTM or Convolutional. In addition, we made specific modifications to the model architecture that can be changed and later improved, adding flexibility to the model for receiving new features.

IV. Multi-variable Auto-Encoder

To solve the problem of time series predictions related to the COVID-19 pandemic, we propose a multivariate input data-driven model capable of predicting an amount h of samples for COVID-19 deaths. The model is an expansion of an early auto-encoder model proposed [22], where we present a multivariate version of the original model.

The proposed model is a conventional auto-encoder, but with some modifications in 1) how the model observes the inputs, 2) how the model work with the latent space, and 3) how the train steps update the network weights. The first proposed change is to add a new auto-encoder for each input feature. Figure 2 shows the first proposed modification, which takes place in how the model behaves concerning a series of inputs.

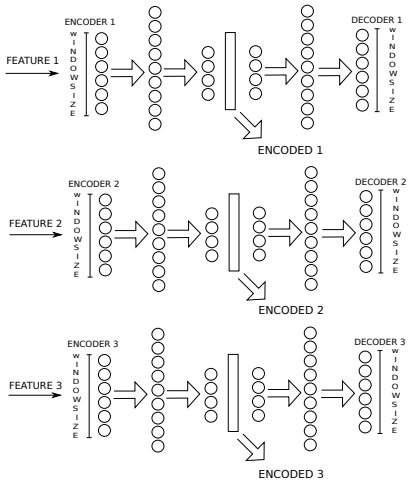

Fig. 2. Representation of how the proposed auto-encoder creates a new auto-encoder network for each input series.

From Figure 2, it is possible to observe how the model deals with three series as inputs. A new auto-encoder network is created for each series (or feature), thus having the same number of encoded series. The architecture of each of these auto-encoders can vary. Although, it must always be a layer of greater or equal size between the input and the last layer of the encoder. An attempt to increase the field of possible nuances from the inputs without changing the input layer length, which can contribute to better feature extraction from the encoder. This layer has the larger number of neurons in Figure 2. The symbol $\Rightarrow$ represents a linear transformation occurring in the transition between layers. From this point, it is possible to understand the second proposed modification. All these encoded values are unified (concatenated) into a single 1-D series, thus forming a single latent space. Figure 3 shows the second proposed modification.

Figure 3 shows that the latent space has a reduced representation of each input series. This final latent space undergoes

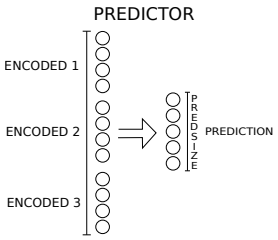

Fig. 3. Representation of how the proposed model reorganizes the latent space and makes the prediction.

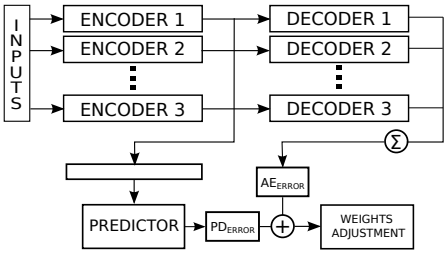

Fig. 4. High-level flowchart of the training step for the proposed model.

one more linear transformation to return the predicted series. The model tries to minimize the error between the predicted and expected value. For the proposed model, this occurs in two situations. The first is when each auto-encoder tries to recreate the input of its respective series. The errors of all auto-encoders are summed, resulting in a single error called ae_{error}. $e_1, e_2, \ldots, e_n$ being the reconstruction errors of each auto-encoder in a model. The second is the prediction error, which we called pd_{error}. The set of equations (3) shows how ae_{error} and pd_{error} are calculated.

$$\begin{aligned} ae_{error} &= \sum_{i=1}^{n} e_i \\ pd_{error} &= Loss(expected \mid predicted) \end{aligned} \tag{3}$$

We need a loss function to calculate the errors. In this case, we named it $Loss$, so we have the prediction pd_{error} and the individual auto-encoder errors as the output of this function. With the results obtained in the equations (3), we have the total model error.

$$t_{error} = ae_{error} + pd_{error} \tag{4}$$

Finally, the equation (4) shows the accumulated total error obtained on the model.

Figures 2 and 3 shows the addition of "window size" and "pred size" items. They represent the window size for inputs and the number of samples for the model prediction. Once the inputs are in the expected format, it is enough to define the network sizes, the loss function, and a weight calculation optimization algorithm. With all this defined, it is possible to train the model. Figure 4 shows a flowchart that also exemplifies the training step.

Figure 4 shows a representative flow of information followed in the training. The networks receive the inputs, where the encoder creates the vector with the encoded series. Then, those series flow through two different paths. One goes to the decoder, and another is to be concatenated and passed through the predictor. Then, the loss function calculates the errors, sums them, and fits the weights.

To implement this model and carry out the experiments, we use the Pytorch library [23] in the *python* programming language.

V. Experiments

To evaluate the ability of the proposed model to assimilate a multivariate data-set, we train eight models with different combinations of input series. Thus, it is possible to assess in isolation whether or not a given series has a positive impact on the final result of the prediction.

A. Data-set

For the experiments, we used data from São Paulo - Brazil. The COVID-19-related series was the number of cases and deaths from the *World Health Organization* [24]. We also used temperature, humidity, and the AQI [25] from the locale as possible side-related information concerning COVID-19 spread. We normalized the data with the "*MinMax Scale*" normalization, where for a data-set $X = \{X_1, X_2, \ldots, X_i\}$ follows the following formulas.

$$\begin{aligned} X_{std} &= \frac{X - X_{min}}{X_{max} - X_{min}} \\ X_i &= X_{std} * (X_{i-max} - X_{i-min}) + X_{i-min} \end{aligned} \tag{5}$$

In the equation (5) we have X_{std} as the standard deviation of the total data-set. Then, normalization is applied to each series i contained in the data-set X. This operation results in a new normalized data-set, where the values range from 0 to the maximum of 1.

The data-set used in the experiments corresponds to all five input series starting on April 1, 2020 and ending on September 1, 2021, totaling 522 days (samples). Figure 5 shows the data collected and used in the tests. It is important to note that the data used in the tests follow the seven-day moving average format.

B. Models and Training Methodology

Initially, the normalization is applied, and then the models are trained. We trained a new model for each feature combination, in which the randomly generated initial weights are the same for all eight models. This approach reduces the variance that the initialization of the weights can cause in the model convergence. This seed weights fix makes the comparison fairer and more reliable since all models start with the same weights. Thus, the only thing that differentiates them is the series as input. Table I shows the series combinations used in the tests.

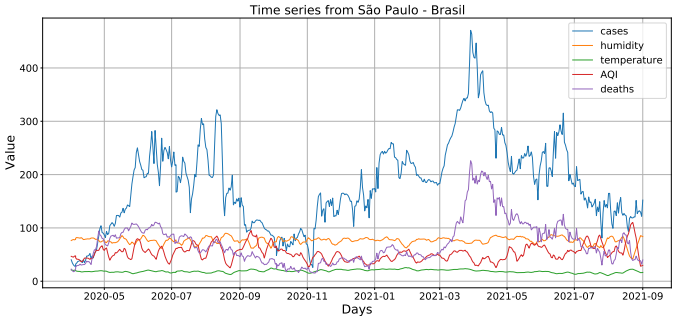

Fig. 5. Plot of a seven-day moving average for all variables used in the experiments. For better data visualization of the other series, we divided by 10 the values of the "Case" time-series.

TABLE I
SERIES COMBINATIONS

Combination	Series (*features*)	Prediction
1	deaths	deaths
2	deaths, cases	deaths
3	deaths, cases, humidity	deaths
4	deaths, cases, temperature	deaths
5	deaths, cases, AQI	deaths
6	deaths, cases, humidity, temperature	deaths
7	deaths, cases, humidity, AQI	deaths
8	deaths, cases, humidity, temperature , AQI	deaths

Regarding the values used in the auto-encoders architecture, we used 7 samples for the input window size and to amount of desired predicted values. As shown, there is an increase in the number of neurons in the layer following the input layer. We assigned 35 neurons to this layer. To the latent space layer, we delegated 7 neurons. These three layers are the encoder from a single input series. Thus, the decoder of each of the networks has the same encoder architecture but in reverse. Although, in this case, they become identical in the number of neurons $[7, 35, 7]$.

Another important hyperparameter used for training the model was 32 for the batch size of network parameters adjustment. Besides that, all activation functions used were the Rectifier Linear Unit (ReLU), and the learning rate was 1×10^{-4}. Finally, each model in the combinations presented in Table I was trained for 250 epochs. Therefore, all training data pass through to the network for each epoch.

We formulated the data-set for each input window have an overlap of 6 samples. So, only the sample corresponding to the current time instant is distinct from the past input. We used 85% inputs for training and 15% to model validation during training. The evaluation metric used to quantify the models during the train was the Root Mean Squared Error (RMSE). It is important to remark that the model never had an adjustment in its parameters over the inputs of the validation set. These validation inputs can always be considered new to the model. For the final evaluation of each combination, we formulate a new data-set with no overlap in its entries. This time, the entire data-set passes through the network. Although, the network never adjusted its parameters for the samples referring to the validation set. Thus, the model evaluation happens in two moments, for inputs samples we used for training and those in the validation set (never adjusted weights for those). In this final evaluation, in addition to the RMSE, we also measured the Mean Absolute Error (MAE) and the Median Absolute Error (MDAE).

VI. RESULTS

The experiments described in the section V aim to test the proposed architecture and evaluate in a simple way which series may influence model response. To quantify each combination, Table II presents the evaluation metrics collected only for the test (or validation) data. As a remark, the model never adjusted its parameters for those validation samples.

TABLE II
EVALUATION METRICS OBTAINED FOR NORMALIZATION "*MinMax Scale*" IN TEST SAMPLES.

Combination	RMSE	MAE	MDAE	(Mean)
1	**0.040**	**0.027**	**0.019**	*0.029*
2	0.042	0.030	0.023	0.032
3	0.052	0.038	0.028	0.039
4	0.049	0.036	0.026	0.037
5	0.052	0.038	0.030	0.040
6	0.048	0.035	0.027	0.037
7	0.046	0.034	0.026	0.035
8	0.044	0.032	0.024	0.033

From Table II, it is possible to observe the highlighted the lowest values of each metric.

Figures 6 and 7 visually show the model prediction for combination 1 with the lowest metrics values and to the combination 8 with the higher number of input features.

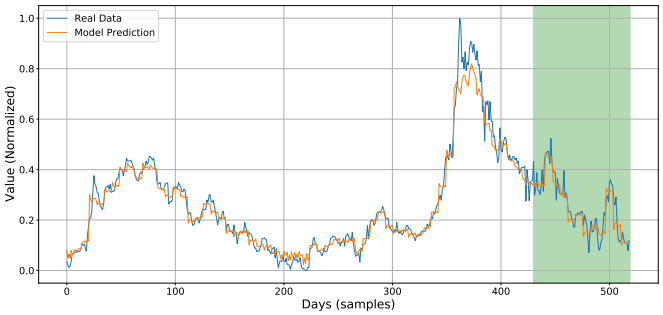

Fig. 6. Plot comparing the real data and the data predicted by the model with only the time series referring to the deaths as input. Highlighted (green) is the validation region, where the model never adjusted the weights for these samples.

Given the Figures 6 and 7, it is possible to observe the produced model approximation that the model is close to the desired data values. This approximation becomes fairer at validation, highlighted in (green) the figures, in which such metric values we expressed in II. It is also possible to notice that for samples referring to the portion that participated in the model train, the model does not perform so closely to the death values compared to the validation. This behavior may occur due to a smaller number of validation samples and the high

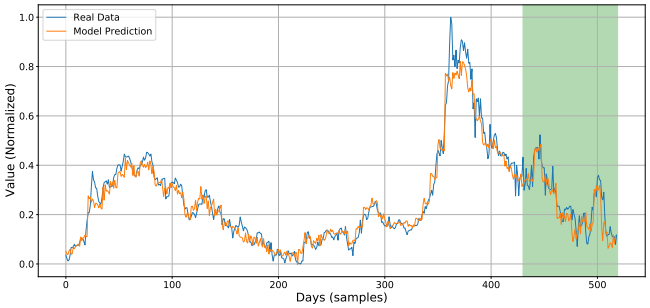

Fig. 7. *Plot* comparison of the real data and the data predicted by the model with the time series referring to the deaths, cases, humidity, temperature, and AQI as input. Highlighted (green) is the validation region, where the model never adjusted the weights for these samples.

variation in the series values to predict in the training samples. The combination of 1 outperforms the others, with the lowest value of all metrics and metrics mean. That is the simplest regression case, so the training algorithm and optimization function tend to adjust the weights more influenced by the values they want to approximate. In other words, the algorithm can reach the output easily, using just one of the inputs, because this input takes the information the model desire. The other input series can add information and help to generalize, but they add noise to the output.

VII. Conclusion

In this work, we proposed a novel data-driven model for predicting time series related to the COVID-19 pandemic. From the experiment performed, we observed that the model showed optimistic proficiency in predicting a single time series linked to COVID-19 from one, two, three, four, or five series as inputs. In addition, it was also possible to observe an increase in the evaluation metrics values and noise as the number of features increased. This behavior can be related to the inherent neural network feature of adjusting the weights to find a small way to convergence. In future works, we will try to change the train loop and split the model weights adjustment into two phases, where the auto-encoders and the predictors are adjusted apart.

References

[1] C. Anastassopoulou, L. Russo, A. Tsakris, and C. Siettos, "Data-based analysis, modelling and forecasting of the covid-19 outbreak," *PloS one*, vol. 15, no. 3, p. e0230405, 2020.

[2] Q. Li, X. Guan, P. Wu, X. Wang, L. Zhou, Y. Tong, R. Ren, K. S. Leung, E. H. Lau, J. Y. Wong *et al.*, "Early transmission dynamics in wuhan, china, of novel coronavirus–infected pneumonia," *New England journal of medicine*, 2020.

[3] W. H. Organization *et al.*, "Novel coronavirus (2019-ncov): situation report, 11," 2020.

[4] I. Cooper, A. Mondal, and C. G. Antonopoulos, "A sir model assumption for the spread of covid-19 in different communities," *Chaos, Solitons & Fractals*, vol. 139, p. 110057, 2020.

[5] Z. Yang, Z. Zeng, K. Wang, S.-S. Wong, W. Liang, M. Zanin, P. Liu, X. Cao, Z. Gao, Z. Mai *et al.*, "Modified seir and ai prediction of the epidemics trend of covid-19 in china under public health interventions," *Journal of thoracic disease*, vol. 12, no. 3, p. 165, 2020.

[6] L. Djaparidze and F. A. Lois, "Sars-cov-2 waves in europe: A 2-stratum seirs model solution," *medRxiv*, 2020.

[7] A. M. Elsaid and M. S. Ahmed, "Indoor air quality strategies for air-conditioning and ventilation systems with the spread of the global coronavirus (covid-19) epidemic: Improvements and recommendations," *Environmental Research*, p. 111314, 2021.

[8] H. Xu, C. Yan, Q. Fu, K. Xiao, Y. Yu, D. Han, W. Wang, and J. Cheng, "Possible environmental effects on the spread of covid-19 in china," *Science of the Total Environment*, vol. 731, p. 139211, 2020.

[9] I. M. Ismail, M. I. Rashid, N. Ali, B. A. S. Altaf, and M. Munir, "Temperature, humidity and outdoor air quality indicators influence covid-19 spread rate and mortality in major cities of saudi arabia," *Environmental Research*, p. 112071, 2021.

[10] E. D. Freitas, S. A. Ibarra-Espinosa, M. E. Gavidia-Calderón, A. Rehbein, S. A. Abou Rafee, J. A. Martins, L. D. Martins, U. P. Santos, M. F. Ning, M. F. Andrade *et al.*, "Mobility restrictions and air quality under covid-19 pandemic in são paulo, brazil," 2020.

[11] B. Coppin, *Artificial Intelligence Illuminated*, ser. Jones and Bartlett illuminated series. Jones and Bartlett Publishers, 2004. [Online]. Available: https://books.google.com.br/books?id=LcOLqodW28EC

[12] J. F. Torres, D. Hadjout, A. Sebaa, F. Martínez-Álvarez, and A. Troncoso, "Deep learning for time series forecasting: A survey," *Big Data*, vol. 9, no. 1, pp. 3–21, 2021.

[13] R. H. Shumway, D. S. Stoffer, and D. S. Stoffer, *Time series analysis and its applications*. Springer, 2000, vol. 3.

[14] Y. Lecun, "Phd thesis: Modeles connexionnistes de l'apprentissage (connectionist learning models)," 1987.

[15] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[16] J. Zhai, S. Zhang, J. Chen, and Q. He, "Autoencoder and its various variants," in *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2018, pp. 415–419.

[17] W. Bao, J. Yue, and Y. Rao, "A deep learning framework for financial time series using stacked autoencoders and long-short term memory," *PloS one*, vol. 12, no. 7, p. e0180944, 2017.

[18] A. Sagheer and M. Kotb, "Unsupervised pre-training of a deep lstm-based stacked autoencoder for multivariate time series forecasting problems," *Scientific reports*, vol. 9, no. 1, pp. 1–16, 2019.

[19] ——, "Time series forecasting of petroleum production using deep lstm recurrent networks," *Neurocomputing*, vol. 323, pp. 203–213, 2019.

[20] M. R. Ibrahim, J. Haworth, A. Lipani, N. Aslam, T. Cheng, and N. Christie, "Variational-lstm autoencoder to forecast the spread of coronavirus across the globe," *PloS one*, vol. 16, no. 1, p. e0246120, 2021.

[21] D. P. Aragão, E. V. Oliveira, A. A. Bezerra, D. H. dos Santos, A. G. da Silva Junior, I. G. Pereira, P. Piscitelli, A. Miani, C. Distante, J. S. Cuno, A. Conci, and L. M. Gonçalves, "Multivariate data driven prediction of covid-19 dynamics: Towards new results with temperature, humidity and air quality data," *Environmental Research*, vol. 204, p. 112348, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0013935121016492

[22] I. G. Pereira, J. M. Guerin, A. G. Silva Júnior, G. S. Garcia, P. Piscitelli, A. Miani, C. Distante, and L. M. G. Gonçalves, "Forecasting covid-19 dynamics in brazil: a data driven approach," *International Journal of Environmental Research and Public Health*, vol. 17, no. 14, p. 5115, 2020.

[23] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[24] World health organization - coronavirus disease (covid-19) pandemic. [Online]. Available: https://www.who.int/emergencies/diseases/novel-coronavirus-2019

[25] World air quality index project. [Online]. Available: https://aqicn.org/data-platform/covid19/