Songklanakarin J. Sci. Technol.
43 (3), 687-695, May - Jun. 2021

http://www.sjst.psu.ac.th

Original Article

Forecasting the PM-10 using a deep neural network

Chinawat Chairungrueang[1] and Rati Wongsathan[2*]

[1] *Department of Industrial Engineering, Faculty of Engineering, North-Chiang Mai University, Hang Dong, Chiang Mai, 50230 Thailand*

[2] *Department of Electrical Engineering, Faculty of Engineering, North-Chiang Mai University, Hang Dong, Chiang Mai, 50230 Thailand*

Received: 15 December 2019; Revised: 23 March 2020; Accepted: 9 May 2020

Abstract

The air pollutants related to PM-10 are increasingly adversely affecting people in upper Northern Thailand, especially during the annual dry season. Due to the highly nonlinear dynamics of PM-10 contributed by various factors, in this study a deep neural network (DNN) has been implemented as a tool forecasting PM-10 for air quality alerts. In its design, the time lags of PM-10 and significant meteorology conditions, as well as the well-correlated fire-hotspots as major PM-10 sources in this area, are included in the predictor set. The training hyperparameters were optimized automatically by a genetic algorithm, whereas the DNN's parameters were fine-tuned using back-propagation algorithm. Besides, regularization based on a dropout technique was employed to prevent over-fitting. In testing the proposed DNN-based PM-10 forecasting model outperformed the others. For one-day ahead forecasting it provides a good up to 85% accuracy.

Keywords: deep neural network, PM-10, genetic algorithm, dropout, machine learning

1. Introduction

For a decade, during the high haze episode in the annual dry season (January-April), people in upper Northern Thailand (UNT) have been experiencing severe air pollution related to particulate matter (PM). The PM with diameter below 10 μm, called PM-10, is used as a health indicator in this area while there are only limited monitoring stations for PM-2.5. The various biomass open-space burns and forest fires – both local and in nearby outside areas – are considered the major primary PM sources (Pasukphun, 2018). Until now, the PM-10 is traditionally measured and announced daily in the morning, which cannot serve as an advance alert of health risks. Therefore, an implementation that would accurately forecast PM-10 could play an important role among the air quality monitoring tools.

In the literature, various weather research and forecasting (WRF) models have been used to simulate advection and dispersion of PM-10. Although they can provide a high resolution, the long processing time and required heavy computing resources remain disadvantages (Amnuaylojaroen & Kreasuwun, 2011; Macatangay, Gagtasa, & Sonkaew, 2017). Aside from such deterministic models, various data-driven statistical models have been proposed to predict PM-10, including very simple (linear) models based on regression techniques and moving average methods (Pimpunchat, Sirimangkhala, & Junyapoon 2014), but also very complex (nonlinear) models like neural networks (NNs) (Wongsathan, 2018a) and adaptive neuro-fuzzy inference systems (ANFIS) (Wongsathan, 2018b). When compared, NN and ANFIS have similar performances and outperform the rest. A shallow NN (SNN) (i.e., NN having a single hidden layer) can forecast moderate PM-10 levels well, the performance deteriorates around the peak. To improve the PM-10 forecasting models, a new efficient approach to neural networks is our focus.

Recently, deep neural networks (DNN) have been applied successfully in various machine learning applications. In regression problems, these provide advantages in forecasting over the SNN despite similar numbers of hidden

*Corresponding author
Email address: rati@northcm.ac.th

nodes (Merkel, Povinelli, & Brown 2018), and over support vector machines for a small training dataset (Feng, Zhou, & Dong, 2019; Phyo & Jeenanunta 2019). Nonetheless, their performance in practical problems deteriorates, not performing as expected. The poor performance is due to improper training, caused by the vanishing gradient problem (VGP) (Hochreiter & Schmidhuber, 1997) likely to occur with a DNN, as well as over-fitting, and the approach suffers also from high computational load. However, DNN has seldom been tested for PM forecasts.

In this paper, the DNN is formulated as an alternative PM-10 forecasting model type (Section 2.3) (Section 2.2) to predict daily PM-10 one-day to one-week ahead. Before training the DNN (i.e. fitting its parameters), the DNN's hyperparameters (ones affecting the training algorithm) are optimized by using a genetic algorithm (GA) (Goldberg, 1989) (Section 3.1) (Section 2.4). GA is a metaheuristic method based on evolutionary search, and is used to substitute for laborious and slow manual tuning. Among the hyperparameters, the combination of squash and un-squash activation functions is our choice to reduce the VGP, while preserving this type of nonlinear transformations in the network. Furthermore, the dropout regularization technique (Srivastava *et al.*, 2014) (Section 2.3) is applied to overcome over-fitting and to remove co-dependencies amongst many neurons. However, the conventional dropout DNN produces a mean value instead of a certain value that would be proper in a classification tasks. For forecasting it is therefore modified. To accelerate the convergence of training while maintaining a full DNN, dropout is employed during some first epochs and is disabled for the rest. Therefore, three types of DNNs: (1) full DNN, (2) dropout DNN, and (3) dropout accelerated DNN, are implemented as candidate PM-10 forecasting models. They are associated with the set of the significant input variables composed of historical PM-10 data, the number of hotspot counts, related meteorological parameters, and seasonal trends in terms of periodic functions. The collected data for 2012-2019 (Section 2.1) are used to train, validate, and test them all. After achieving proper DNN-based PM-10 forecasting models from these experiments, their performances are evaluated through error metrics and statistical tests. The forecast results are further discussed and compared (Section 3). Section 4 presents the conclusion and directions for future work.

2. Methods

2.1 Pre-processing data

Three provinces that are severely affected by PM-10 in the UNT region, namely, Chiang Mai (CM), Chiang Rai (CR), and Mae Hong Son (MHS), are used in this case study. Some meteorological factors and correlated hotspot counts, among others, which relate to the PM-10 behavior, are included in the forecasting models. In order to select the significant predictors for the model, forward selection was used to identify an optimal set, i.e., the variable that is most significant based on correlation coefficient with PM-10 is initially used as input to a pilot SNN/DNN model, and then more predictor variables are added to the model as long as its P-value is below some pre-set level. Therefore, the input variables to the DNN included lags of historical PM-10, previous day meteorological parameters: average pressure (*P*), average temperature (*T*), average relative humidity (*RH*), and average wind speed (*WS*); previous day hotspot counts (*HSP*), and seasonal trend functions $\sin(2\pi d/120)$ and $\cos(2\pi d/120)$, where $d=\{1,\ldots,120\}$ is the index for a day during 4 months (Jan-Apr). The data were collected for 2012-2019 from various agencies. The entire data were pre-processed and cleared of missing and outlier values to transform into daily average values. These variables have a wide range of the relative changes (ratio of variance to the range) so that the SNN cannot characterize sufficiently these dynamic changes, whereas the DNN, having a higher dimensional set of hidden parameters, may overcome this problem.

2.2 DNN-based PM-10 forecasting model

In a feed-forward DNN (Figure 1), an increase in the dept (number of hidden layers) of the NN improves the performance of DNN from that of SNN. However, due to the back-propagation algorithm (BPA) used in machine learning, small derivatives from squashing activation functions (e.g., hyperbolic tangent $tanh(\cdot)$ and sigmoid $\sigma(\cdot)$) are multiplied backward iteratively, and this leads to VGP. Then the output errors fail to reach back further nodes effectively stopping the learning. Therefore, multiple hidden layers cannot be trained efficiently. This problem can be solved using non-squashing activation functions, such as the rectified linear unit (ReLU) function, $\varphi(x) = \max[0, x]$, the leaky ReLU (LReLU) function, $\varphi(x) = \max[ax, x]$, or the exponential LU (ELU), $\varphi(x) = \max[-a(e^x-1), x]$, where a is a small negative value and x is the given input, which produce comparatively large derivatives compared to the arbitrarily small values (for large x) from $tanh(\cdot)$ and $\sigma(\cdot)$.

For DNN-based PM-10 forecasting (Figure 1), the output of L-hidden layer DNN ($L > 2$) each having a dimension of $N^{(l)}$ is expressed as

$$\mathrm{PM}_{DNN}(t+j) = tanh\left(\mathbf{W}^{(o)} \times \boldsymbol{\varphi}^{(l)}\left(\mathbf{W}^{(l)} \times \mathbf{X}^{(l-1)} + \mathbf{b}^{(l)}\right) + b^{(o)}\right),$$
$$\forall l \in \{1,\ldots,L-1\},\ j = \{1,..,7\} \quad (1)$$

where $\mathbf{X}^{(0)}$ is the $N^{(0)}$ sized input vector of the input layer, $\mathbf{X}^{(l-1)}$ is the $N^{(l-1)}$ sized input vector of the $(l\text{-}1)^{\text{th}}$ layer, $\mathbf{W}^{(l)}$ is the $(N^{(l)} \times N^{(l-1)})$ weight coefficient matrix of the l^{th} layer, $\mathbf{b}^{(l)}$ is the $N^{(l-1)}$ sized bias vector of the l^{th} layer, $\mathbf{W}^{(o)}$ is the $(N^{(l)} \times N^{(l-1)})$ weight coefficient matrix of the output layer, $b^{(o)}$ is the bias value of the output layer, and $\boldsymbol{\varphi}^{(l)}$ are the $N^{(l)}$ element-wise activation functions. In Equation. (1), the hyperbolic tangent was chosen for activation function at the output layer, accounting for the nonlinear PM-10 levels, so the training output is within the range [-1, 1]. To avoid large activated outputs from the ELU functions and to maintain stable training, all the input variables are also scaled to this range.

2.3 Dropout DNN-based PM-10 forecasting model

A dense (fully connected) DNN means that almost all activations will be processed to calculate the output of the network. This is costly. Moreover, the time required to complete the training increases with the number of weights, which in turn increases exponentially with the number of hidden layers. Furthermore, using more layers requires more

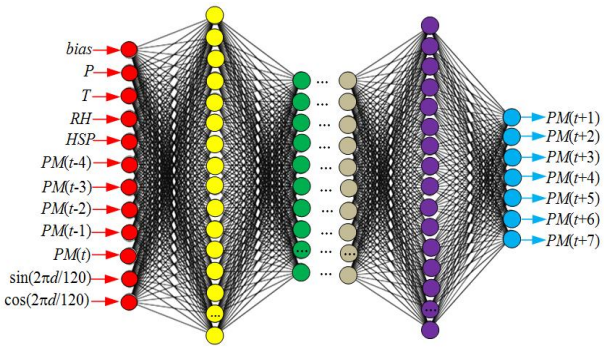

Figure 1. The structure of DNN-based PM-10 forecast model

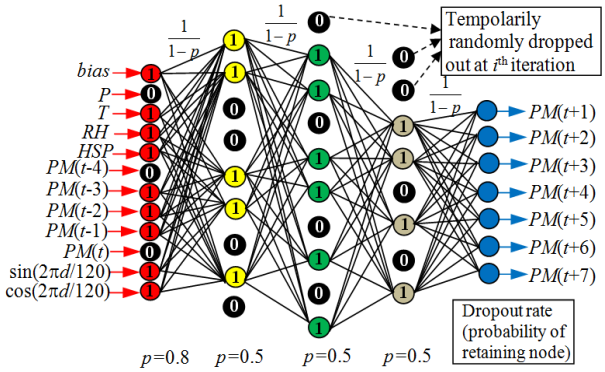

Figure 2. The inverted dropout DNN-based PM-10 forecasting model

training data as well. These are serious issues. Many regularization strategies have been developed to reduce over-fitting. However, the dropout technique among them is less computationally costly, averaging over several models to improve generalization. Regarding this technique, randomly selected nodes called dropouts, are trained rather than the entire network. During the forward and backward phases, the dropout ratio p (black circles in Figure 2) is set to zero or temporarily eliminated. This means that the incoming/ outgoing connections are removed while the rest are trained through BPA as usual, resulting in a thinned DNN. In the next training iteration, other nodes will be selected for training. However, given a total of m nodes in a DNN, 2^m thinned DNNs can be generated, so it is impossible to train them all. For example, there exist 2^{91} sub-network DNNs from a DNN (11, 20, 20, 20, 20, 7) model, where 11 represents the number of input parameters, the 20s are as $N^{(1)}$, $N^{(2)}$, $N^{(3)}$ and $N^{(4)}$; and 7 represents one-week ahead PM-10 forecast. In general, the fraction of hidden neurons dropped out is the dropout rate p=0.5, whereas a smaller dropout rate p=0.8 for the visible or input layer is suggested in (Goodfellow, Bengio, & Courville, 2016). Here, the dropout rate refers to the probability of retaining the neuron, which is different from the definition in Keras, a deep learning library written in Python language. Dropout is not used in the output layer.

There exist two common strategies using the dropout technique. One is inverted dropout at the training phase by multiplying the factor $1/(1–p)$ to the activated neurons for compensating the deactivated neurons, and another is scaling the activation at the test phase by multiplying the factor $(1–p)$ instead to reduce the activation function outputs accounting for the missing neurons during training. The first one has an advantage in doing nothing at the test phase, and this makes inference faster.

With dropout, the output of DNN-based forecasting PM-10 model is expressed as

$$\mathrm{PM}_{Drop\text{-}DNN}(t+j) = tanh\left(\mathbf{W}^{(o)} \times \frac{\mathbf{DROP}^{(l)}}{1-p^{(l)}} \left(\boldsymbol{\varphi}^{(l)} \left(\mathbf{W}^{(l)} \times \mathbf{X}^{(l-1)} \right) + \mathbf{b}^{(l)} \right) + b^{(o)} \right), \quad (2)$$

Here $\mathbf{DROP}^{(l)}$ is a vector of Bernoulli random variables for each dropout mask layer where $DROP_i^{(l)} \sim \mathrm{Bernoulli}(p^{(l)})$, each of which has probability $p^{(l)}$ of being 1.

Given the training set of $\{(\mathbf{X},\ PM(t+j))_i\}$, i =1,2,…,n; the training of Equations. (1) – (2) seeks to optimize the following objective function,

$$\underset{W,b}{\operatorname{argmin}} J(W,b) = \frac{1}{n}\sum_{i=1}^{n} \delta^{n-i} \left\| PM_i - PM_i^{DNN} \right\| \quad (3)$$

where PM_i and PM_i^{DNN} are the measured and forecast PM-10, respectively, and $\delta \in (0, 1]$ is a forgetting factor that helps prevent the BPA from being stuck at a local solution.

To prevent over-fitting, one can add an extra term called weight decay or L^2 penalty (Helmbold & Long, 2018) into Equation (3). Besides, training can be stopped when the performance starts to deteriorate. Another effective solution for preventing over-fitting is regularization by simplifying the architecture of the DNN network as much as possible. Also, high-performance hardware supercomputing such as using the graphics processing unit (GPU) instead of the central PU (CPU) can complete the training faster.

In training, batch gradient descent, mini-batch gradient descent, or stochastic gradient descent (SGD) can be used to optimize the DNN Equation (1) and dropout DNN Equation (2) as follows,

$$(\mathbf{W}^{(l)}, \mathbf{b}^{(l)}) \leftarrow (\mathbf{W}^{(l)}, \mathbf{b}^{(l)}) + \eta \frac{\partial J}{\partial (\mathbf{W}^{(l)}, \mathbf{b}^{(l)})}, \quad (4)$$

$$(\mathbf{W}^{(l)}, \mathbf{b}^{(l)}) \leftarrow (\mathbf{W}^{(l)}, \mathbf{b}^{(l)}) + \eta \mathbf{DROP}^{(l+1)} \left(\frac{\partial J}{\partial (\mathbf{W}^{(l)}, \mathbf{b}^{(l)})} \right), \quad (5)$$

respectively, where η is the pre-defined learning rate. It is noted that for Equation (5), a dropout rate that is too high might lead to inaccurate predictions due to the low number of available connections/neurons.

Normally, the dropout is disabled in the testing phase. However, when making forecasts this produces only the mean value. So, in this work, the dropout DNN is enabled in the testing phase also. Besides, the dropout is used for accelerating the training by placing it in some of the first training epochs, starting with a big dropout ratio (equivalent to the crossover in the genetic process of GA) and attenuating to a small dropout ratio (equivalent to the mutation rate at the

end of the evolution of GA for running out the local trap). At $p = 1$, the whole DNN is back again for remaining training.

For further improving the training efficiency, consider dropout at the visible layer with randomly initializing the weights to prevent feature co-dependence. The information about the order of significant predictors may be lost due to dropout even though later layers have been extensively trained, so that forecasting is extremely difficult especially when reaching the peak. In this work, the most significant predictor variable of $PM(t)$ (i.e., that with highest correlation coefficient) and the parameter of *HSP* that reflects the open burning are omitted from drop out. In addition, a large number of connection weights is provided to them instead of a fully random dropout.

After setting the number of input and output neurons according to the dimensions of the input vector and the forecasting period, respectively, the number of hidden neurons in each hidden layer, including other relevant parameters used in training the DNN, is decided by a genetic algorithm.

2.4 DNN optimization using GA

In this work, 4 cases of NN-model, namely SNN, full DNN, dropout DNN and dropout accelerated DNN, have been implemented in forecasting PM-10. The conventional dropout DNN may not be appropriate to the forecast problem, while classification problems can require only approximate or mean values for the threshold function. Here, the dropout is enabled both in the training and testing phases. In the dropout accelerated DNN, the dropout is used only at some of the first epochs. All models are trained using Matlab deep learning libraries running on CPU. Since NN is a bio-inspired model, there is no rigorous way to identify the hyperparameters that crucially impact the training, which optimizes the NN parameters (i.e., weights and biases). Typically, the hyperparameters fall into 2 categories: (1) model-specific hyperparameters that determine the network structure (e.g. the numbers of hidden layers and neurons, activation function type, and dropout ratio), and (2) optimizer hyperparameters that determine how the network is trained (e.g., learning rate, mini-batch size, and the number of epochs). Choosing appropriate hyperparameters plays a key role to success in maximizing model performance. There are many techniques to find good choices of hyperparameters, such as manual search (Lecun *et al.*, 1998), grid search (GS) (Lecun *et al.*, 1998), random search (RS) (Bergstra & Bengio, 2012), and Bayesian optimization (BO) (Snoek, Larochelle, & Adams, 2012). To avoid the exploding time complexity when searching for a large number of parameters, such as in the first three methods, the BO based on the Gaussian process has probably an advantage over them.

In this work, genetic algorithm (GA), based on an evolutionary approach to develop the best fit solution, is applied to find the best set of hyperparameters. Regarding the internal parameters of an SNN, i.e., its weights and biases, these are optimized using BPA as usual; however, the number of hidden nodes has no rigorous basis for selecting it, and is chosen by starting with a single node and increasing the node count until the performance no longer improves. The hyperparameters of the DNN considered in this study are detailed in Table 1. There are 6 model-specific types of hyperparameters (L, $N^{(l)}$, *Epoch*, $p^{(1)}$, $p^{(l)}$, and φ) and 2 optimizer hyperparameters (η and *BZ*) with their specific ranges. The parameters $p^{(1)}$ and $p^{(l)}$ are set to 0 for the fully connected DNN.

In the process of GA, the number of individual chromosomes (representing solutions) of N_{chr} per generation is a vector possessing certain genes (representing hyper parameters) (Figure 3) and is randomly initialized and encoded in N_{bit}-length binary string of the j^{th}-gene. To optimize the hyperparameters for DNN, the parameters L, $N^{(l)}$, η, $p^{(1)}$, and $p^{(l)}$ are normalized within the pre-defined ranges $[Gene_j^{min}, Gene_j^{max}]$ as follows,

$$Gene(i,j) = \frac{\left(Gene_j^{max} - Gene_j^{min}\right)}{2^{N_{bit}}} \times y(i,j) + Gene_j^{min}, \quad (6)$$

where $y(i, j)$ is the real value converted from the binary string of the j^{th}-gene. The other hyperparameters of *BZ*, *Epoch*, and φ having 8 levels each are represented by {000, 001, 011, …, 111}. For instance, if $N_{bit} = 3$, the number of combinations of the hyperparameters set is up to 16,777,216.

The individual chromosomes are assessed using the average MSE of each DNN in forecasting the PM-10 through the BPA under the following specific fitness function,

$$f_{i,n} = -\log MSE_{i,n} + C_{f_{i,n}} \frac{\sum_{k=1}^{N^{(1)}} \left(\partial J / \partial \theta^{(k)}\right)_n}{\sum_{k=1}^{N^{(L)}} \left(\partial J / \partial \theta^{(k)}\right)_n}, \quad (7)$$

where $f_{i,n}$ is the fitness of the i^{th}-chromosome at the n^{th}-generation considering both improving accuracy in term of MSE and reducing VGP in terms of the gradient of J with respect to the parameter $\theta^{(k)} = \{W^{(k)}, b^{(k)}\}$ of the first hidden layer with respect the output layer, and C_f is the weighted constant in the range [0.08, 0.02]. Thus, the hyperparameters

Table 1. The undecided hyperparameters of DNNs

Parameter	Abbreviation	Range
1) Learning rate	η	$[0.001, 0.05] \in \mathrm{R}$
2) Mini-batch size	*BZ*	{12, 24, 36, 48, 60, 72, 84, 120}
3) The number of hidden layers	L	$[3, 34] \in \mathrm{I}$
4) The number of hidden neurons of each l^{th}-hidden layer	$N^{(l)}$	$[3, 34] \in \mathrm{I}$
5) The number of epochs	*Epoch*	{10, 50, 100, 200, 500, 1000, 2000, 5000}
6) Dropout ratio (input layer)	$p^{(1)}$	$[0.5, 0.9] \in \mathrm{R}$
7) Dropout ratio (hidden layer)	$p^{(l)}$	$[0.2, 0.7] \in \mathrm{R}$
8) Activation function	φ	{'ReLU', 'LReLU', 'ELU', 'ReLU+LReLU', 'ReLU+ELU', 'tanh+ReLU', 'tanh+LReLU', 'tanh+ELU'}

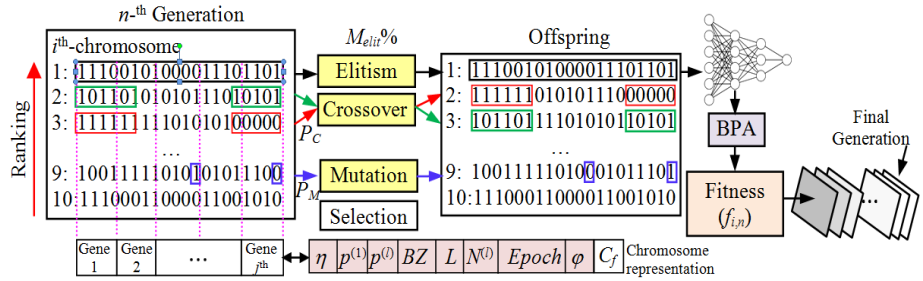

Figure 3. The flowchart of hyperparameter optimization for DNN using GA

are optimized while minimizing the error and preserving the training. From Equation (7), the higher the fitness value, the higher score the chromosome gets.

To maintain some higher chromosomes and unconditionally passing them to the next generation, for each generation the elitism strategy is used to keep the dominant chromosomes in top M_{elit}%. On the other hand, the rest are selected to the next generation using the roulette wheel method, by assigning a higher probability of selection to those with higher fitness values. Between two individuals the preserved and the selected chromosomes are crossed at random positions to exchange information in the crossover stage at probability P_c. In order to escape from local minima, the newly generated chromosomes are mutated at random positions with probability P_m. After genetic operations, the NN are trained by BPA with initial values for 100 times and the average of MSE and gradient are taken for the fitness function. The process of GA is then repeated until there is no further improvement in the convergence rate or in meeting the maximum generation.

After obtaining the forecast models and completing the forecasts, the performances of the NNs are evaluated using the three error metrics MAE, RMSE, and MAPE shown in Equations (8) – (10).

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|PM_i^{Measured} - PM_i^{Forecast}\right|, \quad (8)$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(PM_i^{Measured} - PM_i^{Forecast}\right)^2}, \quad (9)$$

and

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{PM_i^{Measured} - PM_i^{Forecast}}{PM_i^{Measured}}\right| \times 100\%, \quad (10)$$

where the first two metrics asses the model performance, whereas the last metric indicates the accuracy. In addition, the non-parametric Wilcoxon Signed-Rank test is employed to evaluate the hypothesis of the difference between forecast and measured PM-10 concentration levels through the P-value, in order to assess the quality of forecasting models.

3. Results and Discussion

For hyperparameter optimization using GA, the parameters were set as follows: N_{chr}= 10, N_{bit}= 5, M_{elit}= 10%, double point crossover with P_c= 0.9, and multiple point mutation with P_m= 0.15. After meeting the stopping criterion of maximum generation set as 500, the best chromosome (solution) of the DNNs corresponding to the connecting weights and biases found with BPA is shown in Figure 4 using three datasets for CM, CR, and MHS provinces.

Due to each area having a different dataset, the best sets of hyperparameters also differ and were achieved by different generations. At the maximum epoch, the dropout accelerated DNN outperformed the rest. The convergence speed and the optimal fitness obtained for an SNN are very slow and low, respectively, compared to the DNNs. This also verifies the ability of the DNNs to capture dynamic changes better than an SNN. The convergence of the best solution is shifted to the left, indicating shorter time to reach accuracy (less generations in GA process).

It is seen from Table 2 that most of the best solutions tend to employ the combination of hyperbolic tangent and various ReLUs as activation functions, meaning that the VGP was reduced. As regards selected hidden neurons, their number in the DNNs varied in the range 360 – 5,860. During setting the hyperparameters of the DNNs, they are optimized by SGD through BPA in computing the gradient at each iteration. The error performances of the SNN, DNN, dropout DNN, and dropout accelerated DNN (or Dropout acc. DNN for short) using the training data of 2012-2016 for one-day ahead forecast, as an example, are compared in Figure 5. The errors continuously decrease with epochs.

While the smallest errors are obtained from the dropout DNN in the training phase, a lower accuracy is found in the validation phase and possibly in the test phase. The dropout accelerated DNN outperformed the alternatives both in the training phase and in unseen data during the validation phase.

The validation results for the dropout accelerated DNN for one-day to one-week ahead forecasts are given in Figure 6. It performs very well and moderately well for one-day and three-day ahead forecasts, respectively. In contrast, the performance deteriorates for the longer one-week ahead forecasts. Therefore, the dropout accelerated DNN was selected as the PM-10 forecasting model for the study areas. To verify the performance, the forecast PM-10 were assessed with test data.

The performances of the SNN and DNN based PM-10 forecasting models in terms of the MAE and RMSE in the test phase are detailed in Table 3. By averaging the results, the dropout accelerated DNN outperforms the DNN, SNN, and dropout DNN, respectively. All the models provide high

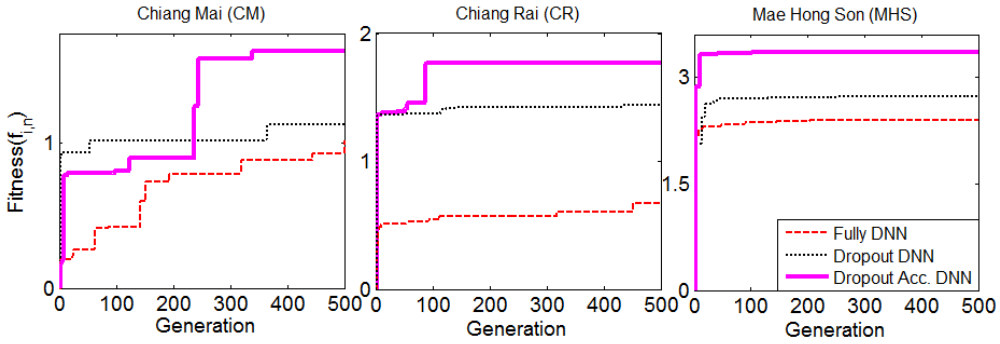

Figure 4. Convergence of the best fit chromosome against the number of generations

Table 2. DNN hyperparameter tuning by GA

Hyper-parameter	CM			CR			MHS		
	Fully DNN	Dropout DNN	Dropout Acc. DNN	Fully DNN	Dropout DNN	Dropout Acc. DNN	Fully DNN	Dropout DNN	Dropout Acc. DNN
η	0.0040	0.0190	0.0115	0.0055	0.0055	0.0025	0.0145	0.0130	0.0040
BZ	60	120	120	60	48	120	48	60	60
L	5	13	7	6	16	10	6	15	8
$N^{(l)}$	8	15	10	5	18	12	10	20	9
$Epoch$	2000	5000	2000	1000	2000	2000	5000	2000	1000
$p^{(1)}$	-	0.85	0.725	-	0.812	0.775	-	0.887	0.712
$p^{(l)}$	-	0.527	0.293	-	0.59	0.403	-	0.574	0.340
φ	tanh+ ReLU	ELU	tanh+ LReLU	tanh+ ReLU	tanh+ LReLU	tanh+ ReLU	tanh+ LReLU	tanh+ ELU	tanh+ ELU

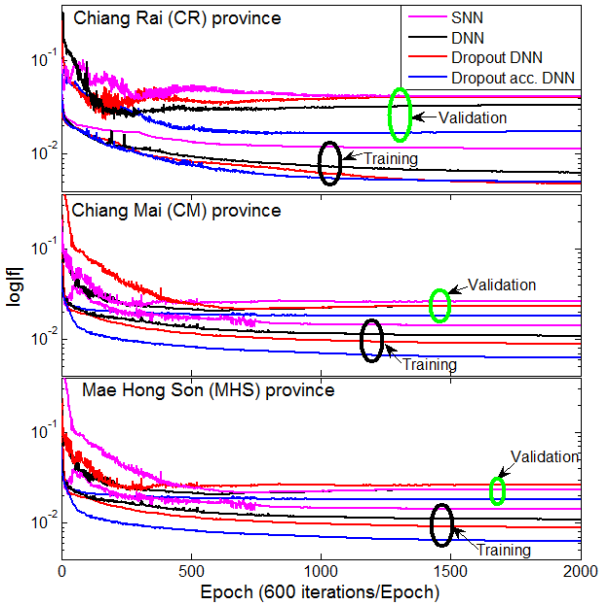

Figure 5. Comparison of error performances between SNN, DNN, dropout DNN and dropout accelerated DNN in one-day ahead forecasts

accuracy for a one-day ahead forecast, whereas their performances are deteriorated for 3-day and one-week ahead forecasts. This may indicate that the instant PM-10 is naturally affected by the current uncertain meteorology variations, unexpected source volumes, and other unknown disturbances. To address the problem, one may use the previous days forecast as a basis to forecast the next day. However, error propagation is a major drawback. Moreover, this aspect requires two or more forecasting models, with redundancy.

In addition, a statistical test at 95% confidence level (α = 0.05) through the non-parametric Wilcoxon Signed-Rank test shows that the proposed model yielded P-values greater than 0.05 among the rest for 1-day ahead forecast for all the three study areas, with a statistically insignificant difference between the forecast and the measured PM-10 concentration levels. However, for 3- and 7-day ahead forecasts, the forecasting models give P-values less than 0.05, showing that these predictions did not fit the data well, which supports the results stated above.

To investigate the forecasting performance of the dropout accelerated DNN the forecast results are shown in Figure 7. It is seen that the low to moderate peaks are captured very well with the proposed DNN but capturing the highest peak is unsatisfactory. To overcome the problem, seeking other significant predictors that cause perturbations, and placing them around the highest peak occurrence would improve the performance. Besides, the errors in forecasts from a good fitted model are usually normally distributed (i.e., zero mean or not biased) and uncorrelated (i.e., no information left in the errors). The degree of distribution is generally expressed by mean absolute deviation (MAD), whereas the correlation of errors is assessed from the sample autocorrelation function (ACF) plot. As a result, for one-day ahead forecast, the proposed DNNs have mean near zero and

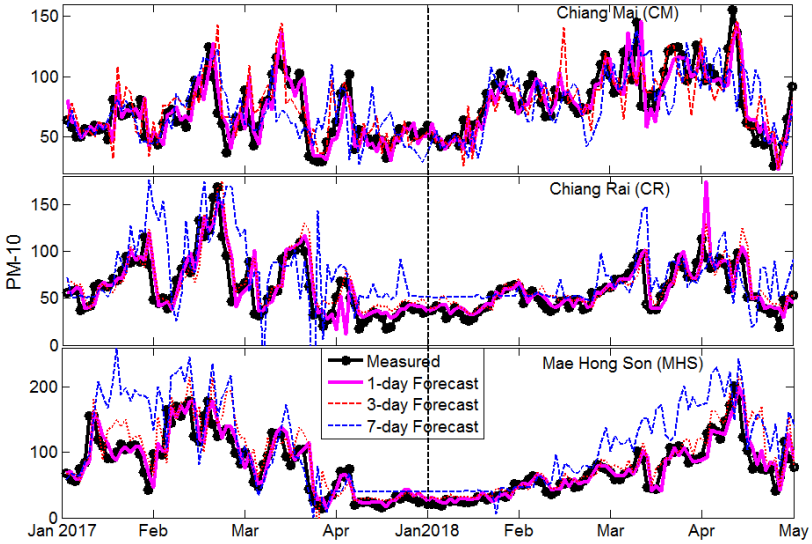

Figure 6. Validation results for PM-10 forecasts by the dropout accelerated DNN

Table 3. Performance in one-day to one-week ahead forecasts for SNN, DNN, dropout DNN, and dropout accelerated DNN, for all study areas

Forecast Period	Model Structure	Area											
		CM				CR				MHS			
		RMSE	MAE	MAPE	P-value	RMSE	MAE	MAPE	P-value	RMSE	MAE	MAPE	P-value
1-day	SNN	20.79	13.88	16.13	0.492	27.43	16.15	16.99	0.479	24.77	15.62	16.44	0.434
	DNN	18.56	13.21	15.89	0.300	26.38	15.56	16.57	0.564	21.23	14.82	16.33	0.443
	Dropout DNN	25.78	16.78	16.59	0.533	29.39	20.32	17.32	0.406	29.33	18.39	17.32	0.402
	Dropout acc.DNN	17.38	11.06	14.59	0.761	24.26	13.87	15.41	0.502	20.09	14.04	15.17	0.623
3-day	SNN	38.33	22.55	24.83	0.024	47.83	29.38	28.89	0.002	36.77	26.36	25.95	0.052
	DNN	40.08	23.00	26.30	0	45.23	27.20	28.13	0.111	34.22	24.93	25.38	0.079
	Dropout DNN	43.22	28.39	27.77	0	50.38	31.23	30.33	0	39.02	29.73	27.02	0
	Dropout acc.DNN	36.64	22.38	25.78	0.035	42.11	26.73	27.01	0.135	35.17	23.95	25.11	0.073
7-day	SNN	34.97	27.32	26.11	0.213	56.98	37.32	38.19	0	49.36	32.78	37.08	0.0019
	DNN	36.73	30.45	32.72	0	62.38	42.35	38.97	0	52.30	40.32	39.33	0.074
	Dropout DNN	35.84	28.32	27.23	0	57.29	40.02	38.77	0	50.08	36.33	37.35	0.075
	Dropout acc.DNN	33.77	25.40	25.93	0.234	55.78	36.92	37.24	0	49.16	34.67	36.12	0.002

low MAD, and no spike at any lag in an ACF plot, indicating normal distribution and no correlation, respectively. In contrast, for 3-day and one-week ahead forecasts, the results are the opposite.

4. Conclusions

The forecast of PM-10 with many correlated predictors is ineffective when using an SNN. This paper proposed state-of-art DNNs to forecast the daily PM-10. The proposed DNN training was optimized with a GA, as well as accelerated and using dropout regularization technique, and this alternative provided better forecasting accuracy among the alternatives tested when applied to three study areas. However, achieving high accuracy only in a very short horizon forecast (i.e., only one-day ahead) is a disadvantage since this is not sufficient to inform the public in advance about health risks. Furthermore, peak forecasting needs to be improved. To overcome the problems, a long short-term memory recurrent NN (LSTM-RNN), which can learn the nonlinear patterns well and store information over a long period using a memory unit within a structure, might be appropriate for PM-10 forecasting model, and could be tested in a future study.

Acknowledgements

The author directly acknowledges the research fund support from the Research Institute of North-Chiang Mai University.

References

Amnuaylojaroen, T., & Kreasuwun, J. (2011). Investigation of fine and coarse particulate matter from burning areas in Chiang Mai, Thailand using the WRF/CALPUFF investigation of fine and coarse particulate matter. *Chiang Mai Journal of Science*, *39*(2), 311-326. Retrieved from https://www.researchgate.net/publication/253242252

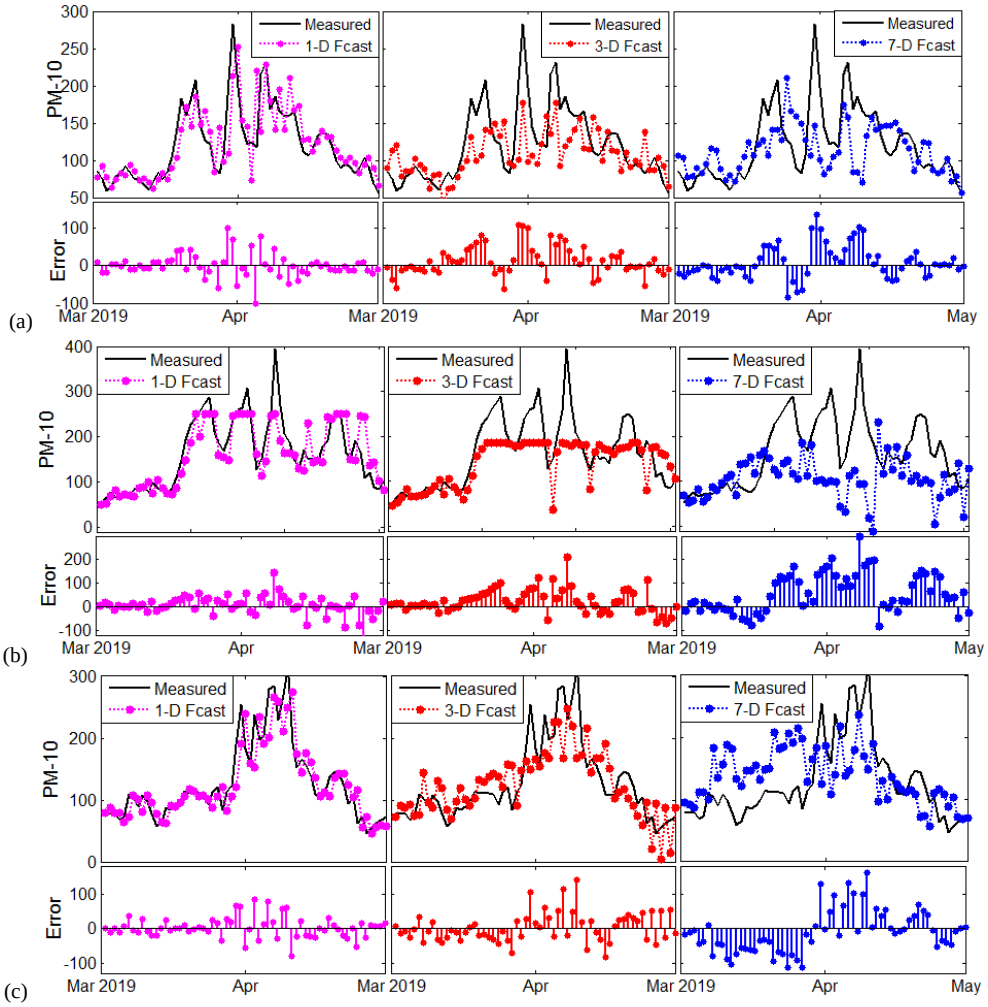

Figure 7. Testing results of PM-10 forecasts by the proposed dropout accelerated DNN for (a) CM, (b) CR, and (c) MHS areas.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, *13*, 281-305. Retrieved from https://pdfs.semanticscholar.org/8e28/30fb6ba9ea201d4ec5d2d800ab9a9bf9677c.pdf

Feng, S., Zhou, H., & Dong, H. (2019). Using deep neural network with small dataset to predict material defects. *Material and Design*, *162*(15), 300-310. doi:10.1016/j.matdes.2018.11.060

Goldberg, D. E. (1989). *Genetic algorithm in search, optimization and machine learning*. Addison-Wesley Longman Publishing Company Publisher, Boston, MA, USA.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press., November.

Helmbold, D. P., & Long, P. M. (2018). Surprising properties of dropout in deep networks. *Journal of Machine Learning*, *18*, 1-28. Retrieved from https://www.researchgate.net/publication/326109066

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780. doi:10.1162/neco.1997.9.8.1735

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the Institute of Electrical and Electronics Engineers*, *86*(11), 2278-2324. doi:10.1109/5.726791

Macatangay, R., Gagtasa, G., & Sonkaew, T. (2017). Non-chemistry coupled PM10 modeling in Chiang Mai city Northern Thailand: A fast operational approach for aerosol forecasts. *Journal of Physics: Conference*, *901*, 1-6. doi:10.1088/1742-6596/901/1/012037

Merkel, G. D., Povinelli, R. J., & Brown, R. H. (2018). Short-term load forecasting of natural gas with deep neural network regression. *Energies*, *11*, 1-12. doi:10.3390/en11082008

Pasukphun, N. (2018). Environmental health burden of open burning in northern Thailand: A Review. *Pibulsongkram Rajabhat University Journal of Science and Technology*, *3*(3), 11-28. Retrieved from https://www.tci-thaijo.org/index.php/Scipsru/article/view/143514

Phyo, P. P., & Jeenanunta, C. (2019). Electricity load forecasting using a deep neural network. *Engineering and Applied Science Research (EASR)*, *46*(1), 10-17. Retrieved from https://www.tci-thaijo.org/index.php/easr/article/view/116025

Pimpunchat, B., Sirimangkhala, K., & Junyapoon S. (2014). Modeling haze problems in the North of Thailand using logistic regression, *Journal of Mathematical and Fundamental Sciences*, *46*(2), 183-193. doi:10.5614/j.math.fund.sci.2014.46.2.7

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. *Proceedings of the 25th International Conference on Neural Information Processing Systems*, *2*, 2951-2959. Retrieved from https://papers.nips.cc/paper/4522-practical-bayesian-optimization-of-machine-learning-algorithms.pdf

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Sukakhutdinov, R. (2014) Dropout: A simple way to prevent neural network from overfitting. *Journal of Machine Learning Research*, *15*, 1929-1958. Retrieved from https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf

Wongsathan, R. (2018). The hybrid neural networks-ARIMA/X models and ANFIS model for PM-10 forecasting: A case study of Chiang Mai, Thailand's high season. *Engineering Journal Chiang Mai University*, *25*(1), 203-213. Retrieved from http://researchs.eng.cmu.ac.th/UserFiles/File/Journal/25_1/17Rati.pdf

Wongsathan, R. (2018). Improvement of PM-10 forecast using ANFIS model with an integrated hotspots. *Science and Technology Asia*, *23*(3), 62-71. doi: 10.14456/scitechasia.2018.25