Hindawi
Discrete Dynamics in Nature and Society
Volume 2020, Article ID 2792481, 14 pages
https://doi.org/10.1155/2020/2792481

Research Article

Evaluation of Key Parameters Using Deep Convolutional Neural Networks for Airborne Pollution (PM10) Prediction

Marco Antonio Aceves-Fernández, Ricardo Domínguez-Guevara, Jesus Carlos Pedraza-Ortega, and José Emilio Vargas-Soto

Faculty of Engineering, Autonomous University of Queretaro, 76010 Queretaro, Mexico

Correspondence should be addressed to Marco Antonio Aceves-Fernández; marco.aceves@gmail.com

Received 7 November 2019; Accepted 4 January 2020; Published 10 February 2020

Academic Editor: Stefan Balint

Particulate matter with a diameter less than 10 micrometers (PM10) is today an important subject of study, mainly because of its increasing concentration and its impact on environment and public health. This article summarizes the usage of convolutional neural networks (CNNs) to forecast PM10 concentrations based on atmospheric variables. In this particular case-study, the use of deep convolutional neural networks (both 1D and 2D) was explored to probe the feasibility of these techniques in prediction tasks. Furthermore, in this contribution, an ensemble method called Bagging (BEM) is used to improve the accuracy of the prediction model. Lastly, a well-known technique for PM10 forecasting, called multilayer perceptron (MLP) is used as a comparison to show the feasibility, accuracy, and robustness of the proposed model. In this contribution, it was found that the CNNs outperforms MLP, especially when they are executed using ensemble models.

1. Introduction

In recent years, the concern over the environment has been increasing rapidly. This is mainly due to human activity. Research about the environment has been a topic of public interest because of the recent increase of findings about the threats pollution has in many aspects of nature.

Particulate matter (PM) is the name of all the organic and inorganic particles which are suspended in the air we breathe. The particles with a diameter less than 10 micrometers are called PM10. In particular, these subsets of particles are of our interest because they have been recognized as an important factor on environmental pollution and public health. It has been found that PM10 takes part on the development of cardiovascular diseases, that at the same time generates more health problems [1]. In Figure 1, a diagram displaying the respiratory system and where PM10 get trapped, as well as PM with diameter less than 2.5 micrometers (PM2.5) and PM with diameter less than 0.1 micrometers (PM0.1), is shown.

The elements that are present in PM10 concentration have been studied in [3] where they describe the percentages of organic compounds on a 24 hour sample of PM.

Having accurate models that let us know the behavior of PM10 concentrations may allow to prevent the exposition of people to harmful environments, thus reducing their probabilities to develop cardiovascular diseases.

Previous approaches to the problem of PM10 modeling include the usage of artificial neural networks, fuzzy logic, and evolutionary computation. In [4] is implemented a PSO to a neurofuzzy method to enhance the modeling performance.

On the other hand, convolutional neural networks (CNNs) are one of the tools of artificial intelligence that can be used to perform the task of modeling and forecasting PM10 concentrations. CNNs have demonstrated to outperform other neural networks architectures in many applications [5, 6]. The usage of CNN has been widely used for image classification [7, 8], not so widely explored for prediction, although some studies have shown promising results [9].

In this work, an extended study of the application of CNN to PM10 forecasting is presented. The main

characteristic of a CNN is the convolutional layer; this layer consists upon the application of a filter to the input data. In this way, the filter returns as an output some features that may not be evident in the raw data. After that procedure, the next process is the same one of a neural network (NN) with neurons that are connected in layers to compute a desired output, having as input the result of the convolution layer.

Lastly, ensemble methods are an approach which can be summarized as a compilation of trained models that returns a set of predictions that must be mixed to obtain a final and unique prediction. This can be seen as a voting process where the output of each trained model has a weight in the final decision for the desired prediction. In [10] is presented an introduction to the basic concepts of ensemble methods applied to classification and regression trees. In that work is said that "ensemble methods are the strongest procedures known" for many applications [10] (p. 292).

Furthermore, in this work, a bagging ensemble model (BEM) was used to improve the accuracy of single models. The fundamental characteristic of this ensemble method is that every model contained has equal weight in the combined prediction.

The main contribution of this work is the exploration of CNN for PM10 forecasting through an ensemble method. Neither CNN has been used for PM10 modeling nor its implementation with an ensemble method.

2. Experiments

2.1. Dataset. The dataset was obtained from a public database of a Mexican government dependency called Secretariat of Environment (SEDEMA by its acronym in Spanish) [11]. This database contains measurements of different environmental variables in Mexico City. This dataset contains hourly data for each variable since 1986 [12].

From that database, 7 variables were selected (Table 1). The criterion for that selection was the amount of invalid data which each variable contained. Some of the reported values in the dataset were displayed as −99 in the cases where the sensor had a measurement error. In that sense, the selected variables for this work were the ones that presented less measurement errors in the years ranging from 2000 to 2018.

Subsequently, the data were preprocessed to replace the measurement errors with a more representative value. It must be noted that the dataset contains highly nonlinear data (Figure 2).

With the purpose of saving computing time, the tests with all the models were performed with data from January of each year. That decision was taken based on experimentation where 13 models were implemented with data from each month and the full dataset. The results suggested that January is a good subset to take as a sample for testing purposes. With this step, the data are reduced from approximately 1,200,000 samples to 100,000 samples.

In Figure 3 is shown a boxplot with data distribution of January since 2000 to 2018 to observe the behavior of PM10 through time.

In the same sense, in Figures 4 and 5 show the behavior of PM10 in the months of August and December, respectively.

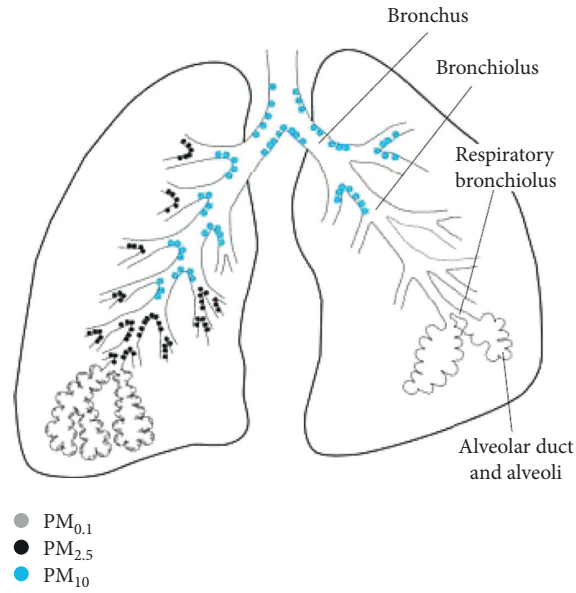

Figure 1: Representation of the respiratory system and the main areas where different types of PM get trapped; PM10 specifically get stuck on the bronchi and bronchioles [2].

Table 1: Selected variables and description.

Variable	Metric	Minimum	Maximum
PM10	Microgram/cubic meter ($\mu g/m^3$)	2	234
Temperature (TMP)	Degrees celsius (°C)	2	31.6
Wind direction (WDR)	Degrees Azimuth (AZM)	0	360
Wind speed (WSP)	Meters/second (m/s)	0.2	6.5
Relative humidity (RH)	Percentage (%)	3	94
Solar ultraviolet radiation type A (UVA)	Milliwatt/squared centimeter (mW/cm^2)	0	6.067
Solar ultraviolet radiation type B (UVB)	Minimum dose of erythema/hour (MDE/h)	0	5.558

From Figures 3–5, we can note that the variations of concentration through each month and also through all years are consistently high.

In most applications, normalization methods are reported to be an important factor that may generate improvements in the accuracy of a model [11], for that reason, a z-score normalization was performed to the input data.

Z-score normalization consists on the subtraction of all the original values (T_i) and the mean of the dataset (μ_T), then divided by its standard deviation (σ_T). In this way, the normalized dataset (N_i) is computed:

$$N_i = \frac{T_i - \mu_T}{\sigma_T}. \quad (1)$$

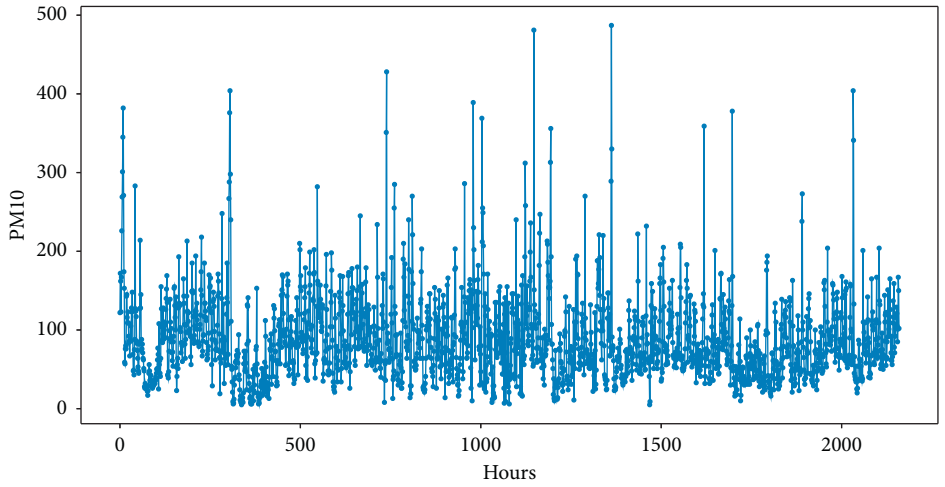

Figure 2: Graph of a sample of the dataset, displaying the hourly PM10 concentration from January 2000 to March 2000. It can be seen the high variations in concentration in that period of time.

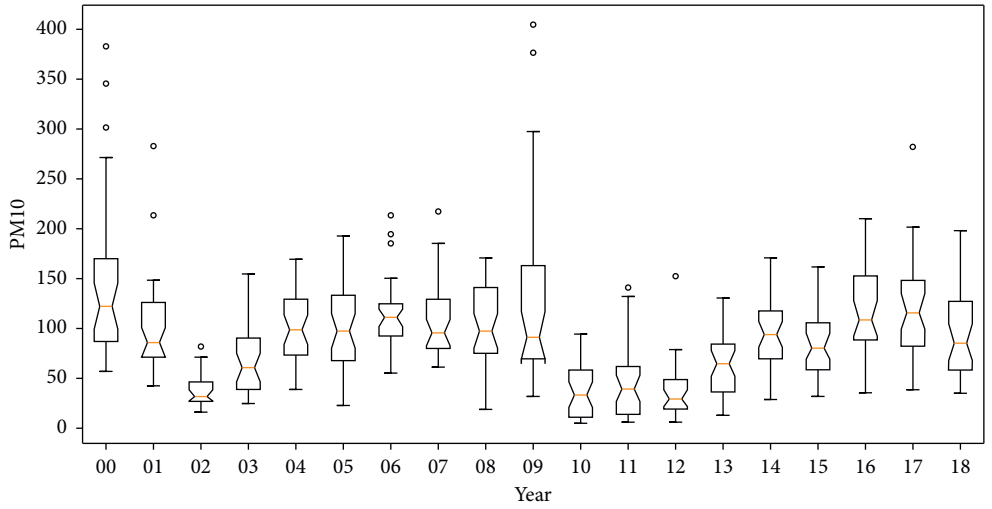

Figure 3: Behavior of PM10 concentrations by year in the month of January since 2000 to 2018. The boxplots let us see the differences in the median, quartiles, and outliers.

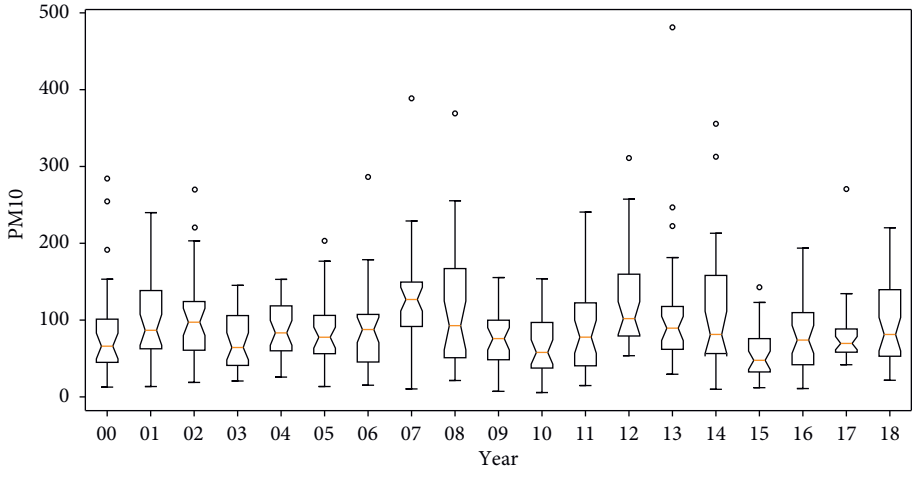

Figure 4: Behavior of PM10 concentrations by year in the month of August since 2000 to 2018. The boxplots let us see the differences in the median, quartiles, and outliers.

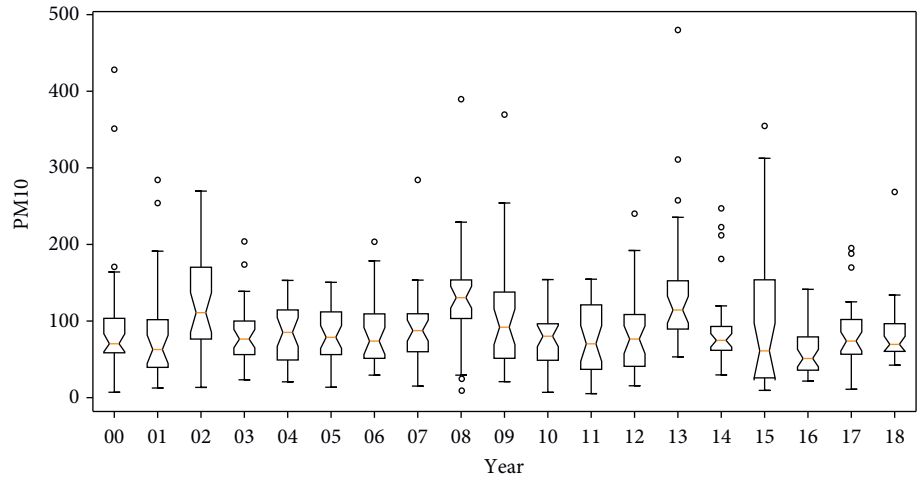

Figure 5: Behavior of PM10 concentrations by year in the month of December since 2000 to 2018. The boxplots let us see the differences in the median, quartiles, and outliers.

A characteristic of this normalization is that the resulting normalized dataset has a mean equal to 0 and a standard deviation of 1.

2.2. Multilayer Perceptron. The multilayer perceptron (MLP) architecture is one of the cases of study of this work because it is one of the most popular architectures today for PM10 forecast [13–17]. MLP usually consists of an input layer, one or two hidden layers, and an output layer. All of them with fully connected neurons between consecutive layers (Figure 6).

2.3. Convolutional Neural Networks. The architecture proposed for this work is the CNN; it was selected due to its potential of feature extraction of the input data [18]. This characteristic present in CNN is due to the application of a kernel through the input data. This kernel is basically a matrix of n rows and m columns, where n must be less than the total rows of the input data and m must be less than the total columns of the input data.

In this contribution, two types of CNN were selected: one-dimensional CNN (1DCNN) and two-dimensional CNN (2DCNN). The main difference of this architecture is the kernel used to perform the convolution on the input data. In the 1DCNN, the kernel slides in one dimension through data and in the case of 2DCNN, the kernel slides in two dimensions through data. The performance may depend on the application where the CNN is applied, which is the reason behind both models being tested in this contribution.

1DCNN consists of a convolution layer that applies a one-dimensional filter to the input data (Figure 7).

Figure 8 shows the next steps, where a flatten layer shapes the output of the convolution filters to be a one-dimensional vector. Next, a fully connected layer is added with a dropout of 15% to finally reduce to an output layer with one neuron to get the expected forecast.

This type of CNN has a movement restriction on the convolution filter, as it only can slide one dimension at a time, which restricts the feature extraction to a smaller window.

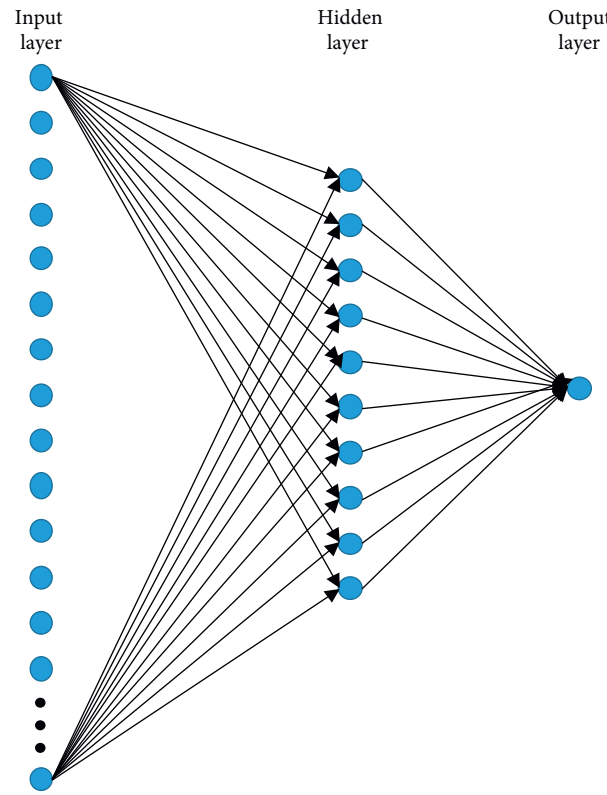

Figure 6: Architecture of multilayer perceptron. This is a simple representation of the MLP, the input layer does not have a defined number of neurons because it may depend on the desired number of previous hours in this application. The hidden layer is defined with 10 neurons, and the output layer has one neuron because it is the number of hours we want to predict.

In this work, 1DCNN was used with one convolutional layer and two convolutional layers. The second variant performs another round of transformations with new kernels to the output of the first convolution layer.

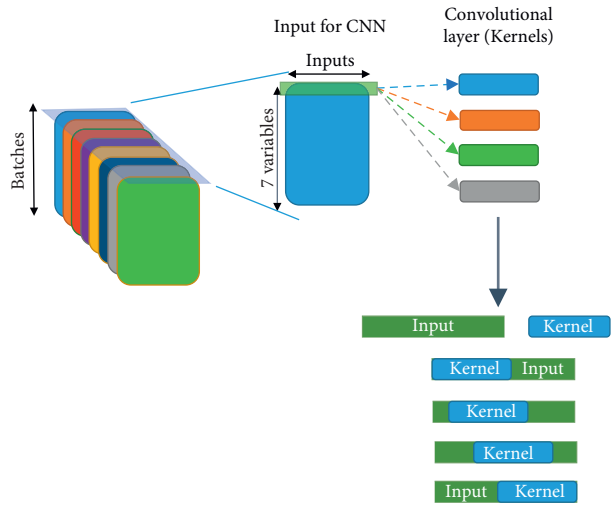

FIGURE 7: Visualization of the one-dimensional kernel sliding through the input. It is displayed the full dataset as input, it is processed in batches with size $7 \times n$ (n is the number of inputs). Then certain number of kernels is applied to each batch.

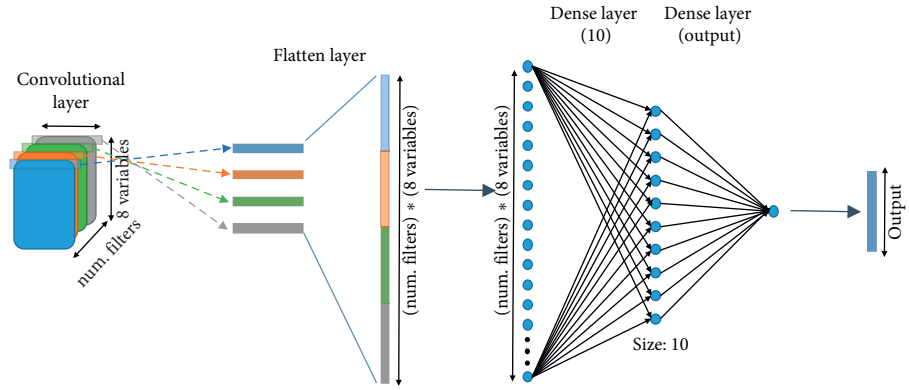

FIGURE 8: Architecture of one-dimensional convolutional neural network (1DCNN). The process described in Figure 7 takes place; then the output of the kernels applied are flattened to get into the input neurons.

On the other hand, 2DCNNs may be a good approach because it does not have the moving restriction over input data like 1DCNN. The basic architecture is the same as 1DCNN; the only difference is the kernel size applied to the input data (Figure 9).

A complete visualization of the architecture is shown in Figure 10. Notice that the procedure followed for this architecture is the same as the one for 1DCNN, with the difference of the behavior and size of the kernel.

2.4. Hyperparameters. All the internal parameters that can be modified in a model are called hyperparameters; these are crucial on the final accuracy of the model and there is where their importance relies [19]. Such hyperparameters of both 1DCNN and 2DCNN were tuned in order to determine the extent in which the models are able to forecast accurately the behavior of PM10 data.

One of the most important parameters is the activation function. This is a mathematical function that is performed in each neuron, and its task is to trigger a neuron at different intensities depending on the received input. Four well-known activation functions were tested in the architectures to determine which one is the best for this application, which are Linear, Rectified Linear Unit (ReLU), Sigmoid, and Softplus.

Some activation functions are faster to compute; also some are better for some tasks due to the way they process the input data. In [20] is made a comparison of activation functions in which the linear activation function presents a faster computation but the worst performance from all the tested variations in a classification problem.

The linear activation function basically allows the input value to be triggered as output of the neuron, shown in equation (2) and Figure 11.

$$f(x) = x. \tag{2}$$

One of its benefits is that negative values can be managed in the CNN, something that may be useful for the selected normalization method applied here.

The second activation function is the Rectified Linear Unit (ReLU). This activation function is a variation of the linear activation function, the main difference is that the negative values are transformed to zeros, and the positive

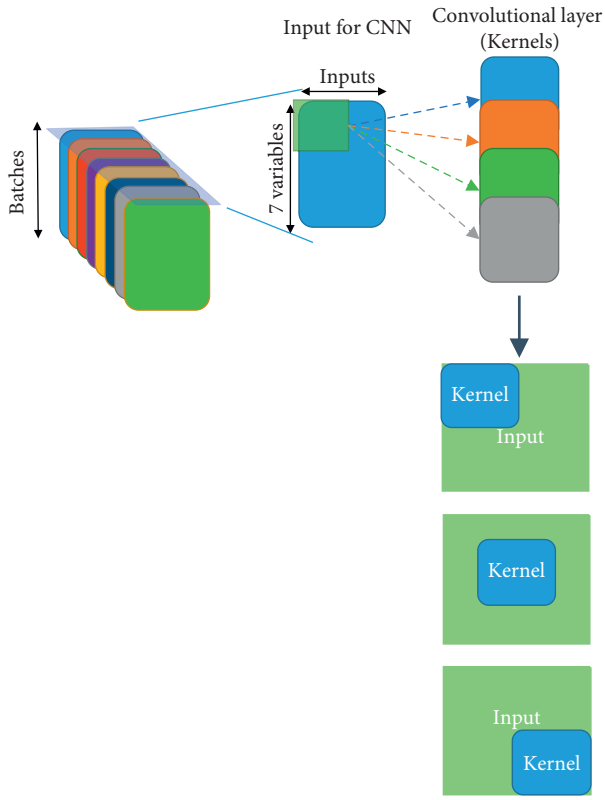

Figure 9: Visualization of the two-dimensional kernel sliding through the input. Here the kernels are two-dimensional and the output results two-dimensional too.

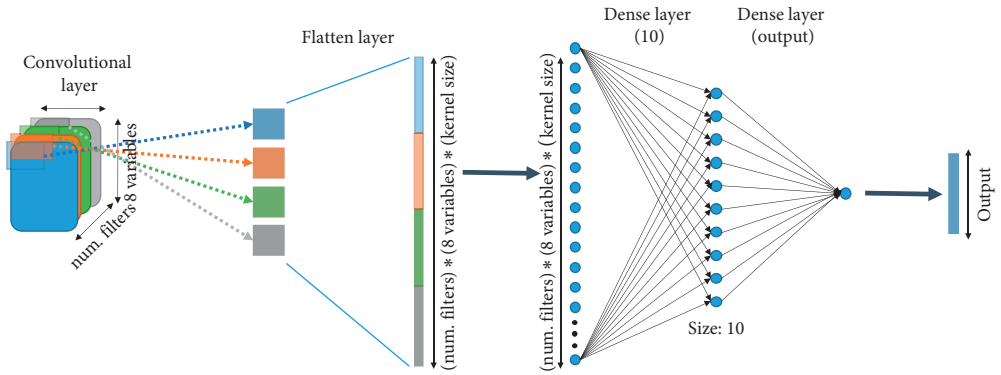

Figure 10: Architecture of two-dimensional convolutional neural network. The same process of Figure 9 takes place; then each kernel output is flattened and then stacked to get into the input neurons.

values are passed without transformation. In equation (3) and Figure 12 is shown its behavior.

$$f(x) = \begin{cases} 0, & \wedge x < 0, \\ x, & \wedge x \geq 0. \end{cases} \tag{3}$$

The training of deep neural networks with ReLU and a parameterized ReLU (PReLU) has been studied in [21–23], both doing research in a theoretical perspective of the performance of those activation functions.

Next, the sigmoid activation function has an output that ranges between 0 and 1. In equation (4) and Figure 13 is shown its behavior.

$$f(x) = \frac{1}{1 + e^{-x}}. \tag{4}$$

Lastly, the softplus activation function has an output higher than 0, it is similar to ReLU with the difference that softplus has a less abrupt adaptation on the values in the

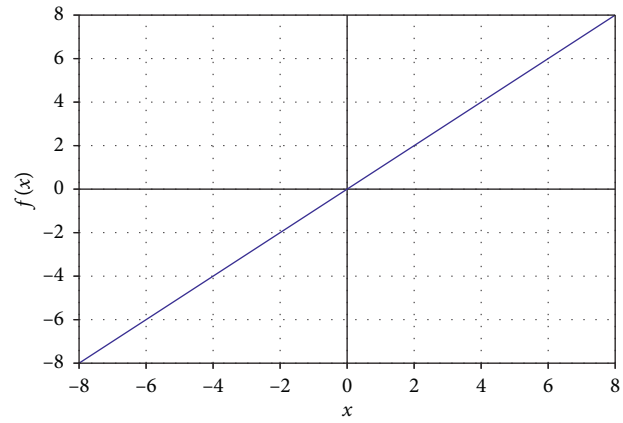

Figure 11: Linear activation function. It is one of the simplest activation functions, and it is fast to compute and does not have restrictions in the output.

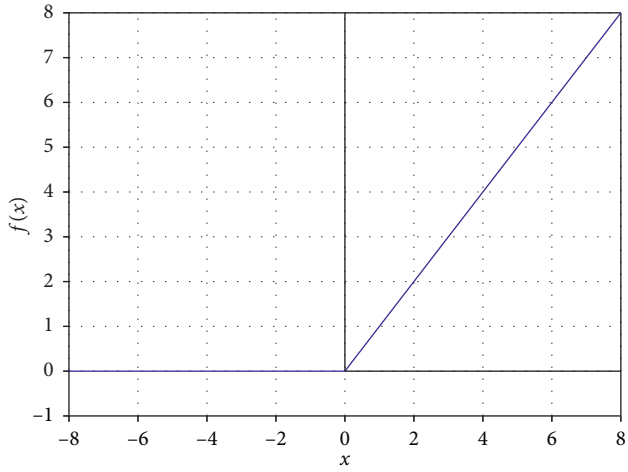

Figure 12: ReLU activation function. This activation function is also fast to compute but it allows a transformation of the input data.

limits near 0. In equation (5) and Figure 14 is shown its behavior:

$$f(x) = \ln\left(1 + e^{x}\right). \tag{5}$$

Another important hyperparameter is the optimizer for the backpropagation algorithm. Optimizers are algorithms that allow neural networks to have a better performance in every iteration. They are based on certain criteria depending on the optimizer chosen. Optimizers determine how the weights of each layer will be updated, so they are a crucial part on the performance of a neural network.

Adam is one of the most popular optimizers today. This optimizer is based on an adaptive momentum estimation. Adam is "well suited for problems that are large in terms of data and/or parameters. The method is also appropriate for nonstationary objectives and problems with very noisy and/or sparse gradients" [24] (p. 1). This was the optimizer selected for this work.

The last hyperparameter is the kernel size. This hyperparameter is exclusive of CNN because it defines the

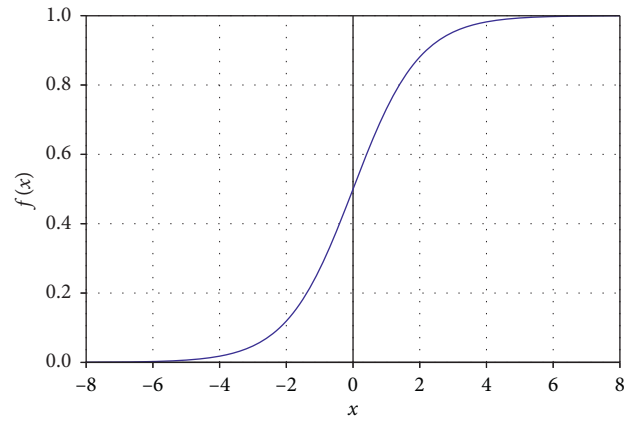

Figure 13: Sigmoid activation function. This activation function is well suited for classification problems, specifically for binary classification. It is slower to compute compared to ReLU and linear activation functions.

convolution kernel which is the main characteristic of this architecture. This consists of a two-dimensional matrix with different values in each cell. When a convolution layer is applied, the first step is the creation of kernels. There is no rule to determine the number of kernels to use of the values of each cell.

With this process, a large number of transformations are applied to the data. Here is where the features are extracted through the combination of the input variables. The size of the kernel has an effect on the process. With smaller kernels, the combination is with the nearest cells. On the other hand, larger kernels may result in a poor representation of data as they combine many cells.

2.5. Ensemble Methods. The main idea of ensemble methods is to determine a more precise prediction by means of the vote of diverse models. This has been studied in [25, 26] where it was concluded that a more general model for prediction is obtained when an ensemble method is applied, but not in all applications.

Bagging is one of the most popular ensemble methods. It consists on the creation of diverse trained models and then assigning an equal voting importance to each model to finally compute the prediction of the ensemble (Figure 15). This is obtained by the computation of the mean of the predictions of each individual model in the ensemble.

In comparison with other ensemble methods, bagging ensemble is more consistent with the results [26]. Stacking ensemble method, as an example, returns highly variable results, in some cases increasing the prediction accuracy and in other ones giving an inaccurate prediction even if all models in the ensemble have high accuracy. In contrast, bagging ensemble returns a better performance most of the times even if it is a slightly improvement [26].

2.6. Metrics. The use of reliable metrics is essential to probe the performance of the results. Is also important to have metrics commonly used with the purpose of making

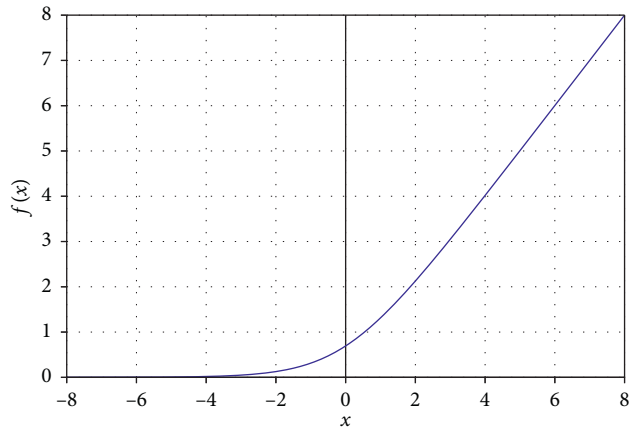

Figure 14: Softplus activation function. This one is also slower to compute than ReLU and linear activation function, but is well suited for regression applications and does not have a drastic change in the output values like ReLU.

comparisons with other works. In this work, the metrics used are root mean square error, mean absolute percentage error, index of agreement, and coefficient of determination.

In Table 2 are shown the formulas to obtain each metric as well as the values in which they range.

3. Results

In this study, several architectures were tested for MLP and CNN. To have general results, each value of RMSE, MAPE, IOA, and R^2 shown in this section is the average of three runs of the experiment. For every architecture, it was performed an ensemble with all the variants created. In addition, the best ensemble of all the set of architectures was determined. In order to get the best ensemble, the Root Mean Square Error (RMSE) was taken in account, so the algorithm used for this purpose consisted on discarding one architecture and computing the RMSE of the resulting ensemble until the chosen architectures showed a higher RMSE.

In the case of MLP, there were implemented 40 different models with the combinations of the activation functions and number of neurons in the hidden layer. The number of neurons selected are the ones in the set $A = \{2, 4, 6, 8, 10, 12, 14, 16, 18, 20\}$, which means there were 10 models for each activation function.

In Table 3 is presented a summary with the average results for each activation function, all models, and final ensemble.

From the 40 architectures, the best ones are presented in Table 4; it is clear that the best architectures are the ones with the linear activation function.

Once the evaluation of every architecture was performed, a simplified model with selected architectures for the ensemble was obtained. The results are presented in Table 5 in which are presented the models contained in the ensemble and the result of the ensemble. Note that the results are not of the individual models.

In Table 6 is presented a summary of the results for the 1DCNN with one convolutional layer by activation function. The metrics presented are the average of all the architectures tested. Those architectures include the combinations of the four activation functions with the kernel sizes ranging from 2 to 7. That means there are 24 models in total.

The best architectures are presented in Table 7. Again it can be seen that linear activation function was performed better in this case.

The selected architectures for the ensemble are presented in Table 8. In this ensemble 6 models provided the higher performance.

In Table 9 is presented the summary of results for every combination of 1DCNN with two convolutional layers. In this case, the activation functions were combined with two kernel sizes ranging from 2 to 7. Not all the combinations were possible due to limitations on the dimensionality reduction on each layer.

The best architectures included a small kernel size and a medium size kernel. The best ones are presented in Table 10.

A combination of 13 models was the ensemble that performed better in this case. In Table 11 are presented the selected architectures next to the metrics of the ensemble.

The resulting ensemble of 1DCNN with two convolutional layers is diverse. It contains at least two architectures of each activation function, but it is noted that most of the kernel sizes on both layers are small. In Table 11 is presented this ensemble.

In Table 12 is presented the summary of results for every combination of 2DCNN by activation function. Those architectures are the combination of the activation functions with the first kernel size in the set $B = \{10, 20, 30, 40, 50\}$ and the second kernel size ranging from 2 to 7. There were 120 models in total.

The selection of the best architectures is presented in Table 13 where it is noted that in the first kernel, the value of 50 is repeated 2 times and in the second kernel, the value of 3 is repeated 2 times.

In Table 14 is presented the ensemble for 2DCNN with the selected models that presented a higher performance. In this Case, 12 models were selected.

The final ensemble of 2DCNN architectures is the one with better results from the single architecture ensembles.

Once every variant of architectures were tested, the four resulting sets of ensembles previously obtained were introduced into the final ensemble model. The purpose of this is to determine which the best ensemble for those architectures is. In Table 15 is presented the average results of each individual model (Mean) with only the January data and also with the full dataset. Also, it is presented the result of the ensemble containing all models.

Finally, in Table 16 is presented the resulting ensemble of this work. This is the one the best performance and it is the model that actually represents in a better sense the full dataset used.

Even when the RMSE highly decreases, an analysis of error behavior between the ensemble predictions and the real value reveals an important fact. The error is increased due to some outliers that affect the measurement. In

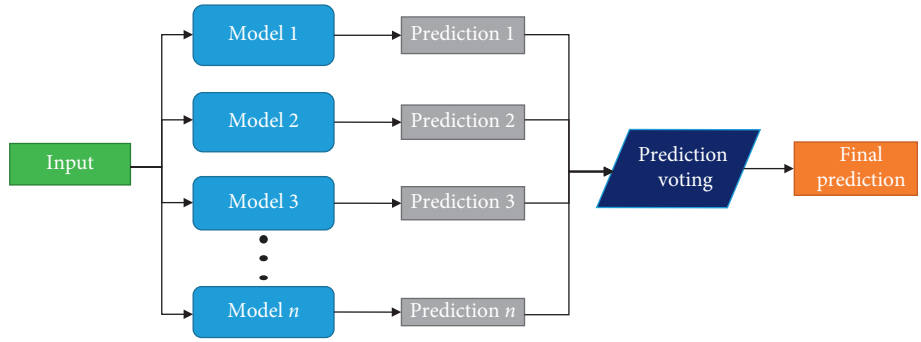

Figure 15: Bagging ensemble diagram. There are 4 stages in this process. At first is the input to each model in the ensemble. Next, the output of each model is computed, followed by the voting process to finally get the prediction of the ensemble.

Table 2: Metrics used to evaluate the results.

Metric	Formula	Range of values	Preferred values
Root mean squared error (RMSE)	$\mathrm{RMSE} = \sqrt{(1/n)\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$	$0 \le x \le \infty$	Lower
Mean absolute percentage error (MAPE)	$\mathrm{MAPE} = (1/n)\sum_{j=1}^{n}\lvert y_j - \hat{y}_j / y_j \rvert$	$0 \le x \le 1$	Higher
Index of agreement (IOA)	$\mathrm{IOA} = 1 - (\sum_{j=1}^{n}(y_j - \hat{y}_j)^2 / \sum_{j=1}^{n}(\lvert \hat{y}_j - \mu_T \rvert + \lvert y_j - \mu_T \rvert)^2)$	$0 \le x \le 1$	Higher
Coefficient of determination (R^2)	$R^2 = (\sum_{i=1}^{n}(x_i - x')(y_i - y') / \sqrt{\sum_{i=1}^{n}(x_i - x')^2 \sum_{i=1}^{n}(y_i - y')^2})^2$	$0 \le x \le 1$	Higher

Table 3: Summary of 40 MLP architectures.

Architecture	Activation Function	Mean			
		RMSE	MAPE	IOA	R^2
MLP	Linear	19.764	0.2369	0.8695	0.5866
	ReLU	20.3885	0.2461	0.8576	0.56
	Sigmoid	20.3289	0.2493	0.8638	0.5626
	Softplus	20.105	0.2408	0.8608	0.5722
Total		20.1466	0.2433	0.8629	0.5703

Table 4: Three best MLP architectures.

Architecture	Activation function	Number of neurons	RMSE	MAPE	IOA	R^2
MLP	Linear	10	19.758	0.2369	0.8713	0.5869
		16	19.7369	0.2353	0.8673	0.5878
		20	19.716	0.2352	0.8698	0.5886

Table 5: Final ensemble for MLP architectures.

Architecture	Activation function	Number of neurons	RMSE	MAPE	IOA	R^2
MLP	Linear	10	19.7213	0.239	0.863	0.582
		12				
		14				
		16				
		20				
	Softplus	14				

The architectures that presented a better combined performance are presented here, next to the metrics of the ensemble. The linear activation function is the most repeated in this ensemble.

Figure 16 is shown a boxplot where the distribution of errors between the predictions and real data is displayed.

Is clear that most of the error values are ranging in lower values. Taking the mean of the absolute errors returns a result of 10.9 $\mu g/m^3$ and the median is 8.52 $\mu g/m^3$. Then, we notice that some of the outliers are considerably large, so that increases the RMSE measurement.

By performing the same evaluations with the ensembles of each architecture, it is evident that this behavior is present in all of them, as shown in Figure 17.

Table 6: Summary of 24 1DCNN with one convolutional layer architectures.

Architecture	Activation function	Mean			
		RMSE	MAPE	IOA	R^2
CNN	Linear	19.4408	0.2375	0.8698	0.6
	ReLU	22.3676	0.2702	0.8297	0.4703
	Sigmoid	21.0785	0.2626	0.8575	0.529
	Softplus	21.0008	0.2574	0.857	0.5325
Total		20.9719	0.2569	0.8535	0.533

Table 7: Results of 1DCNN with one convolutional layer architectures and full ensemble.

Architecture	Activation function	Kernel size	RMSE	MAPE	IOA	R^2
CNN	Linear	2	19.3736	0.2395	0.8782	0.6028
		3	19.3865	0.2365	0.8725	0.6023
		6	19.4022	0.2371	0.8695	0.6016

Table 8: Final ensemble for 1DCNN with one convolutional layer architectures.

Architecture	Activation function	Kernel size	RMSE	MAPE	IOA	R^2
CNN	Linear	2	19.0053	0.2315	0.8805	0.624
		5				
	Sigmoid	6				
		3				
	Softplus	2				
		3				

Table 9: Summary results of 1DCNN with two convolutional layers architectures.

Architecture	Activation function	Mean			
		RMSE	MAPE	IOA	R^2
CNN	Linear	19.557	0.2411	0.8694	0.5952
	ReLU	23.0543	0.2759	0.8278	0.4373
	Sigmoid	21.2224	0.2602	0.8633	0.5229
	Softplus	21.2177	0.2586	0.8541	0.5233
Total		21.2628	0.2589	0.8537	0.5197

Table 10: Best architectures of 1DCNN with two convolutional layers.

Architecture	Activation function	First kernel size	Second kernel size	RMSE	MAPE	IOA	R^2
CNN	Linear	2	2	19.3009	0.233	0.8723	0.6058
		2	6	19.3982	0.2319	0.8635	0.6018
		4	3	19.3451	0.2348	0.8745	0.604

Table 11: Architectures present in 1DCNN ensemble.

Architecture	Activation function	First kernel size	Second kernel size	RMSE	MAPE	IOA	R^2
CNN	Linear	2	2	18.9024	0.215	0.8762	0.6093
		2	3				
		2	5				
		2	6				
	ReLU	4	2				
		5	2				
		3	3				
		6	2				
	Sigmoid	2	2				
		2	3				
		2	4				
	Softplus	2	2				
		2	3				

Table 12: Summary of results for 2DCNN.

Architecture	Activation function	Mean			
		RMSE	MAPE	IOA	R^2
CNN	Linear	19.6121	0.2407	0.8697	0.5928
	ReLU	22.1328	0.2706	0.8232	0.4814
	Sigmoid	26.0486	0.3214	0.6712	0.3927
	Softplus	22.7749	0.2649	0.7656	0.4798
Total		**22.6421**	0.2717	0.7842	0.4867

Table 13: Three best models of 2CNN architectures.

Architecture	Activation function	First kernel size	Second kernel size	RMSE	MAPE	IOA	R^2
CNN	Linear	20	3	19.2497	0.2314	0.8704	0.6079
		50	3	19.3115	0.2338	0.8729	0.6053
		50	6	19.2326	0.2317	0.8747	0.6086

Table 14: Final ensemble for 2DCNN architectures.

Architecture	Activation function	First kernel size	Second kernel size	RMSE	MAPE	IOA	R^2
CNN	Linear	30	2	18.7697	0.2295	0.8805	0.6221
		20	3				
		40	3				
		30	4				
		50	6				
	ReLU	40	3				
		10	4				
	Sigmoid	50	3				
		50	4				
	Softplus	10	5				
		30	5				
		20	6				

Table 15: Average result of each individual model.

Architecture	Input	RMSE	MAPE	IOA	R^2
Mean	January data	20.2219	0.2404	0.8608	0.566
	Full data	20.1178	0.2431	0.8635	0.5703
Ensemble with all architectures	January data	18.7889	0.2218	0.8793	0.6264
	Full data	18.9535	0.2266	0.8769	0.6198

Table 16: Final ensemble model with full data.

Architecture	Activation function	First kernel size	Second kernel size	Number of neurons	RMSE	MAPE	IOA	R^2
MLP	Linear	—	—	10	14.9469	0.2433	0.8994	0.6789
		—	—	12				
	ReLU	3	3	—				
		10	4	—				
CNN	Sigmoid	2	2	—				
		2	3	—				
		50	3	—				
	Softplus	2	3	—				
		20	6	—				
		30	5	—				

This final ensemble contains 10 architectures; from those, the 80% are of a convolution type. It is noted that the RMSE highly decreased for this ensemble, in comparison with the previous ensembles presented.

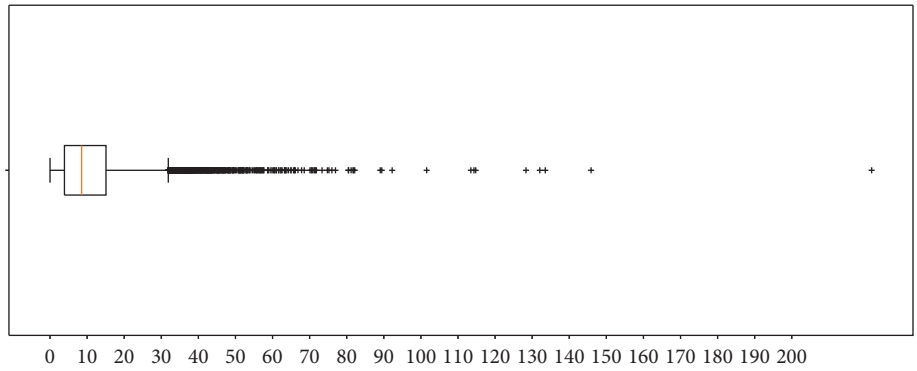

Figure 16: Distribution of absolute errors between predictions of final ensemble and real data. In the X axis is presented the absolute error of all the test set and its distribution.

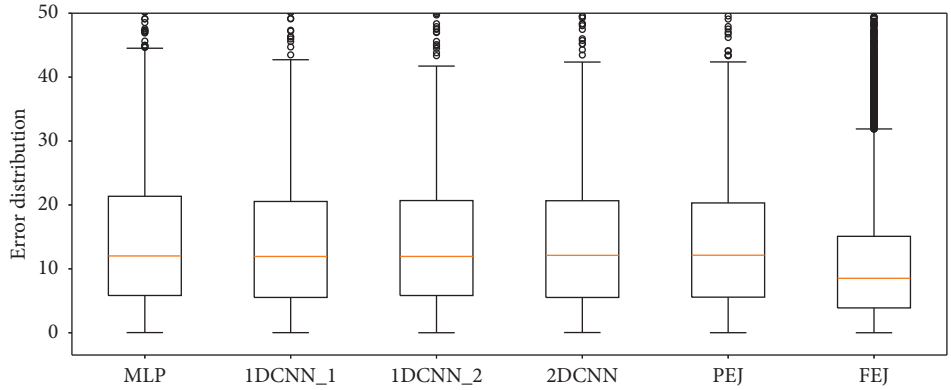

Figure 17: Error distribution in every architecture. The order of the architectures is presented by order of appearance in this work. First is the multilayer perceptron (MLP), next is the one-dimensional CNN with one convolutional layer (1DCNN_1). The third model is the one-dimensional CNN with two convolutional layers (1DCNN_2). The fourth model is the two-dimensional CNN (2DCNN). The fifth model is the partial ensemble trained with the data of the month of January (PEJ). Finally, the sixth model is the final ensemble trained with the full dataset (FEJ).

It can be seen in Figure 17 that the final ensemble trained and tested with the full dataset has a lower mean and median than the other models. Also, it presents a narrower boxplot and more outliers. Those results imply that the final model has better predictive power.

4. Discussion

By analyzing the results for each architecture, it is noticed that in the case of MLP, the best architectures had 10, 16, and 20 neurons in the hidden layer. All of them with the linear activation function. The best RMSE for MLP was 19.716 being the worst architecture of all the tested ones. This architecture was taken for comparison purposes and the results of the other three architectures are measured under this result.

In the case of 1DCNN with one convolution layer, the best results were obtained again with the linear activation function and kernel sizes of 2, 3, and 6. Being the kernel size of 2 the one with the lowest RMSE of 19.3736. This implies an improvement of approximately 1.8% against MLP.

Next, for 1DCNN with two convolution layers, the linear activation function was once again the most adequate. In the same sense, the best kernel sizes on both layers were 2, 3, 4, and 6. This is consistent with the results of 1DCNN with one convolution layer. For this architecture, the best RMSE was the one with kernel size of 2 in both convolution layers, having a value of 19.3009. This represents a 2.1% improvement against MLP.

Lastly, for 2DCNN, the linear activation function ended up with better results. The best kernel sizes for the axis of features were 3 and 6 continuing to be consistent with the results of both 1DCNN. In the case of the kernel size for the previous hours axis, the most adequate were 20 and 50. The best RMSE for this architecture was 19.2497, representing an approximate of 2.4% improvement over MLP.

From that data it can be seen that the best activation function for this application resulted to be the linear activation. Also, is noticed that smallest kernel sizes tend to be better. The improvements are not too significant but are consistently better for every convolution architecture.

For the next set, the ensembles of individual architectures had a similar behavior. For MLP the RMSE was 19.7213, being a worst result than the individual model by 0.02%. In the case of the ensemble of 1DCNN with one convolutional layer, the RMSE was 19.0053, representing a 3.6% improvement. For the 1DCNN with two convolutional

layers, the RMSE ended up as 18.9024; this implies a 4.1% improvement. Finally, 2DCNN scored a RMSE of 18.7697 being a 4.8% improvement.

After the individual ensembles, the ensemble with the mixture of architectures resulted with 80% of convolutional architectures, scoring a RMSE of 18.6875, which is a 5.2% improvement.

The last ensemble resulted was then tested with the full dataset, giving a RMSE of 14.9469. This one is a more significant improvement of 24.2%.

Looking at these results, it is evident that ensemble models boost the predictive capabilities in every case. Lastly, although the improvement may seem insufficient just with a CNN, it may be concluded that a combination of the right architectures and hyperparameters with an ensemble method represents an advantage, to greatly improve upon the results.

5. Conclusions

In this work, an extended study of convolutional neural networks and its comparison with multilayer perceptron are presented. Also, a bagging ensemble method is performed to improve the accuracy of the model. From the results presented, it can be noticed that the individual models with best accuracy were the ones with a convolution layer. Even though the differences are small, those improvements make possible to have a better model when they are combined with an ensemble method. In the final ensemble model obtained, 80% of the individual models are based on a convolutional architecture. This reveals that convolutional architectures may have a better representation for this application.

Finally, the last bagging ensemble method presented an improvement of 20% related to all the best individual models tested. This is a significant improvement that let us know that it is a good approximation in the prediction of PM10.

Data Availability

The data used are readily available via the government, and it could be downloaded by any person interested.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

The authors would like to acknowledge the support of Consejo Nacional de Ciencia y Tecnología (CONACyT) through the National Scholarship for postgraduate studies.

References

[1] Y. O. Khaniabadi, G. Goudarzi, S. M. Daryanoosh, A. Borgini, A. Tittarelli, and A. De Marco, "Exposure to PM10, NO_2, and O_3 and impacts on human health," *Environmental Science and Pollution Research*, vol. 24, no. 3, pp. 2781–2789, 2017.

[2] S. A. Cormier, S. Lomnicki, W. Backes, and B. Dellinger, "Origin and health impacts of emissions of toxic by-products and fine particles from combustion and thermal treatment of hazardous wastes and materials," *Environmental Health Perspectives*, vol. 114, no. 6, pp. 810–817, 2006.

[3] J. Wasim, I. Minasm, G. S. Euripides, M. W. Jack, L. Petros, and G. Bing, "Concentrations of aliphatic and polycyclic aromatic hydrocarbons in ambient PM2.5 and PM10 particulates in Doha, Qatar," *Journal of the Air & Waste Management Association*, vol. 69, no. 2, pp. 162–177, 2019.

[4] B. Ordóñez-De León, M. A. Aceves-Fernandez, S. M. Fernandez-Fraga, J. M. Ramos-Arreguín, and E. Gorrostieta-Hurtado, "An improved particle swarm optimization (PSO): method to enhance modeling of airborne particulate matter (PM10)," *Evolving Systems*, 2019.

[5] M. B. Beissinger, *Deep Generative Networks for Sequence prediction: Master of Science in Engineering*, University of Pennsylvania, Philadelphia, PA, USA, 2017.

[6] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *Proceedings of the 2015 IEEE International Conference on Computer Vision, CentroParque Convention Center*, Santiago, Chile, 2015.

[7] A. Krizhevsky and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems, Lake Tahoe, Nevada*, MIT Press, Cambridge, MA, USA, 2012.

[8] N. Liu, J. Han, D. Zhang, S. Wen, and T. Liu, "Predicting eye fixations using convolutional neural networks," in *Proceedings of the 2015 IEEE International Conference on Computer Vision, CentroParque Convention Center*, Santiago, Chile, 2015.

[9] A. Busia and N. Jaitly, "Next-step conditioned deep convolutional neural networks improve protein secondary structure prediction," *Special Issue of Bioinformatics*, Prague Congress Centre, Prague, Czech Republic, 2017.

[10] R. A. Berk, "An introduction to ensemble methods for data analysis," *Sociological Methods & Research*, vol. 34, no. 3, pp. 263–295, 2006.

[11] Secretaría del Medio Ambiente, 2019, https://www.sedema.cdmx.gob.mx/.

[12] Calidad del Aire, 2019, http://www.aire.cdmx.gob.mx/default.php?opc=%27aKBhnmM=%27.

[13] A. K. Paschalidou, S. Karakitsios, S. Kleanthous, and P. A. Kassomenos, "Forecasting hourly PM10 concentration in Cyprus through artificial neural networks and multiple regression models: implications to local environmental management," *Environmental Science and Pollution Research*, vol. 18, no. 2, pp. 316–327, 2011.

[14] J. B. Ordieres, E. P. Vergara, R. S. Capuz, and R. E. Salazar, "Neural network prediction model for fine particulate matter (PM 2 .5) on the US—Mexico border in El Paso (Texas) and Ciudad Juarez (Chihuahua)," *Environmental Modelling & Software*, vol. 20, no. 5, pp. 547–559, 2015.

[15] H. Abderrahim, M. R. Chellali, and A. Hamou, "Forecasting PM10 in Algiers: efficacy of multilayer perceptron networks," *Environmental Science and Pollution Research*, vol. 23, no. 2, pp. 1634–1641, 2016.

[16] M. G. Cortina–Januchs, J. Quintanilla–Dominguez, A. Vega–Corona, and D. Andina, "Development of a model for forecasting of PM10 concentrations in Salamanca, Mexico," *Atmospheric Pollution Research*, vol. 6, no. 4, pp. 626–634, 2015.

[17] A. Kurt, B. Gulbagci, F. Karaca, and O. Alagha, "An online air pollution forecasting system using neural networks," *Environment International*, vol. 34, no. 5, pp. 592–598, 2008.

[18] X. Li, Q. Ding, and J.-Q. Sun, "Remaining useful life estimation in prognostics using deep convolution neural networks," *Reliability Engineering & System Safety*, vol. 172, pp. 1–11, 2018.

[19] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. 2012, pp. 281–305, 2012.

[20] E. A. M. A. Shenouda, "A quantitative comparison of different MLP activation functions in classification," *Advances in Neural Networks—ISNN 2006*, vol. 3971, pp. 849–857, 2006.

[21] R. Arora, A. Basu, P. Mianjy, and A. Mukherjee, "Understanding deep neural networks with rectified linear units," in *Proceedings of the International Conference on Learning Representations*, Vancouver Convention Center, Vancouver, BC, Canada, May 2018.

[22] K. He, X. Shang, S. Ren, and J. Sun, "Delving deep into rectifiers : surpassing human-level performance on ImageNet classification," 2015, https://arxiv.org/abs/1502.01852.

[23] P. V. de Campos Souza, C. F. Guimaraes Nunes, A. J. Guimares et al., "Self-organized direction aware for regularized fuzzy neural networks," *Evolving Systems*, 2019.

[24] D. P. Kingma and J. L Ba, "Adam: a method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations*, San Diego, CA, USA, May 2015.

[25] I. Syarif, E. Zaluska, A. Prugel-bennett, and G. Wills, "Application of bagging, boosting and stacking to intrusion detection," *Machine Learning and Data Mining in Pattern Recognition*, vol. 7376, pp. 593–602, 2012.

[26] M. Graczyk, T. Lasota, B. Trawiński, and K. Trawiński, "Comparison of bagging, boosting and stacking ensembles applied to real estate appraisal," *Intelligent Information and Database Systems*, vol. 5991, pp. 340–350, 2010.