LSTM perfomance analysis for predictive models based on Covid-19 dataset

Isac Cruz-Mendoza[1], Jonathan Quevedo-Pulido[1], Luz Adanaque-Infante[1,2]
[1] Universidad Nacional Mayor de San Marcos UNMSM, Lima, Perú
[2] INICTEL-UNI, Universidad Nacional de Ingeniería UNI, Lima Perú
isac.cruz@unmsm.edu.pe,11190108@unmsm.edu.pe,ladanaque@inictel-uni.edu.pe

***Abstract*—Within the large amount of data that can be processed with Neural Networks (NN), COVID-19 is leaving us a lot of information that is susceptible to be treated and set trends regarding the development of the disease in the country. The present work shows the implementation and the optimization of a Long Short-Term Memory (LSTM) Neural Network in two different simulation environments, with a dataset related to the number of infected people by COVID-19 in Peru, in order to optimize the prediction level on the number of infected people on following days.**

***Index Terms*—Optimization, Neural Networks, LSTM, Performance, MATLAB, Colab.**

I. INTRODUCTION

COVID-19 is produced by a newly identified s train of coronavirus SARS-CoV2, which belongs to the family coronaviridae, with RNA genome. Due to the imminent advance of the disease, the governments of many countries of the world have adopted policies of social isolation and massive diagnostic tests. Peruvian government implemented a platform that allows extraction of raw data which can be used on prediction models [1], in specific w ith n eural n etworks. Use of neural networks as learning tools through the computer is well known when a large amount of data is involved. Neural networks workflow s tarts w ith t raining p hase, which can be carried out in different environments such as MATLAB or Colab, and continues with validation and testing. These two last parts depend on both,the data and the aim of the network. Different types of systems can be validated through the training of a neural network, to offer solutions in terms on prediction models in a short time. Several authors works on the implementation of different approaches in training and validation phases, which works with recurrent networks for time series [2]

Regarding diseases prediction, several works were found on literature, using a Recurrent Neural Network (RNN) and LSTM for dysplidemia prediction [3]. Some works realized a complete study for a model prediction based on a Artificial Neural Network (ANN), others shows a LSTM network analysis, both of them for cardiovascular diseases prediction models [4], [5]. For Parkinson disease, a perceptron based model has been used [6], and for glucose level on blood prediction, different types of Neural Networks was implemented and compared [7], [8]. Dengue is a disease that spreads uncontrollably in Amazonian regions, some studies included a RNN analysis to implement predict models [9], also for influenza, with a multi channel neural network [10]. The main part of this work is the training and validation of a LSTM on MATLAB and Colab in order to obtain a prediction model with acceptable levels of efficiency and error. The paper is organized as follows, in the next section, training methodology of the LSTM is presented. In section III, we present LSTM test analysis. Performance comparisons are given in section IV. Finally, conclusions are shown in section V.

II. TRAINING METHODOLOGY

A. Dataset of Covid-19 positive cases

Statistical data related to the SARS - CoV2 pandemic is released daily. In this research, people infected dataset was taken of ”the Republic Data”, obtaining official data of infections from day one. The selected dataset is from 06/03/2020 until 27/05/2020, give us an average of 83 days. Figure 1 shows the workflow followed.

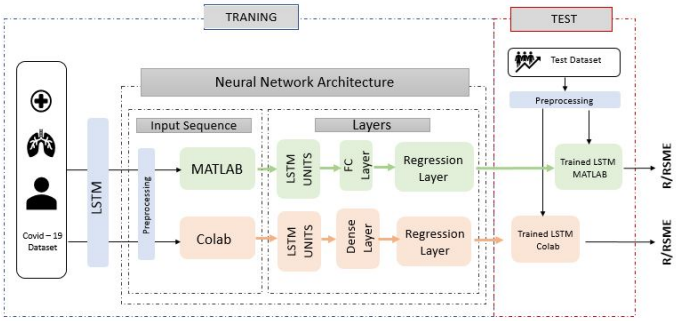

Fig. 1. LSTM for Covid-19 disease prediction models

B. LSTM Neural Network

Recurrent neural networks (RNN) are very useful for processing data sequences because of their loop structure, which works as a memory. Also, RNN are known as dynamic networks, in which the output depends on the current input, the hidden states of the network and the previous inputs and outputs. LSTM (Long Short-Term Memory) network belongs to the category of Gated RNNs, which obtains accurate models for time series. These networks implement a repetitive modules chain and four layers that interact with each other. LSTM network add or removes information on the state of the cell,

because of switching gates built with sigmoid function and a point multiplier, for this reason, LSTMs can remember or forget values. Each LSTM module is made up of four gates, an input gate, external-input gate, forget gate and output gate. These gates executes internal operations within each module, with an internal state which remembers the values linked to cells in a hidden state, which represents the output of the module [7].

C. Implementation of a LSTM on Colab

Google colaboratory or Colab, is a Phyton virtual tool which designates a processor, such as GPU or TPU, in order to execute functions created by the programmer. Machine learning applications can be run on this platform, like Neural Networks. Colab has different types of libraries, offering a dynamic approach, in this application, we use Keras library and three steps, preprocessing of input data, network structure and training phase.

1) Data Pre-Processing: It is necessary to separate data for training and test phases of the LSTM, in this work, two first months (march and april) data was used for training, and last month data (may) was used for testing phase, as shown in 2, Due to the size of the dataset, we implemented a normalization process, then the data with the highest value does not influence the decision making of the network. Compare to the MATLAB approach, where the data enters in a block, in Colab the scaled data enters on the LSTM one by one.

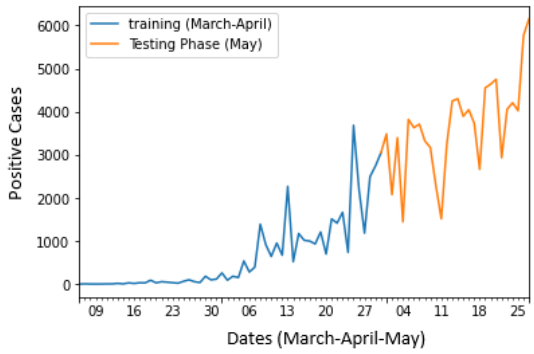

Fig. 2. Training and Testing Phase on Colab

2) Network Description: For the LSTM implementation on Colab we used a sequential approach, with one neuron at the input, 30 neurons in the hidden layer and a dense layer at the output of the network, this is a layer of artificial neurons (NN) highly connected. The LSTM implementation includes an NN with a linear activation function, $F(X) = X$ which only reflects the output value.

3) Training Phase: This phase was implemented with 150 epochs, with RMSprop optimizer and a default learning rate of 0.001. The loss has been parameterized with the MSE (mean squared error) function, so that the loss in each iteration reaches the minimum value. The last epoch (250) has a loss of 0.0288 of loss, this result is reflected in figure 3, we observe that from the 80th epoch the errors do not vary in great magnitude. After the 80th epoch, it can be noticed an overfitting, therefore, performing the training with 80 epochs is enough to achieve acceptable results in terms of predicted values.

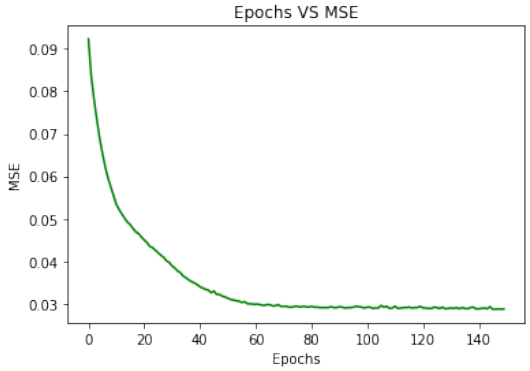

Fig. 3. Training Error(Epoch=250)

D. LSTM Implementation on MATLAB

MATLAB is a mathematical package created in the 70s by Clever Moler, over the years it has been expanding its functions through toolbox related to data processing, machine learning and neural networks. We used command-line functions provided by MATLAB for the development of this work. The methodology in this approach work as follows: Preprocessing of the input data, network implementation and training.

1) Pre-processing of input data: The data obtained from the Covid 19 per day positive case dataset is in a linear format and has 83 data in total. Due to the size of the dataset, data blocks were formed. The equation 1 shows a sequence of 11 numbers data that was ordered in a block of 7 numbers data forming the equation 2. The last column was taken as the expected value for the network and the rest as input data. This procedure was applied to the 83 data in order to each entry to provide more information on training phase.

$$T = [1; 5; 2; 2; 2; 5; 5; 16; 5; 28; 15] \tag{1}$$

$$data = \begin{bmatrix} 1 & 5 & 2 & 2 & 2 & 5 & 5 \\ 5 & 2 & 2 & 2 & 5 & 5 & 16 \\ 2 & 2 & 2 & 5 & 5 & 16 & 5 \\ 2 & 2 & 5 & 5 & 16 & 5 & 28 \\ 2 & 5 & 5 & 16 & 5 & 28 & 15 \end{bmatrix} \tag{2}$$

After the formation of data blocks, a standardization was performed by subtracting the average and dividing each data by the standard deviation, in order to achieve better training and avoid divergence. Finally, 80 % of the data was separated for training and the other 20% was used for network testing.

2) Network description: The network architecture used in this work has four stages: an input sequence, an LSTM layer, a fully connected network, and a single output regression layer. Some tests were carried out to obtain good results regarding

the mean square error, as observed in table I. After these tests, we chose 15 neurons for the network architecture on input layer and 250 LSTM units in the LSTM layer.

TABLE I
RMSE COMPARISON

N°	Input Length	Units LSTM	RMSE	Learning Rate
1	8	250	1511.2	0.001
2	13	250	700.3885	0.001
3	13	225	718.266	0.001
4	13	275	906.1428	0.001
5	15	250	678.8516	0.001
6	15	400	1386.4	0.001
7	15	200	1144.3	0.001
8	15	200	2179.5	0.005
9	15	200	1688.2	0.0005
10	20	250	1266.2	0.001

3) Training network: Network training consists of entering the input data blocks together with the expected outputs into the network. Initially the network was trained with 250 iterations, an ADAM optimizer and a learning rate of 0.0005. However, the optimal learning rate was 0.001. Figure 4 it shows the variation of the RMSE throughout the training during the 250 iterations. These can be reduced up to 150 as observed in figure 5, after 150 iterations we can see that the error does not present significant variations. This indicates that an early stop can be implemented in the network in order to avoid a possible overfitting.

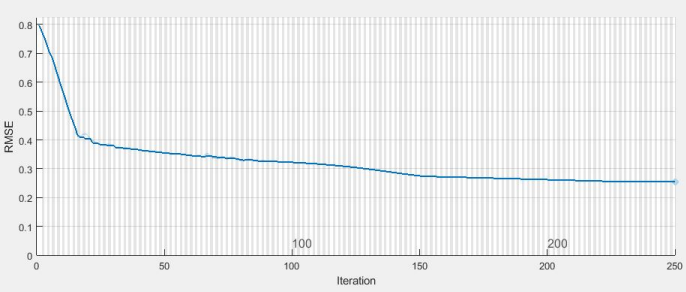

Fig. 4. Training RMSE variation

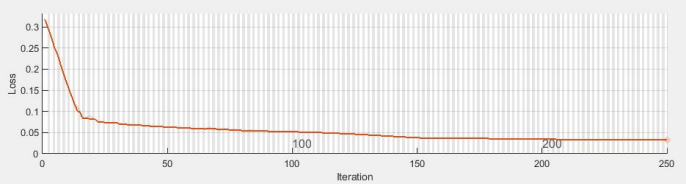

Fig. 5. Training Loss variation

III. TEST ANALYSIS ON LSTM

On the test phase, neural networks use several metrics in order to compare results between different models and to evaluate which of them has the best performance. To achieve the best performance, we used RMSE and MSE metrics, which were calculated by solving the equations 3 and 4, respectively.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_t - \hat{y}_t)^2 \tag{3}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_t - \hat{y}_t)^2} \tag{4}$$

Where n is the number of values predicted by the LSTM, y_t is the real value and $\hat{y}_t$ is the predicted value of the neural network.

A. Test on Collab

While in MATLAB the data is tested in blocks, in Colab the test is one by one, otherwise we will have a greater error and the prediction graph will not follow the trend of the real values. For testing phase of the LSTM trained network, the data from 04/30/2020 of the dataset were used and normalized between 0 and 1, to avoid errors on the network trained on section II of this work, this data was tested one by one. For that reason, prediction results were normalized, so we applied a re scaling to obtain the real values, as observed on figure 6, which shows a continued trend over time. The real values are plotted on the red line, and the prediction values are plotted on blue line.

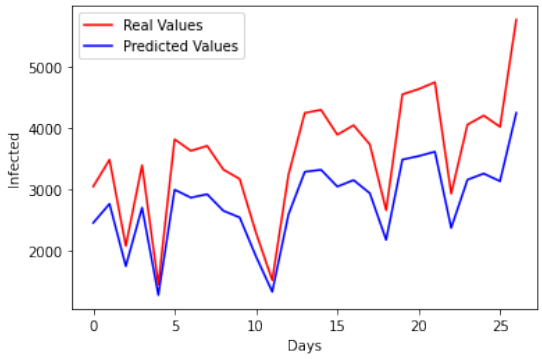

Fig. 6. Trend diagram of validation data

Figure 7 shows the comparison of the predicted data with the real data in a dispersion graph, the regression line is plotted in blue, and the dispersion points are plotted in red, it can be noticed that the trend is linear and increasing. The result of the RMSE error is 813, due to the reduced length of the dataset. Despite this, prediction level is still acceptable. The correlation index gives a result of 0.5376.

B. Test on MATLAB

To evaluate the results obtained by the network in MATLAB, test data was entered in 15-value block format. Test phase gave us results such as the mean square error and the correlation index of the data obtained from the predictions, to study the relationship between the time variable and the number of positive cases per day. For the calculation of the correlation index, the equation 5, where S_{xy} represents the co variance, and the standard deviation was represented by S_x and S_y. Solving the equation 5 we obtained 0.6282 as a result.

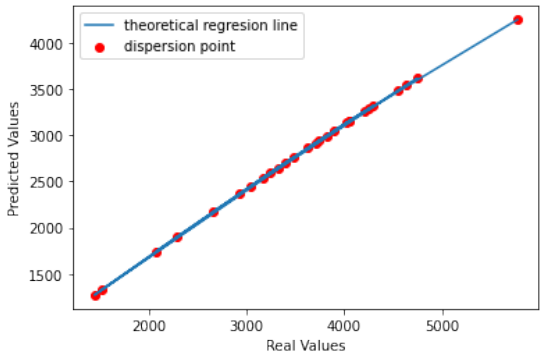

Fig. 7. Dispersion Diagram of predicted values

The positive value indicates that the relationship between the variables time and positive cases per day is direct. As for the numerical value, it represents that there is dispersion in the obtained data.

$$r = \frac{S_{xy}}{S_x S_y} \tag{5}$$

The prediction obtained is shown in the upper part of the figure 8, the actual values are shown in blue and the network prediction in red. In the lower part of the 8, the difference between the network output and the actual value is graphically observed. The value obtained from the RMSE is shown in the central part.

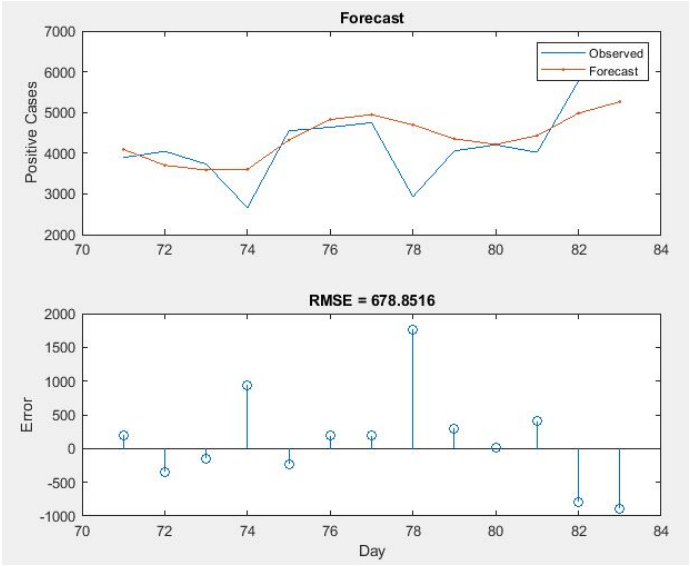

Fig. 8. Prediction with Test data

IV. PERFOMANCE COMPARISONS

Several comparable aspects were found between MATLAB and Colab, like the number of units on training phase, in MATLAB we used 250 units, meanwhile in Colab we used only 30 units, this explains the increased use of resources by MATLAB. Regarding iterations, in MATLAB more iterations were necessary for the network to reach an acceptable error, while in Colab the optimal values are reached in 80 iterations in MATLAB 150 iterations are needed to reach them. The RMSE error, described by equation 4 was considered as a metric in our work. For the analysis in MATLAB, a result of 678 was obtained, while for Colab, 813 was obtained, which means a more reliable prediction in MATLAB analysis as seen on figures 8 and 6. Regarding the correlation index, in Colab we obtained 27 prediction data, with a correlation index of 0.5376. While in MATLAB, 13 prediction data are obtained with a correlation index of 0.6282. This means that there is a lower level of dispersion of the data in MATLAB compared to Colab. As a result, Colab requires a few number of iterations but the RMSE error is higher than the RMSE error in MATLAB.

V. CONCLUSIONS

Although the dataset used for this analysis is small, results shows that the best training approach is obtained with Colab, and the best testing approach is obtained with MATLAB. This makes us think of a mixed solution in order to obtain an accurate prediction model. The inclusion of a larger dataset, re powering training stages, and implement it partially on reconfigurable platforms like FPGAs can be a continuation of this work.

REFERENCES

[1] Secretaría de Gobierno Digital Presidencia del Consejo de Ministros, "www.datosabiertos.gob.pe," 01 de Abril de 2020.

[2] Y. Gao, S. Q. Wang, J. H. Li, M. Q. Hu, H. Y. Xia, H. Hu, and L. J. Wang, "A prediction method of localizability based on deep learning," *IEEE Access*, pp. 1–1, 2020.

[3] S. Cui, C. Li, Z. Chen, J. Wang, and J. Yuan, "Research on risk prediction of dyslipidemia in steel workers based on recurrent neural network and lstm neural network," *IEEE Access*, vol. 8, pp. 34153–34161, 2020.

[4] Y. Ma and S. Wang, "The application of artificial neural network in the forecasting on incidence of a disease," in *2010 3rd International Conference on Biomedical Engineering and Informatics*, vol. 3, pp. 1269–1272, 2010.

[5] H. D. Park, Y. Han, and J. H. Choi, "Frequency-aware attention based lstm networks for cardiovascular disease," in *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 1503–1505, 2018.

[6] Z. A. Bakar, N. M. Tahir, and I. M. Yassin, "Classification of parkinson's disease based on multilayer perceptrons neural network," in *2010 6th International Colloquium on Signal Processing its Applications*, pp. 1–4, 2010.

[7] A. Aliberti, A. Bagatin, A. Acquaviva, E. Macii, and E. Patti, "Data driven patient-specialized neural networks for blood glucose prediction," in *2020 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, pp. 1–6, 2020.

[8] S. Ambekar and R. Phalnikar, "Disease risk prediction by using convolutional neural network," in *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pp. 1–5, 2018.

[9] M. Chovatiya, A. Dhameliya, J. Deokar, J. Gonsalves, and A. Mathur, "Prediction of dengue using recurrent neural network," in *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, pp. 926–929, 2019.

[10] B. Fu, Y. Yang, Y. Ma, J. Hao, S. Chen, S. Liu, T. Li, Z. Liao, and X. Zhu, "Attention-based recurrent multi-channel neural network for influenza epidemic prediction," in *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pp. 1245–1248, 2018.