International Journal of Electrical and Computer Engineering (IJECE)
Vol. 10, No. 6, December 2020, pp. 6574~6581
ISSN: 2088-8708, DOI: 10.11591/ijece.v10i6.pp6574-6581

Prediction of atmospheric pollution using neural networks model of fine particles in the town of Kennedy in Bogotá

Juan Camilo Pedraza, Oswaldo Alberto Romero, Helbert Eduardo Espitia
Facultad de Ingeniería, Universidad Distrital Francisco José de Caldas, Colombia

Article Info

Article history:

Received Feb 17, 2020
Revised Jun 1, 2020
Accepted Jun 14, 2020

Keywords:

Air monitoring
Atmospheric pollution
Neural networks
Particulate material
Smart cities

ABSTRACT

This work shows an application based on neural networks to determine the prediction of air pollution, especially particulate material of 2.5 micrometers length. This application is considered of great importance due to the impact on human health and high impact due to the agglomeration of people in cities. The implementation is performed using data captured from several devices that can be installed in specific locations for a particular geographical environment, especially in the locality of Kennedy in Bogotá. The model obtained can be used for the design of public policies that control air quality.

Corresponding Author:

Oswaldo Alberto Romero,
Facultad de Ingeniería,
Universidad Distrital Francisco José de Caldas,
Cra. 7 # 40b-53, Bogotá, Colombia,
Email: oromerov@udistrital.edu.co

1. INTRODUCTION

Air pollution is an acute threat, it is a phenomenon that has a particular impact on human health. The changes that occur in the chemical composition of the atmosphere can change the climate, produce acid rain or destroy the ozone layer, all phenomena of great global importance. The World Health Organization (WHO) considers air pollution as one of the most important global priorities [1].

The use of non-renewable resources in the production of energy, such as oil or coal, generates important emissions of sulfur dioxide (SO2), carbon monoxide (CO), among others. On the other hand, the means of transport used in everyday life constitute another alarming source of contamination. A large part of these pollutants emitted into the environment is generated by automobiles [2].

In consequence, three main steps are proposed to be followed to address air pollution problems. The definition of air quality parameters that need to be controlled; the monitoring of these through the use of specific hardware and software; and the adoption of solutions aimed at reducing the concentration of harmful substances and ensuring clean long-term air [3]. As for air quality parameters, these are determined by the environmental policies outlined by the city government. The monitoring, for this experiment, is carried out by the surveillance stations that are installed in different points of the city. In the case of Bogotá, despite the efforts made by environmental authorities, academic institutions and citizens in general, the capital city of Colombia is today one of the most polluted cities in Latin America. One of the pollutants of greatest concern in the city is the particulate material, since its levels frequently exceed the air quality standards [2].

The Secretaria Distrital del Ambiente of Bogotá has the "Red de Monitoreo de Calidad del Aire de Bogotá - RMCAB", which comprises 13 fixed monitoring stations and a mobile station operating in different parts of the city, equipped with cutting-edge technology to allow continuous monitoring of the concentrations

of particulate matter (PM10, PST, PM2.5), pollutant gases (SO2, NO2, CO, O3) and weather variables precipitation, wind speed and direction, temperature, solar radiation, relative humidity, and barometric pressure [2]. Figure 1 establishes a comparison of a human hair vs PM10 and PM2.5 to observe the importance of the particulate matter.

At the global level, different sources of information have been used to mitigate and predict levels of air pollution and one of the most promising tools are the artificial neural networks, since these allow analyzing a large amount of data and making predictive models that allow forecasting future pollution levels. In this work, an artificial neural network was chosen for the prediction of statistical data and the result was an analysis of a prototype in a machine learning model for the prediction of air pollutants of particulate matter (PM) in an area of Bogotá (Kennedy).

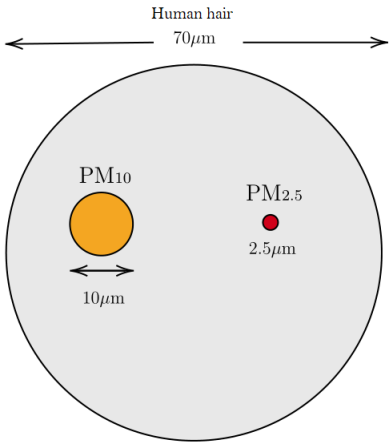

Figure 1. Comparison of a human hair vs. PM10 vs. PM2.5

2. RESEARCH METHOD

This work employs neural networks since they are widely used in the forecasting process in different areas, as can be seen in [4-6]. To carry out the design of this artificial neural network (NN), the following methodological steps were followed:

- Selection of input variables.
- Normalization.
- Architecture selection: amount of NN for hidden layer.
- Selection of the activation function.
- Selection of the learning algorithm.
- Training and validation of NN.

2.1. Selection of input variables

Data provided by the Kennedy air quality monitoring station, as shown in Table 1 was used for the selection of input variables; this station is under the administration of the Secretaría Distrital del Ambiente of Bogotá [7], through the air quality monitoring network of Bogotá RMCAB, see Figure 2. Especially, in this paper is under study the data taken from the monitoring station in Kennedy (delimited in red in Figure 2), considering the value of IBOCA (Índice Bogotano de Calidad de Aire).

The information from this station was compiled through the monthly reports published by the Bogota Air Quality Monitoring Network, which specifies the main atmospheric pollutants that the monitoring station is capable of measuring, and the average data of each pollutant, see Table 2; this paper is particularly focused on PM2.5 particulate material. The public information available from the Kennedy monitoring station was consolidated until 2018; one of the first steps taken with these data was its normalization, a procedure that consisted of cleaning the data of empty fields so that the model did not present errors.

According to the available data of Table 2 the input variables are: month, year, maximum value of PM2.5 [$\mu g/m^3$], number of exceedances to the standard of 24 hours (24H), the percentage of valid data, the percent of IBOCA favorable, moderate and regular (the sum of these three must be equal to 100%), and finally is used the data associated to the number of anomalies. The out variable corresponds to the average value of PM2.5. Table 3 presents the general statistics of the training data.

Table 1. Characteristics of the monitoring station in Kennedy RMCAB [7]

Code assigned to the station in the RMCAB		9 (Fixed station)	
Type of SVCA		Automatic	
Location		Carrera 86 # 40 - 55 Sur	
Geographical coordinates		Latitude: 04.37.29,9	Longitude: -74.09.40,7
Type of zone		Urban	
Type of station		Background	
Equipment	Parameter	Name	Analytical principle
	PM10	Met one bam 1020	Beta attenuation
	PM25	Met one bam 1020	Beta attenuation
	CO	Thermo scientific 48i	Infrared absorption with gas filter
	SO2	Tapi 100E	Pulsed fluorescence
	NO2	Thermo scientific 42i	Chemiluminescence
Location of the sampling point			Green area
Height of the sampling point			7.0 m
Height of meteorological station			10.0 m

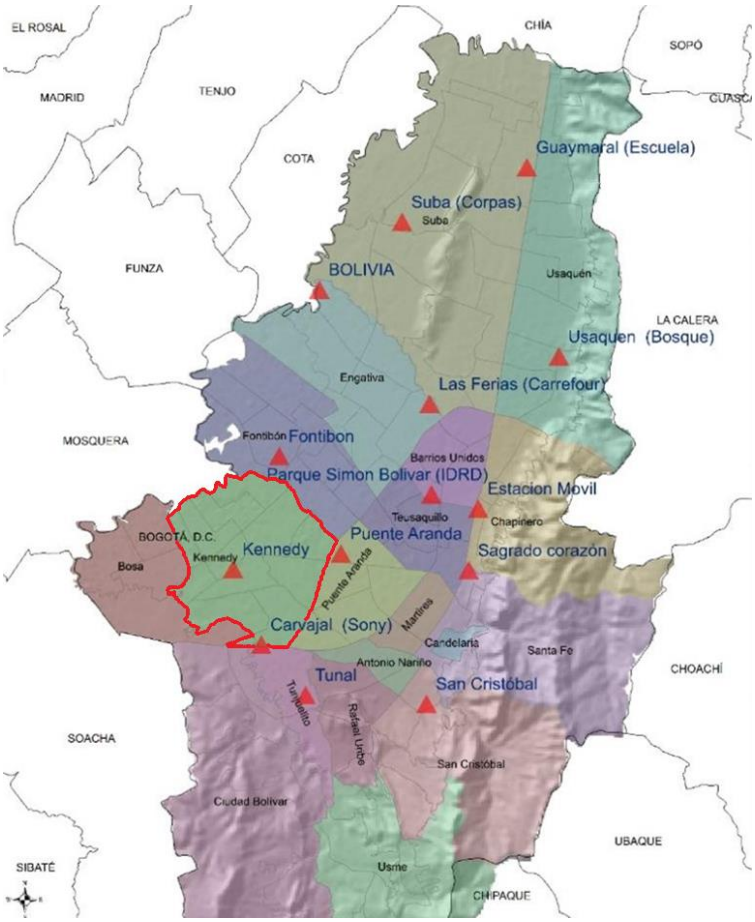

Figure 2. Stations of monitoring network in Bogotá, RMCAB [7]

Table 2. Example of data used

Month	Year	Average PM2.5	Max PM2.5	Standard 42H	Valid Data	IBOCA Value			Anomalies
						Favorable	Moderate	Regular	
1	2018	26	39	0	100	1	89	10	0
2	2018	33	48	0	100	0	63	37	0
3	2018	32.2	52	1	100	0	57	43	0
4	2018	26.6	39	0	100	0	94	6	0
5	2018	21.3	34	0	94	0	100	0	0
6	2018	16.9	28	0	100	22	78	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 3. General statistics of the input training data

Variable		Amount	Mean	STD	Min	Max
Month		29	6,310,345	3,506,152	1	12
Year		29	2.016,931034	0,842235	2016	2018
Max PM2.5		29	40,820,424	5,350,128	28.3	52
Standard 24H		29	0,135279	0,181166	0	1
Valid Data		29	96,689,655	3,140,785	87	100
IBOCA	Favorable	29	346,935	4,306,330	0	22
	Moderate	29	75,319,749	8,018,339	57	100
	Regular	29	21,216,309	10,137,875	0	43
Anomalies		29	0,034483	0,185695	0	1

2.2. Normalization

There are different scales and ranges within the collected data, although, the model could converge with the information in this way, making the training more difficult and making the model dependent on the units used for the input [8]. In addition, to normalize the values is used the Standard Normal Distribution method (1), which allows to reduce any normal distribution to the standard normal format, since it has an arithmetic average and unique standard deviations, whose values are zero and one, respectively [9].

$$z = \frac{x-\mu}{\varphi} \quad (1)$$

where x corresponds to the value of the variable to be standardized, μ represents the arithmetic average and φ the standard deviation of the variable, this ensures that there are only values within the range of 0 and 1.

2.3. Architecture selection

Cardinality selection of the hidden layers (if there are more than one) seems to be an unclear issue when designing a neural network (NN), too many units can lead to low generalization capacity. On the other hand, few units can leading to the NN do not have sufficient capacity to solve the problem [10]. Regarding the network topology, determining the number of layers that integrate it and the number of hidden neurons to be included in each layer is a complex task that directly affects the generalization capacity of the model. Since every neural network necessarily has an input layer which receives external stimuli, the problem is limited to establishing the number of extension of the hidden layers [11].

Although it has been demonstrated that the universal approaching property of MLP (Multilayer Perceptron) network functions requires a maximum of two hidden layers, in most cases a single hidden layer is sufficient to achieve optimal results. Lippmann [12] considers that networks with a single hidden layer are sufficient to solve arbitrarily complex problems, provided that the hidden layer includes at least three times the number of input nodes. Meanwhile, Hecht-Nielsen and Lippman apply an extension of Kolmogorov theorem to show that a network with a single hidden layer integrated $2N + 1$ neurons and transfer function of continuous, non-linear and monotonically increasing is sufficient to compute any continuous function of N input variables [13].

Usually, ad hoc rules are used to determine the number of hidden neurons in each layer. Although they are not mathematically justifiable, they have shown good behavior in various practical applications. Masters [14] proposed a method that he called the geometric pyramid rule, which is based on the assumption that the number of neurons in the hidden layer must be less than the total number of input variables, but greater than the number of output variables. It is considered that the number of neurons in each layer follows a geometric progression, such that for a network with a single hidden layer, the number of intermediate neurons must be close to $\sqrt{N \cdot M}$, where N is the number of variables of input and M the total output neurons; this project takes a total of 9 input variables and one output; thus, 9 neurons were defined in the hidden layer according to [10].

2.4. Selection of the activation function

In both artificial and biological neural networks, a neuron not only transmits the input it receives. There is an additional step, an activation function, which is analogous to the action potential rate [15]. There are many activation functions, for this project is used the activation function called Rectified Linear Unit abbreviated as ReLU, which is defined in (2) and represented in Figure 3.

$$F(x) = \max(0, x) \quad (2)$$

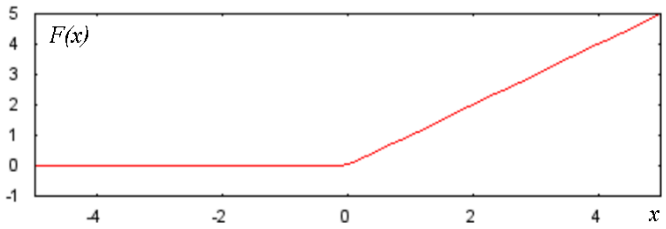

Figure 3. ReLU activation function

The superiority of ReLU is based on empirical research, probably because it has a more useful range of response capacity [16]. In other words, ReLUs allows all positive values to pass without changing them, but assign all negative values to 0. Although there are even more recent activation functions, most of the current neural networks use ReLU or one of its variants [15]. In fact, any mathematical function can be used as an activation function. Suppose that Figure 3 represents the activation function (ReLU, sigmoid or any other), to define the activation function of the neural network model, simply using the one provided by the TensorFlow library [16].

2.5. Selection of the learning algorithm

For modeling the Artificial Neural Network it was used the Keras library [17], which is written in Python, which supports working together with TensorFlow [18] and allows the modeling of artificial neural networks. Keras has two types of models; for this project, it was decided to use the sequential type, defined as a pipeline with its raw data entered in the lower part and the predictions that come out in the upper part. This is a useful conception in Keras, since they were traditionally associated with a layer can also be divided and added as separate layers [19]. The compilation requires that the parameters of the network be specified, as well as the optimization algorithm to be used to train the network and the function used to evaluate the network [19]. Figure 4 shows a fragment of the neural network modeling code using as mentioned above, the Keras and Tensorflow libraries together.

For this case, it was decided to use the RMSprop (Root Mean Square Propagation) optimizer, which is an algorithm used for complete batch optimization [20]. RMSprop tries to solve the problem of the possible wide variation in magnitude of the gradients. Some gradients can be small and others huge, which is a very difficult issue trying to find a unique global learning rate for the algorithm. This adjustment is useful for the support points and flat segments, since large enough steps are taken, even with small gradients; the step size adapts individually with time, so that learning is accelerated in the required direction [21].

Another parameter assigned for the compilation is the loss function, which indicates the value of the prediction error which consists of the sum of all the errors obtained; the metrics value was also defined, which consists of the list of metrics to be evaluated by the model during training and testing. Figure 5 shows a code fragment with the configuration of the aforementioned parameters. Making the respective configurations of the neural network as result it is obtained the report shown in Figure 6, where the layers are displayed, the number of neurons in each layer and the total number of training parameters.

```
def build_model():
  model = keras.Sequential([
    layers.Dense(9, activation=tf.nn.relu,
    input_shape=[len(train_dataset.keys())]),
    layers.Dense(2, activation=tf.nn.relu6),
    layers.Dense(1)
  ])
```

Figure 4. Fragment of the neural network modeling code

```
model.compile(loss='mean_squared_error',
            optimizer=optimizer,
            metrics=['mean_absolute_error',
            'mean_squared_error'])
```

Figure 5. Parameters configuration code

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 9)	90
dense_1 (Dense)	(None, 2)	20
dense_2 (Dense)	(None, 1)	3

Total params: 113
Trainable params: 113
Non-trainable params: 0

Figure 6. Model summary

2.6. Training and validation

In the context of neural networks, learning can be seen as the process of adjusting the free parameters of the network [22, 23]. Starting from a set of random synaptic weights, the learning process looks for a set of weights that allow the network to correctly develop a certain task. The learning process is iterative, in which the solution is refined until reaching a sufficiently good level of operation [24]. The goal of any training algorithm is to minimize the mean square error MSE (or any other loss function), but experience has shown that networks tend to over adjust data [25]. For this reason, the collected data were divided into 80-20: 80% of the data was randomly divided to be used as training data for the neural network and the remaining 20% for the testing or validation stages.

3. RESULTS

Two different machines were used for modeling the neural network: the first is a Windows desktop computer with 8GB of RAM and AMD Ryzen 2600 processor with six cores at 3.40GHz; the development environment of Python 3 and Junyper was installed on this machine. In the other development environment, a Google Collaborative instance was managed where resources are handled according to availability in a virtual machine with Junyper. The result of the error in both machines was almost the same; first machine MSE 3.56 and the second MSE of 3.67.

During the training of the model, it is a suitable practice to verify the iterations while training the artificial neural network, this to stopping the iterations (early stop), since this prevents the model from being overtrained more than necessary. Initially, it was defined 5000 epochs for training process, Figure 7 shows that after approximately 3000 epochs the error does not decrease, instead, it remains stable [16]. In addition, for the first iterations the validation error is less than the training error, while for the last iterations the opposite occurs.

In the result of Figure 7, the training process stops until finalizing the total number of epochs, to accelerate the training process as well as to avoid the overfitting the variable “early stop” is used to verify every 50 epochs if the error was reduced, then the training is stopped, otherwise, it would continue. The result using this option is presented in Figure 8, where it can be observed that at the end of the training process the validation and training error tend to be the same. Finally, Figure 9 shows the result of the prediction of PM2.5 particulate material removing outliers. This result shows that the prediction is achieved for a limited range of data, which may be related to the low amount of data available for training.

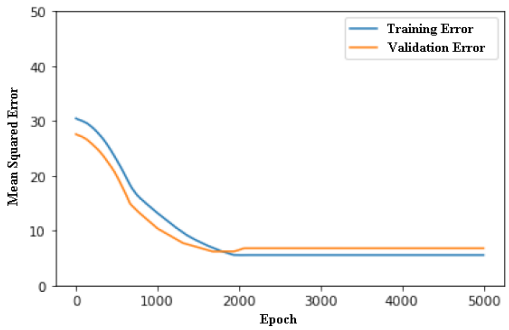

Figure 7. Performance training for PM2.5 data using 5000 epoch

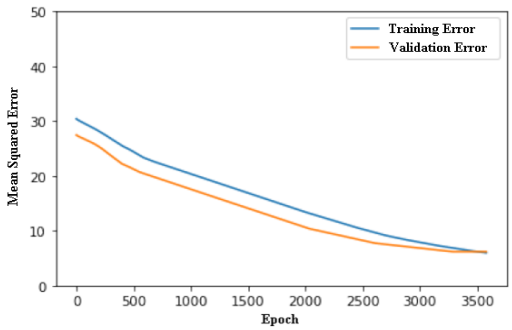

Figure 8. Performance training for PM2.5 data using early stop

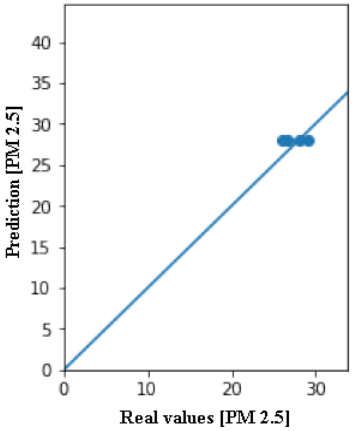

Figure 9. Prediction results

4. CONCLUSION

Currently, there is a suitable amount of libraries that allow the design and modeling of artificial neural networks, where it has been shown that Machine Learning techniques are a suitable tool for modeling prediction and classification problems. In the first stage of the project, from the information collected, the empty fields were cleaned to avoid errors and then normalized. This aspect was of importance to achieve a satisfactory result. In this regard, at the development stage, the standard normal distribution method was used to normalize the input data.

As mentioned above, the data was divided in such a way that records were kept for both training and testing; after making the adjustments on the training of the model it was possible to reduce the final MSE from 30 to 3.56 in the prediction of the average concentration of PM2.5 pollutants. The design of the neural network was based on the rule of the geometric pyramid of the hidden layers obtaining a suitable result for MSE. When there is not much training data, according to literature an alternative is to use a small network with few hidden layers to avoid overfitting. Another useful alternative to avoid overfitting is the early stop, which was considered in this work. The prototype of the neural network applied to predict the PM2.5 pollutants, is an example of the application of Machine Learning methods in the different aspects that affect human beings. The result of this work can be improved to achieve a better prediction for a large value of data.

ACKNOWLEDGEMENTS

The authors thank the Faculty of Engineering of the Universidad Distrital Francisco José de Caldas, especially the Doctorate in Engineering and the specialization in Software Engineering.

REFERENCES

[1] M. G. C. Januchs, "Aplicación de técnicas de Inteligencia Artificial a la predicción de contaminantes atmosféricos," Ph.D. dissertation, Universidad Politécnica de Madrid, Madrid, Spain, pp. 1-225, 2012.
[2] J. A. Z. Pena, "Estrategias para mitigar la contaminación del aire en zonas aledañas a grandes avenidas de Bogotá," M.Sc. dissertation, Universidad Nacional de Colombia, Bogotá, Colombia, pp. 1-129, 2016.
[3] Spider Urban Management Platform, "Soluciones para la contaminación atmosférica en las smartcities," 2019. Available: https://www.urbansolutions.es/es/blog/119-soluciones-para-la-contaminacion-atmosferica-en-las-smart-cities.
[4] R. Kumar V. and P. Dixit, "Daily peak load forecast using artificial neural network," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 4, pp. 2256-2263, 2019.
[5] S. Barhmi and O. El Fatni, "Hourly wind speed forecasting based on support vector machine and artificial neural networks," *International Journal of Artificial Intelligence*, vol. 8, no. 3, pp. 286-291, 2019.
[6] N. H. Abd-Rahman and M. H. Lee, "Artificial neural network forecasting performance with missing value imputations," *International Journal of Artificial Intelligence*, vol. 9, no. 1, pp. 33-39, 2020.
[7] Secretaría Distrital de Ambiente de Bogotá, "Red de Monitoreo de Calidad del Aire de Bogotá – RMCAB," 2019. Available: http://ambientebogota.gov.co/red-de-calidad-del-aire.
[8] I. Nurhaida, et al., "Implementation of deep neural networks (DNN) with batch normalization for batik pattern recognition," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 2, pp. 2045-2053, 2020.

[9] C. S. Pinto and D. C. Galarza, "Fundamentos Básicos de Estadística," Quito, Sin editorial, pp. 1-224, 2017.
[10] L. L. C. Peralta, et al., "Aplicación de Redes neuronales artificiales para la predicción de calidad de aire," *Mecánica Computacional*, vol. XXVII, pp. 3607-3625, 2008.
[11] B. B. Bezabeh and A. D. Mengistu, "The effects of multiple layers feed-forward neural network transfer function in digital based ethiopian soil classification and moisture prediction," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 4, pp. 4073-4079, 2020.
[12] R. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Magazine*, vol. 4, no. 2, pp. 4-22, 1987.
[13] R. Hecht-Nielsen, "Counterpropagation networks," *Applied Optics*, vol. 26, no. 23, pp. 4979-4984, 1987.
[14] T. Masters, "Practical Neural Network Recipies in C++," Morgan Kaufmann, pp. 1-493, 2014.
[15] R. Russel, "Redes Neuronales," CreateSpace Independent Publishing Platform, pp. 1-90, 2018.
[16] R. Flórez and J. Fernández, "Las Redes Neuronales Artificiales," *Netiblo*, pp. 152, 2008. [Online]. Available: https://dialnet.unirioja.es/servlet/libro?codigo=395241.
[17] Keras, "Keras Documentation," 2019. Available: https://keras.io/.
[18] TensorFlow, "Basic Regression: predict fuel efficiency," 2019. [Online]. Available: https://www.tensorflow.org/tutorials/keras/basic_regression.
[19] Unipython, "Cómo desarrollar modelos de Deep Learning con Keras," 2018. [Online]. Available: https://unipython.com/como-desarrollar-modelos-de-deep-learning-con-keras/.
[20] C. Igel and M. Hüsken, "Improving the Rprop Learning Algorithm," *Proceedings of the Second International Symposium on Neural Computation*, pp. 115-121, 2000.
[21] R. V. K. Reddy, et al., "Handwritten Hindi Digits Recognition Using Convolutional Neural Network with RMSprop Optimization," *Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, pp. 45-51, 2018.
[22] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423-1447, 1999.
[23] D. T. V. Dharmajee-Rao and K. V. Ramana, "A Novel Approach for Efficient Training of Deep Neural Networks," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 11, no. 3, pp. 954-961, 2018.
[24] L. F. Bertona, "Entrenamiento de Redes Neuronales basado en Algoritmos Evolutivos," Trabajo de Grado, Universidad de Buenos Aires, Argentina, pp. 1-253, 2005.
[25] J. A. V. Torres and J. A. D. Rivera, "Entrenamiento de una red neuronal multicapa para la tasa de cambio euro-dólar (EUR/USD) Training a multilayer neural network for the Euro-dollar (EUR/USD) exchange rate," *Ingeniería e Investigación*, vol. 27, no. 3, pp. 106-117, 2007.