An Effective DNN Hyperparameter Optimization Method for Network-Wide Traffic Forecasting

Chun-Sheng He
Department of Computer Science and Engineering, National Sun Yat-sen University
Kaohsiung 80424
Taiwan, R.O.C
m093040078@g-mail.nsysu.edu.tw

Chun-Wei Tsai
Department of Computer Science and Engineering, National Sun Yat-sen University
Kaohsiung 80424
Taiwan, R.O.C
cwtsai@cse.nsysu.edu.tw

Ming-Chao Chiang
Department of Computer Science and Engineering, National Sun Yat-sen University
Kaohsiung 80424
Taiwan, R.O.C
mcchiang@cse.nsysu.edu.tw

ABSTRACT

Several recent studies have shown the possibilities of deep learning technologies for traffic forecasting problems. Most researchers in this domain notice that hyperparameter settings have a strong impact on the forecasting accuracy of a deep neural network (DNN) model. To address this issue, an effective metaheuristic algorithm based on grey wolf optimizer (GWO) is presented in this study for finding out a set of suitable hyperparameters for DNN because the design of GWO takes into account the trade-off between exploitation and exploration during the convergence process. To evaluate the performance of the proposed algorithm, it is used to find out a set of suitable hyperparameters of DNN for solving the so-called network-wide time series traffic forecasting problem. Experimental results show that the proposed method can obtain better hyperparameter configurations than other hyperparameter optimization algorithms compared in this study.

KEYWORDS

Grey wolf optimizer, Hyperparameter optimization, Deep neural network.

1 INTRODUCTION

Under limited road resources, the continuously rising number of vehicles may cause a large number of traffic congestions. To mitigate these problems, most countries have paid close attention to the development of intelligent transportation system (ITS) technologies [1] in the hope that they may use it to improve traffic conditions. In recent years, different deep neural network (DNN) methods [2–7] have been presented for solving various prediction problems in recent years. For the DNN model training process, the accuracy of the prediction model will be greatly affected by the external settings [8], such as weight initialization, hyperparameter selection, network architecture, and so forth. That is why we have to select and fine-tune the hyperparameters of DNN for training the model. These hyperparameters can be the type of optimizer, learning rate, batch size, or even the type of activation function. For the researcher who is not familiar with deep learning or the domain, finding out a set of appropriate hyperparameters is a difficult task. Even for an expert, she or he will need to give it a try to get a good set of hyperparameters.

Most of the researchers in the machine learning domain know that the trial-and-error method is not a cost-effective method because DNN models take time to train. How to use a search algorithm to automatically find out a set of suitable hyperparameters has become a promising research topic for years [9–15]. Among the search algorithms, since metaheuristic algorithms provide an effective way to balance the searches of exploration and exploitation, they have been used in several recent studies to find out the suitable hyperparameters for DNNs. Examples are genetic algorithm (GA) [13–15], and particle swarm optimization (PSO) [16–18]. Different to traditional search methods (e.g., grid search or random search), metaheuristic algorithms are able to achieve better performance by maintaining a higher search diversity and using powerful search strategies. In this study, the grey wolf optimizer (GWO) [19] is used for tuning the hyperparameters of the DNN model equipped with long short-term memory (LSTM) and attention layer for the traffic forecasting problem.

The remainder of this paper is structured as follows. Section II gives the definitions of hyperparameter optimization problems and the possible ways for solving this problem. Section III shows the architecture of DNN model with the proposed hyperparameter optimization algorithm. Section IV presents the experimental results and gives a detailed analysis. Finally, the concludes are drawn in Section V.

2 RELATED WORK

2.1 Problem Definition of Hyperparameter Optimization

As we mentioned before, hyperparameters will greatly affect the training performance of DNN models. A slight difference of hyperparameters might degrade the accuracy of a prediction model. To find a high-quality set of hyperparameters with high efficiency, search algorithms were proposed for solving the hyperparameter optimization problem (HPO). Now, let $\mathcal{H}$ denote all the possible combinations of hyperparameters of this optimization problem and $\vec{h} = (h_1, h_2, h_3, \dots h_d)$ denote a possible combination sampled from $\mathcal{H}$. The goal of HPO is to minimize the loss function $\mathcal{L}$ by generating candidate solutions iteratively to obtain the optimal hyperparameter configuration $\vec{h}^*$ so as to boost the accuracy of the model on the traffic forecasting. The traffic data are time series data that are constantly detected by the sensors on the road, e.g., vehicle detector (VD) or camera.

The dataset $\mathcal{D}$ will be divided into three subsets; namely, the

training subset $\mathcal{D}_t$, the validation subset $\mathcal{D}_v$, and the testing subset $\mathcal{D}_T$ where $\mathcal{D}_t \cap \mathcal{D}_v \cap \mathcal{D}_T = \emptyset$. The HPO problem can now be defined as follows:

$$\vec{h}^* = \arg\min_{\vec{h}\in\mathcal{H}} \mathcal{L}(\mathcal{D}_T, \mathcal{M}(\mathcal{D}_t, \mathcal{D}_v, \mathcal{A}, \vec{h})), \quad (1)$$

where $\mathcal{L}$ is the loss function to evaluate the error between the ground truth Y_t and the prediction result by using model $\mathcal{M}$ that is trained by the machine learning algorithm $\mathcal{A}$ using the training dataset $\mathcal{D}_t$, the validation dataset $\mathcal{D}_v$, and the hyperparameter configuration $\vec{h}$.

2.2 The Hyperparameter Optimization Algorithms

Most of the hyperparameter tuning methods in the early stage research use more intuitive ways, such as manual search [20], [21] and grid search [22] to find out the suitable hyperparameters. When training a DNN model, the cost for evaluating the performance of a set of hyperparameters is so high that makes it infeasible to use exhaustive approaches. Thus, researchers have turned to finding out an approximate solution in a limited time. In addition to grid search, Bergstra et al. [9] adopted random search for HPO problems to achieve a better result efficiently by avoiding searching for the same solutions. Sequential model-based optimization (SMBO) [12] is a framework of Bayesian Optimization (BO) for solving HPO problems. It leverages the past input and evaluation value to build a model for determining the next solution. By updating the model iteratively, the model will become stable and have fewer uncertainty regions. Spearmint [10] is a Gaussian process (GP) based algorithm for finetuning the hyperparameters. It works on both continuous and discrete solution spaces but is hard to be useful on conditional hyperparameter space or high-dimensional solution space. The tree-structured Parzen estimator (TPE) was first proposed in [11], which improve the ability of solve HPO problems on non- continuous search space by constructing the two different distributions.

Since metaheuristic algorithms are capable of finding an approximate solution in solving optimization problems within a reasonable time, some recent studies [13–15], [23] have attempted to apply it to HPO. For instance, Loshchilov et al. [23] used covariance matrix adaptation evolution strategy (CMA-ES) to search for the hyperparameters for a DNN model trained by using computer vision datasets. Of course, the genetic algorithm (GA) [13–15] was also used for solving HPO. In addition, some other swarm intelligence methods have also been used in solving HPO in recent years. In [16], [17], Lorenzo et al. used particle swarm optimization (PSO) for tuning hyperparameters of CNN models. The methods BOHB [24] and DEHB [25] have attempted to combine the Bayesian optimization and differential evolution (DE) with hyperband to further improve the performance and efficiency of tuning hyperparameters.

3 PROPOSED METHOD

3.1 The Basic Idea of GWO

The grey wolf optimizer [19] is a novel metaheuristic algorithm inspired by the social hierarchy and cooperative prediction behavior of grey wolves. It aims to establish a mathematical model by imitating the group hunting process and social hierarchy of grey wolves. GWO regards the lowest level wolves, omega, as the search agents to approach the prey (global optimum). Search agents relocate their positions according to the distance from the global optimum step by step, as defined by

$$\vec{X}(t+1) = \vec{X}(t) - \vec{A} \cdot \vec{D}, \quad (2)$$

where $X(t)$ and $X(t+1)$ represent, respectively, the positions of wolves at iterations t and $t+1$. The operator $\cdot$ denotes element-wise multiplication of two vectors, and D is the distance vector between the optimal position and the current position and is affected by the coefficient vector A. It is computed as follows:

$$\vec{D} = \left|\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)\right|, \quad (3)$$

where $\vec{X}_p$ represents the current position of the target and C is a coefficient vector. The aforementioned A and C are generated with randomness to increase exploratory and are computed as follows:

$$\vec{A} = 2\vec{a} \cdot \vec{r_1} - \vec{a}, \quad (4)$$

$$\vec{C} = 2 \cdot \vec{r_2}, \quad (5)$$

where $\vec{r_1}$ and $\vec{r_2}$ are both vectors randomly sampled from $[0, 1]$, the values in $\vec{a}$ are linearly decayed from 2 to 0 over iterations. In practice, the global optimum is an unknown value. To substitute the unknown position of global optimum, the top three solutions so far should be distinguished and recorded for guiding the searchers. Each search agent will generate three auxiliary solutions at the neighborhood according to the guidances. The searchers will get deeper into the better region and the new guidance solutions can be determined for further exploitation. At later iterations of the proposed algorithm, the position of the swarm can be quite close to the optimum. To converge the solutions, the values in the coefficient vector $\vec{A}$ displayed in Eq. (4) become small so that the auxiliary solutions can locate on the position that is not far off the guidances.

3.2 GWO for Hyperparameter Optimization of DNN

In this study, a DNN model is built for the time series vehicle speed prediction. To improve the accuracy of such a model, we proposed a method called grey wolf optimizer for hyperparameter optimization (GWOHPO), based on the grey wolf optimizer (GWO) [19], and used it to search the suitable hyperparameters so that the DNN model trained with this configuration can obtain a better result. The proposed method for solving the HPO problem is as outlined in Algorithm 1. The procedure can be roughly divided into four phases, namely, initialization, transition, evaluation, and determination. The details of each phase will be described in turn below.

Initialization: At the beginning, several sets of hyperparameter combinations are randomly generated within the boundary of a uniform distribution according to the self-defined population size

n. Some hyperparameters, such as the activation functions and batch size, will be distributed in a discrete solution space. These real numbers have to be converted to a discrete space through the rounding-off procedure. After that, all the initial solutions (hyperparameter configurations) will be evaluated so that the top three best guidance solutions $\vec{X}_{\text{gd}} = \{\vec{X}_\alpha, \vec{X}_\beta, \vec{X}_\delta\}$ can be determined. The best model so far $\mathcal{M}^*$ that represents a model that has the lowest error $\mathcal{E}^*$ trained by the hyperparameter configuration $\vec{X}^*$ is updated.

Algorithm 1: The grey wolf optimizer for hyperparameter optimization

Input: $n, \mathcal{D}$
Output: $\vec{X}^*, \mathcal{M}^*, \mathcal{E}^*$
$\vec{X} = \text{Initialization}(n)$
Evaluation$(\vec{X})$
Update $\vec{X}_{\text{gd}}, \vec{X}^*, \mathcal{M}^*, \mathcal{E}^*$
While: $\vec{X}^*, \mathcal{M}^*, \mathcal{E}^*$
    Update $a, \vec{A}, \vec{C}$
    Transition$(\vec{X}, \vec{X}_{\text{gd}})$
    Evaluation$(\vec{X}, \mathcal{D})$
    Determination$(\vec{X}_{\text{gd}}, \vec{X}^*, \mathcal{M}^*, \mathcal{E}^*)$
End While

Transition: In this stage, each solution is updated using GWO. The three best so far solutions $\vec{X}_{\text{gd}}$ are referred to relocate the omega solution to a better position. In order to tap into the potential of the search space between the best solutions and the current one, the formula mentioned before are modified, as follows:

$$\vec{D}_{\text{gd}} = |\vec{C}_j \cdot \vec{X}_{\text{gd}} - \vec{X}|, \tag{6}$$

$$\vec{X}_j = \vec{X}_{\text{gd}} - \vec{A}_j \cdot \vec{D}_{\text{gd}}, \tag{7}$$

where $\vec{A}_j$ and $\vec{C}_j$ are the coefficient vectors calculated using Eq. (4) and Eq. (5), and $\vec{X}_j$ for $j = 1, 2, 3$ will be the relocated position calculated based on the distance $\vec{D}_{\text{gd}}$ between the current position and the guidance positions α, β, and δ, respectively. Given these three positions, the updated location $\vec{X}(t+1)$ is defined as follows:

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3}. \tag{8}$$

For each search agent, the fetched coefficients need to be updated. The coefficient vector $\vec{A}$ in this method is the main component to directly impact the performance. The absolute value of each element in $\vec{A}$ is linearly decreased from 2 to 0 as the number of iterations increases. It decides how close the solution may migrate to the optimum, which means that the population of GWOHPO will explore less as the number of iterations increases and converge finally. New solutions must be checked again whether it exceeds the boundary. Certain hyperparameters should be rounded off and eventually be delivered to the neural network for evaluation. Fig. 1 illustrates the transition process of GWOHPO. The current solution $\vec{X}_\omega(t)$ obtains three auxiliary positions according to the guidance $\vec{X}_\alpha$, $\vec{X}_\beta$, and $\vec{X}_\delta$ for increasing stochasticity. Then, the next position $\vec{X}_\omega(t+1)$ can be determined by these auxiliary positions to migrate the $\vec{X}_\omega(t)$ to the neighborhood of the guidance solutions.

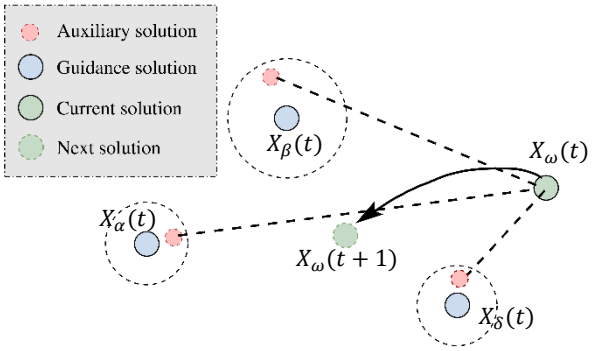

Figure 1: Update strategy in GWOHPO.

Evaluation: In this study, a prediction model constructed by the attention-based LSTM architecture will be used to predict traffic time series data. The main components, LSTM and self-attention mechanism, are used to capture the temporal correlation between traffic states and filter the noise. Dropout layers provide regularization to prevent the network from overfitting. Finally, the features are mapped to the output space by fully connected layer. To speed up the training process, the dataset $\mathcal{D}$ is divided into batches. A hyperparameter configuration consists of learning rate, batch size, dropout rate, the number of units in LSTM and self-attention layer, and the activation functions. To avoid overfitting, a set of data are prepared for validation in each batch.

Determination: With the testing result of each solution, all the solutions can be ranked by their qualities. Thus, the top three best solutions can be updated after evaluating the hyperparameter configurations. According to the new guidance solutions, the population of the GWOHPO can relocate to a better area. If a better hyperparameter is found, the best solution so far $\vec{X}^*$ and the best model $\mathcal{M}^*$ with the loss value $\mathcal{E}^*$ should be replaced.

3.3 The Flowchart of GWOHPO

The flow of the proposed method is as shown in Fig. 2 that can be divided into two parts. The left half is the process of GWOHPO updating hyperparameters, and the right half is the process of black-box evaluation (that is, the result of establishing and training the model based on the hyperparameters). The connection of the two parts is the input of hyperparameters and the output of model evaluation results. Treating the training process of DNN model as a black-box, GWOHPO is used for optimizing the input hyperparameters. Moreover, GWOHPO will record better solutions to attract the search agents for boosting convergence and also provide a decayed coefficient to balance exploration and

exploitation. Search agents will be continuously updated to improve the hyperparameters until the termination condition is reached, and the hyperparameter combination that makes the model have the best evaluation value is retained. In short, given a certain hyperparameter setting for a model, training and testing are regarded as a black-box operation. The external algorithm, GWO, can use the output of the black-box (verification of DNN model) as the objective value of solutions and efficiently optimize the hyperparameter sets with its powerful searching mechanism. Finally, we can obtain the best hyperparameter setting to train a DNN model that can be effectively applied on time series forecasting problems.

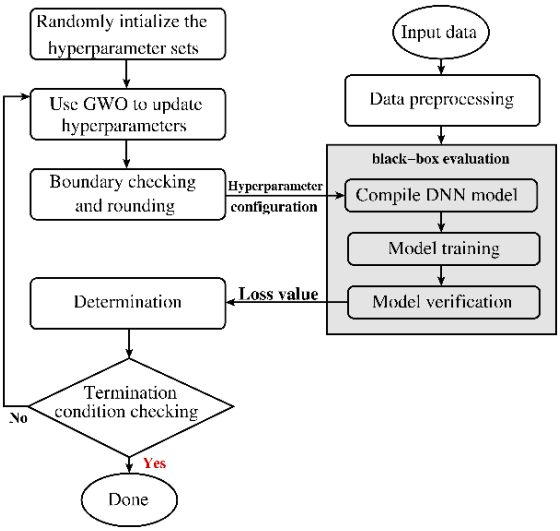

Figure 2: The flowchart of proposed method.

4 EXPERIMENT AND RESULT

4.1 Environment and Parameter Settings

To solve time series problem, we build a DNN model equipped with LSTM and self-attention mechanism. To standardize the performance of the models, we use the opensource framework, Tensorflow, for machine learning. All of the empirical analysis is conducted on the Intel Core i7- 11700K 3.6 GHz CPU with 64 GB of memory and NVIDIA GeForce RTX3070 Ti GAMING OC with 8 GB memory running on Ubuntu 20.04.1. All programs are written in Python 3.8.10 with machine learning library Tensorflow-GPU 2.5.1 and Keras 2.6.0. To measure the applicability of GWOHPO to this problem, five search algorithms will be compared with the GWOHPO. They are random search (RS) [9], Gaussian process (GP) [10], tree-structured Parzen estimator (TPE) [26], genetic algorithm (GA) [15], and particle swarm optimization (PSO) [16]. For fair comparison of GWO and other algorithms, the number of evaluations is set equal to 500 as the termination condition of the HPO search algorithms. All the algorithms are carried out for 5 runs.

The selection of hyperparameters and boundary settings is shown in Table 1. because certain hyperparameters, such as batch size, number of neurons, and activation function, should be input as integers, these values are rounded to fit the format of the solution encoding.

Table 1: Boundary Setting of Hyperparameters

Hyperparameter	Magnitude
Batch size	Uniform (20, 500)
Learning rate	Uniform (0, 0.1)
# of LSTM neurons	Uniform (20, 700)
Dropout rate	Uniform (0, 0.5)
# of Attention neurons	Uniform (20, 700)
# of Dense neurons	Uniform (200, 3000)
Activation function	[ReLU, ELU, softmax, tanh, sigmoid, SELU, GELU]

The model training process will be shut down at the 50-th epoch or once the loss value increasing for 5 consecutive epochs to prevent the model from overfitting and shorten the experiment time. We keep the model with the lowest validation loss for evaluation of the HPO optimization. The widely used optimizer for updating the weights in the neural network, adaptive movement estimation (Adam) [27], is adopted for the model with the gradient calculated by backpropagation method. And, the mean square error (MSE) is served as the loss function with the mean average error (MAE) and the mean average percentage error (MAPE) as supplement to analyze the performance of the model.

4.2 The Traffic Dataset

The data employed in the experiment are extracted from the traffic database of the Freeway Bureau, Ministry of Transportation and Communications (MOTC), Taiwan [28]. These data are collected by the vehicle detectors on the highway system every 5 minutes, include average vehicle speed, traffic volume, vehicle type, and so on. In this study, the forecasting target of DNN model is the average vehicle speed at the next timestamp. Loss value of the model is to evaluate the performance of hyperparameters on optimization problem. To generate the large-scale traffic prediction problem, traffic status data detected from 300 sensor nodes which record the section from Keelung to Zhongli is adopted. The traffic data are captured for the entire year of 2019. The data of the first 11 months are used to train the DNN model, 20% of which are selected for validation, and the entire December data are used to test the performance of the model.

4.3 Results

Since the population size may affect GWOHPO performance for finding the hyperparameters of the DNN model, we conducted parameter experiments on the swarm size first, and the swarm size is starting from 5 and increasing to 20 at intervals of 5. Due to the constant termination condition, the number of iterations will decrease when the swarm size increases. Moreover, if we set swarm size equal to 20, GWOHPO cannot obtain an effective configuration due to insufficient number of iterations. On the other hand, since the elements in coefficient vector $\vec{A}$ will gradually converge to 0 in the later stage, the algorithm will start to exploit

the solution space even though the exploratory search is insufficient that the swarm cannot get enough information to relocate to a better position. From the results of the Table 2, when we set the number of evaluations equal to 500, there is a trend that the smaller the swarm size, the better the result. For each swarm size, we conducted 5 trials and averaged the results of each run. As shown in, GWOHPO can obtain the best results when the swarm size is 5.

Table 2: Result of GWOHPO with Different Swarm Size on Average

Swarm size	MAE	MSE	MAPE
5	1.710	14.713	2.899
10	1.768	15.032	3.019
15	1.727	15.004	2.958
20	3.316	45.138	5.757

The convergence curves for all the algorithms compared in this study are as shown in Fig. 3. At the very beginning of the convergence process, GA first converges to a slightly better solution, but the premature searchers and incompatibility for continuous encoding of GA would make it get the worst performance among all the algorithms. Compared to RS, GP obtains the posterior probability by fewer solutions to estimate the next possible solution. Although GP can efficiently find out an acceptable hyperparameter configuration set, it also may be stuck in a local optimum and cause unstable performance in each run. Moreover, PSO and TPE are not only efficient but also more stable to find better solutions. The former updates the position by referring to the historical best solution and can constantly discover a better position on every iteration, the latter divides the samples into two groups and requires more attempts to build the distribution in the midterm of the convergence process.

Table 3 shows that the final average results for 5 runs on MAE, MSE, and MAPE are 1.710, 14.713, 2.899, respectively. This indicates that GWOHPO outperforms all the other algorithms compared in this study for finding out the hyperparameters. As shown in Fig. 4, GWOHPO can not only make the model reach a lower loss value but also have more stable performance on final result. In summary, compared to other methods, GWOHPO reports similar hyperparameter values and keeps high-quality and stable performance.

5 CONCLUSIONS

In this research, GWOHPO is presented for optimizing the hyperparameters to find out the most suitable architecture and configuration of a neural network that minimize the loss value. LSTM and self-attention mechanism are utilized in the DNN model to solve the time series problem and predict the future traffic status. The data are collected from the highway system in Taiwan and consists of vehicle speed of 300 vehicle detectors with 5-minute time interval to simulate the largescale dataset for evaluating the performance of a DNN model. In the experiment, GWOHPO is first tested on the swarm size. It shows that GWOHPO can get the best performance when the swarm size is set equal to 5 if the number of evaluations is set equal to 500. The experimental results show that GWOHPO can obtain better hyperparameter configurations for the DNN model in traffic forecasting in terms of MAE, MSE, and MAPE. In the future, we will endeavor to improve GWOHPO with a better mechanism to balance the exploration and exploitation and automatically adapt to a parameter setting befitting GWOHPO.

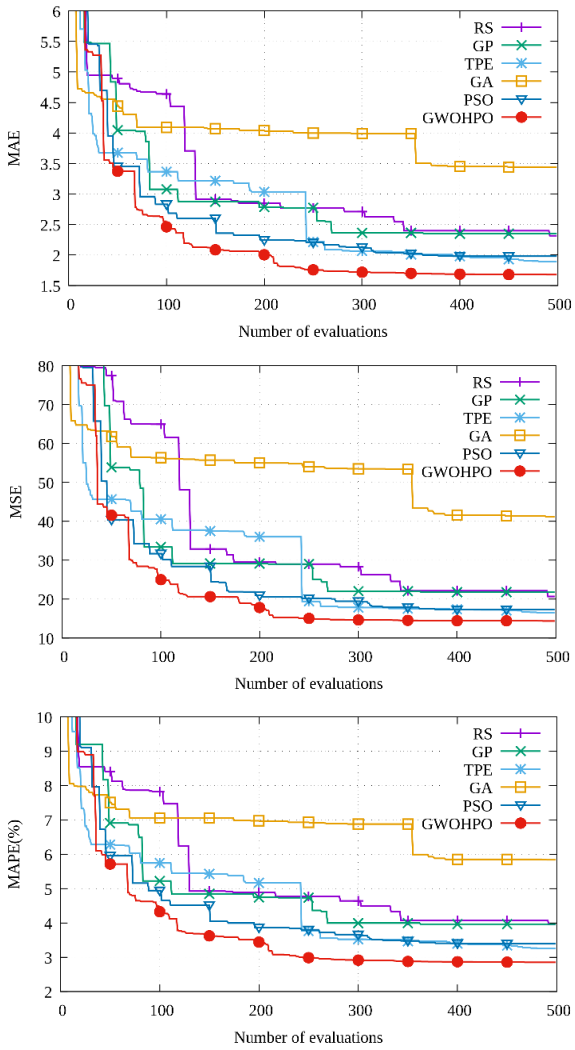

Figure 3: Convergence of MSE, MAE, and MAPE on average.

Table 3: Comparison of Hyperparameter Optimization Algorithm on Average

Algorithm	MAE	MSE	MAPE
RS	2.317	20.700	3.951
GP	2.559	25.442	4.428
TPE	1.894	16.591	3.251
GA	3.438	41.125	5.836
PSO	1.980	17.318	3.390
GWOHPO	1.710	14.713	2.899

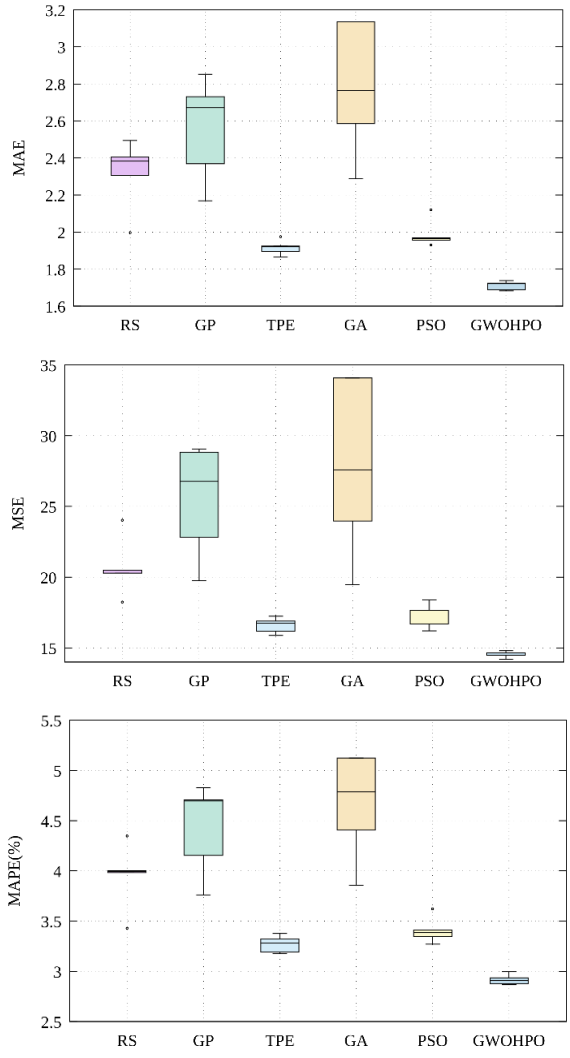

Figure 4: Variation of the output errors on average.

ACKNOWLEDGMENTS

This work was supported in part by the National Science and Technology Council, R.O.C. (Ministry of Science and Technology of Taiwan, R.O.C.), under Contracts MOST108-2221-E-005-021-MY3, MOST110-2221-E-110-005, and NSTC111-2222-E-110-006-MY3.

REFERENCES

[1] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, and C. Chen, "Data-driven intelligent transportation systems: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1624–1639, 2011.

[2] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, "Learningtraffic as images: A deep convolutional neural network for large-scale transportation network speed prediction," *Sensors*, vol. 17, no. 4, p. 818, 2017.

[3] J. Ke, H. Zheng, H. Yang, and X. M. Chen, "Short-term forecasting of passenger demand under on-demand ride services: A spatio-temporal deep learning approach," *Transportation Research Part C: Emerging Technologies*, vol. 85, pp. 591–608, 2017.

[4] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, 2017, pp. 1655– 1661.

[5] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2018, pp. 3634–3640.

[6] Z. Zhao, W. Chen, X. Wu, P. C. Chen, and J. Liu, "LSTM network: A deep learning approach for short-term traffic forecast," *IET Intelligent Transport Systems*, vol. 11, no. 2, pp. 68–75, 2017.

[7] R. Yu, Y. Li, C. Shahabi, U. Demiryurek, and Y. Liu, "Deep learning: A generic approach for extreme condition traffic forecasting," in *Proceedings of the SIAM International Conference on Data Mining*, 2017, pp. 777–785.

[8] F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated machine learning: Methods, systems, challenges*. Springer Nature, 2019.

[9] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. 2, pp. 281–305, 2012.

[10] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1–9, 2012.

[11] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," *Advances in Neural Information Processing Systems*, vol. 24, pp. 1–9, 2011.

[12] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *Proceedings of the International Conference on Learning and Intelligent Optimization*, 2011, pp. 507–523.

[13] X. Xiao, M. Yan, S. Basodi, C. Ji, and Y. Pan, "Efficient hyperparameter optimization in deep learning using a variable length genetic algorithm," *arXiv preprint arXiv:2006.12703*, pp. 1–16, 2020.

[14] Y. Yuan, W. Wang, G. M. Coghill, and W. Pang, "A novel genetic algorithm with hierarchical evaluation strategy for hyperparameter optimization of graph neural networks," *arXiv preprint arXiv:2101.0 9300,* pp. 1–10, 2021.

[15] N. Gorgolis, I. Hatzilygeroudis, Z. Istenes, and L.-G. Gyenne, "Hyperparameter optimization of LSTM network models through genetic algorithm," in *Proceedings of the International Conference on Information, Intelligence, Systems and Applications*, 2019, pp. 1–4.

[16] P. R. Lorenzo, J. Nalepa, M. Kawulok, L. S. Ramos, and J. R. Pastor, "Particle swarm optimization for hyper-parameter selection in deep neural networks," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2017, pp. 481–488.

[17] P. R. Lorenzo, J. Nalepa, L. S. Ramos, and J. R. Pastor, "Hyper- parameter selection in deep neural networks using parallel particle swarm optimization," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2017, pp. 1864–1871.

[18] Y. Wang, H. Zhang, and G. Zhang, "cPSO-CNN: An efficient PSO-based algorithm for fine-tuning hyper-parameters of convolutional neural networks," *Swarm and Evolutionary Computation*, vol. 49, pp. 114–123, 2019.

[19] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014.

[20] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R Müller, "Efficient backprop," in *Neural Networks: Tricks of the Trade*, 2012, pp. 9–48

[21] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527– 1554, 2006.

[22] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio,"An empirical evaluation of deep architectures on problems with manyfactors of variation," in *Proceedings of the International Conference on Machine Learning*, 2007, pp. 473–480.

[23] I. Loshchilov and F. Hutter, "CMA-ES for hyperparameter optimization of deep neural networks," *arXiv preprint arXiv:1604.07269*, pp. 1–8, 2016.

[24] S. Falkner, A. Klein, and F. Hutter, "BOHB: Robust and efficient hyperparameter optimization at scale," in *Proceedings of the Inter- national Conference on Machine Learning*, 2018, pp. 1437–1446.

[25] N. Awad, N. Mallik, and F. Hutter, "DEHB: Evolutionary Hyperband for scalable, robust and efficient hyperparameter optimization," in*Proceedings of the International Joint Conference on Artificial Intelligence*, 2021, pp. 2147–2153.

[26] J. Bergstra, B. Komer, C. Eliasmith, D. Yamins, and D. D. Cox,"Hyperopt: A python library for model selection and hyperparameter optimization," *Computational Science & Discovery*, vol. 8, no. 1, p. 014008, 2015.

[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, pp. 1–15, 2014.

[28] Freeway Bureau, MOTC, Taiwan, "5-minute dynamic vehicle detector information," 2019. [Online]. Available: https://tisvcloud.freeway.gov.tw/