ARDE-N-BEATS: An Evolutionary Deep Learning Framework for Urban Traffic Flow Prediction

Xiaocai Zhang, Zhixun Zhao, and Jinyan Li

***Abstract*—Accurate and reliable traffic flow prediction is difficult due to the highly nonlinear, complex, and stochastic natures of urban traffic flow data, but its solutions are critically important for intelligent transportation systems (ITSs) and Internet of Things (IoT). In this study, a novel deep learning framework, named adaptive reinitialized differential evolution (ARDE)-neural basis expansion analysis for time-series forecasting (N-BEATS), is proposed to address this challenge. With the framework of ARDE-N-BEATS, first, an N-BEATS-based deep learning architecture is formulated for modeling traffic flow data. Second, a novel enhanced evolutionary algorithm, termed ARDE, is presented for optimizing the hyperparameter and structure of N-BEATS. Compared to the vanilla differential evolution (DE) algorithm, ARDE exhibits faster convergence and stronger searching capabilities. Experiments on three real-world traffic flow data sets from Dublin and San Francisco demonstrate that ARDE-N-BEATS can achieve high accuracy of at least 94% for most of the predictions, and outperforms the existing counterpart methods. A comparison between different hyperparameter optimization approaches further reveals that ARDE provides better or very competitive predictions and saves as high as 78.90% of computational expense.**

***Index Terms*—Differential evolution (DE), evolutionary deep learning, intelligent transportation system (ITS), traffic flow prediction.**

I. Introduction

URBAN traffic flow prediction is aimed at estimating the number of vehicles passing through a specific observation point or region within a future time window [1], [2]. It is a fundamental problem being addressed in the intelligent transportation system (ITS) or smart city Internet of Things (IoT), with increasing attention and rapid deployment in recent years [3]. In general, accurate, reliable, and timely traffic flow information is of great significance to the efficacy of various ITS subsystems, especially, advanced transportation management systems, advanced traveler information systems, business vehicle management, and advanced public transportation systems [2], which have been all listed as the fundamental parts of ITS [4]. However, due to the highly nonlinear, complex, and stochastic natures of traffic flow data, accurate and reliable prediction of the traffic flow remains a difficult challenge [3].

Recently, deep learning has drawn a large amount of attention in research fields, such as computer vision, speech recognition, time-series analysis, etc. [5], and has made a significant improvement in traffic flow prediction as well. State-of-the-art deep learning approaches, such as the deep neural network (DNN) [6], stacked autoencoders (SAEs) [7], deep belief network (DBN) [1], long short-term memory network (LSTM) [3], and temporal convolutional network (TCN) have outperformed the traditional statistical and machine learning methods in traffic flow forecasting. However, on the one hand, the deep learning aforementioned may encounter problems, such as high computational complexity [8] (e.g., LSTM), limited learning ability over long sequences (e.g., TCN), and so on. On the other hand, the global optimization of the network hyperparameters remains a tough problem in deep learning despite its remarkable advancements. Hyperparameter optimization reduces human efforts and, moreover, it improves the performance of deep learning models. Such an enhancement is a result of the fact that each different hyperparameter has different optimums to obtain the best performance in different data sets or learning tasks [9], which has been proved in Section IV-C2 with two cases. For hyperparameter tuning, most of the existing deep learning models use a grid search strategy or a random search strategy (RSS) in traffic flow prediction or in other relevant problems [10], [11]. Grid search explores all potential combinations of the hyperparameters with specified grid gaps. Such an exhaustive strategy is computationally expensive in the case of many hyperparameters or large amounts of training samples, which may lead to poor performance in practice [2], [12]. In contrast, RSS scans over a lower dimensional subspace of all possible combinations with the assumption that not all hyperparameters are equally important. A previous study [12] highlighted that random search could obtain satisfactory models in most cases while with much lower computational cost.

We propose to use an improved evolutionary algorithm (EA) for deep learning hyperparameters optimization and in particular to obtain the optimal hyperparameters for the N-BEATS network. N-BEATS [13] is a cutting-edge paradigm that has proven excellent nonlinear mapping ability on challenging benchmarking data sets, and it is computationally efficient [14]. To enhance the time efficiency of N-BEATS model optimization, an improved EA, named adaptive reinitialized

Manuscript received 29 April 2022; revised 29 August 2022; accepted 21 September 2022. Date of publication 5 October 2022; date of current version 24 January 2023. (*Corresponding author: Jinyan Li.*)

Xiaocai Zhang is with the Institute of High Performance Computing, Agency for Science, Technology and Research (A*STAR), Singapore 138632 (e-mail: zhang_xiaocai@ihpc.a-star.edu.sg).

Zhixun Zhao is with the National Key Laboratory of Science and Technology on Blind Signal Processing, Chengdu 610041, China (e-mail: zhaozhixun1991@163.com).

Jinyan Li is with the Data Science Institute, Faculty of Engineering and Information Technology, University of Technology Sydney, Ultimo, NSW 2007, Australia (e-mail: jinyan.li@uts.edu.au).

Digital Object Identifier 10.1109/JIOT.2022.3212056

TABLE I
SUMMARY OF PARAMETRIC METHODS IN TRAFFIC FLOW PREDICTION

Reference	Author	Method
[15]	Ahmed et al.	ARIMA
[16]	Kumar et al.	Seasonal ARIMA
[17]	Ding et al.	Space-time ARIMA
[18]	Lee et al.	Subset ARIMA
[19]	Min et al.	VARMA
[20]	Tan et al.	ARIMA + ES
[21]	Guo et al.	KF
[22]	Wang et al.	Chaos theory + wavelet transform

differential evolution (ARDE), is proposed with the purpose of accelerating convergence speed and also improving searching ability. In summary, the research motivations of this work are concluded as follows.

1) Deep learning has performed well in traffic flow prediction, but more advanced tuning techniques for it have not been investigated thoroughly.
2) N-BEATS is a cutting-edge paradigm that has been proved with significant advancements in time-series modeling.
3) An enhanced time-efficient and robust optimization algorithm is required for deep learning model optimization.

Our new deep learning framework is termed ARDE-N-BEATS, standing for ARDE-based N-BEATS for traffic flow prediction. The main contributions of this study are summarized as follows.

1) This is the first study that leverages EA to automatically optimize the deep architecture of N-BEATS.
2) We develop an improved EA based on differential evolution (DE). We demonstrate how the newly proposed EA enables us to discover the unknown optimal deep learning architecture without much prior knowledge.
3) We conduct extensive tests and comparisons on three real-world traffic flow data sets to confirm the effectiveness and superiority of ARDE-N-BEATS in the traffic flow forecasting problem.

The remainder of this article is organized as follows. Section II delivers a detailed review of traffic flow prediction. Then, details of the proposed ARDE-N-BEATS framework are elaborated in Section III, including preliminary definitions, N-BEATS, ARDE, and ARDE-N-BEATS for traffic flow prediction. Section IV reports experimental results and analysis. Section V concludes this work and presents future work.

II. RELATED WORK

Traffic flow prediction has been intensively investigated by many researchers, because of its importance to ITS [2]. The reviewed studies can be categorized into two groups: the earlier parametric methods and the later nonparametric methods. The parametric methods include autoregressive integrated moving average (ARIMA), Kalman filtering (KF), chaotic time-series analysis, etc. The nonparametric methods consist of k-nearest neighbor (k-NN), support vector regression (SVR), Bayesian network, neural network (NN), and so on. Tables I and II provide a high-level overview of the parametric and nonparametric methods, respectively.

TABLE II
SUMMARY OF NONPARAMETRIC METHODS IN TRAFFIC FLOW PREDICTION

Reference	Author	Method
[23]	Davis et al.	k-NN
[24]	Castro et al.	SVR
[25]	Sun et al.	Bayesian network
[26]	Wang et al.	BCM (i.e., ARIMA + KF + BPNN)
[7]	Lv et al.	SAE
[27]	Yang et al.	LM-based SAE
[1]	Huang et al.	DBN
[6]	Qu et al.	DNN
[3]	Tian et al.	LSTM
[28]	Luo et al.	k-NN + LSTM
[29]	Ma et al.	LSTM + bidirectional LSTM
[30]	Wang et al.	Seq2seq LSTM
[31]	Mihaita et al.	CNN + LSTM
[32]	Wu et al.	CNN + LSTM
[33]	Tian et al.	Multiscale temporal smoothing + LSTM
[34]	Yang et al.	Attention-based LSTM
[35]	Zhao et al.	TCN
[36]	Liu et al.	GRU + FL
[37]	Yan et al.	GNN
[38]	Zhang et al.	GCN
[39]	Qi et al.	GCN + asynchronous graph convolution
[40]	Li et al.	DCRNN (i.e., bidirectional random walks + RNN + encoder-decoder)
[41]	Ye et al.	Transformer

Models for short-term traffic flow forecasting have first been built using the ARIMA [15], a conventional parametric technique. Following that, to enhance the effectiveness of traffic flow prediction, other advanced models based on ARIMA were devised, including seasonal ARIMA [16], space-time ARIMA [17], subset ARIMA [18], vector autoregressive moving average (VARMA) [19], etc. A study of ARIMA with exponential smoothing (ES) for traffic flow forecasting was explored by Tan et al. [20]. In addition to ARIMA-based methods, KF [21], chaos theory, and wavelet transform [22] have also been probed in urban traffic flow prediction. However, as the complex and stochastic time-varying relationship between past traffic and present traffic, parametric methods aforementioned cannot depict it precisely with the quite limited distributional assumptions [2].

Over the past few decades, nonparametric approaches have drawn increased attention. Davis and Nihan [23] adopted k-NN for forecasting highway traffic flow and speculated that k-NN would produce more competitive results with larger amounts of training data. Castro-Neto et al. [24] addressed the traffic flow prediction problem by utilizing SVR, which is notable for its tremendous generalization ability and capability of ensuring the global minima. However, SVR is sensitive to noise and may underperform when the input data contains high level of noise [42]. In [25], a Bayesian network was developed for modeling traffic flow data, where the traffic flow data used for prediction was performed as the cause nodes while the flow to be predicted was taken as the effect node, and the joint probability distribution between cause and effect nodes was characterized as a Gaussian mixture model. The merit of the message passing mechanism in the Bayesian network enables traffic flow modeling even with incomplete data. Furthermore, a Bayesian combination method (BCM) was formulated by Wang et al. [26] to incorporate three different predictors,

i.e., ARIMA, KF, and backpropagation NN (BPNN), to further enhance the performance. BCM is capable of tuning the credits of its component predictors more quickly, as it is sensitive to the perturbed accuracy, resulting in a good prediction performance.

In recent years, NN, especially, those using the deep architecture, has been intensively involved in the field of traffic data analysis due to their remarkable representation learning ability [2]. Lv et al. [7] highlighted an SAE model for urban traffic flow features learning, which proved to outperformed BPNN, SVR, and random walk in terms of prediction accuracy. Yang et al. [27] developed an SAE model with the Levenberg–Marquardt algorithm to further enhance performance when utilizing the Taguchi method for network structure optimization. The experiments show that it can achieve about 90% accuracy in traffic flow prediction. As stated by Huang et al. [1], a DBN is capable of learning traffic flow features with limited prior knowledge. Based on the reported results, it obtains approximately 5% improvement over several parametric methods, Bayesian model, SVR and BPNN. Qu et al. [6] addressed the long-term traffic flow prediction problem by using historical traffic flow data as well as contextual factor data to train a DNN. It can get more robust and superior predictions than the national cooperative highway research program (NCHRP) Report 765. Recurrent NN (RNN) is a type of NN in which at least one cycle exists in the connection graph. Currently, RNN based on LSTM is the most used [43]. LSTM network was adopted for traffic flow prediction in [3]. LSTM is, especially, capable of learning long-term interdependencies to overcome the gradient vanishing problem in RNN, making it outperform most of the nonparametric methods on sequential data [2]. The research work by Luo et al. [28] demonstrated a hybrid model of k-NN and LSTM for spatial-temporal traffic flow forecasting, in which the k-NN is designed to capture the spatial features and the LSTM is to learn the temporal relationships. The newly constructed hybrid model can achieve more accurate predictions than LSTM. Ma et al. [29] established a hybrid network based on LSTM and bidirectional LSTM, which can enhance the stability, especially, for longer time-series data modeling. In [30], a sequence-to-sequence (seq2seq) structure with LSTM was investigated to minimize the accumulation of error propagation for long-term traffic flow forecasting. Meanwhile, deep learning architecture based on LSTM and convolutional NN (CNN) was proposed in [31] and [32] to learn the spatial–temporal characteristics of traffic flow. However, the study [31] proved that the hybrid model based on CNN and LSTM underperforms when compared with the individual LSTM model. A study [33] focused on traffic flow prediction with missing data. A multiscale temporal smoothing is used to infer missing data while a modified LSTM is introduced to learn the prediction residual. The experimental results show that it outperforms other baselines, such as LSTM, SAE, SVR, etc. Yang et al. [34] improved the LSTM network with an attention mechanism, which enables the NN to automatically focus on different parts of its inputs [44]. This method exhibits competitive performance for short-term traffic flow prediction. TCN was first introduced in short-term traffic flow forecasting in [35], showing great improvement over LSTM and SAE. TCN is capable of modeling sequential data, because of the novel mechanisms introduced, including causal convolution, dilated convolution and residual block. The study work by Ketabi et al. [45] provided a comparative analysis of several variants of RNN and conventional methods for spatial–temporal traffic forecasting. The finding indicates that deep learning models outperform the conventional models by 8.2% in terms of the average percentage error of all cases. Liu et al. [36] revealed a federated learning (FL) paradigm based on gated recurrent unit (GRU) network for traffic flow prediction with the purpose of preserving privacy. The experimental results display competitive predictions with user privacy well preserved. A Chebyshev graph NN (GNN) was implemented in [37] to capture the spatial dependencies by learning the complex topological structures among the traffic network. In [38], a graph convolutional network (GCN) with attention mechanism was highlighted for citywide spatial–temporal traffic flow prediction by modeling the region dependencies. Qi et al. [39] presented a GCN framework based on a novel asynchronous spatial–temporal graph convolution operation to extract the relationship in transportation network. Experiments show that the proposed model outperforms existing methods, especially, for long-time prediction. Moreover, it is robust and capable of handling data containing noise. Besides the GCN in spatial–temporal traffic prediction, a novel diffusion convolutional RNN (DCRNN) was constructed by Li et al. [40]. It captures the spatial dependency using bidirectional random walks on the graph. Then, the temporal dependency is learned by the encoder–decoder structure. In [41], a transformer network is built for spatial–temporal traffic flow prediction, which includes three different types of attention layers to jointly capture the temporal and spatial correlations among data. Such implementation of architecture can obtain a significant improvement over the second-best baseline on three data sets, ranging from 1.23% to 3.92%.

III. Methodology

The workflow of our proposed deep learning is depicted in Fig. 1. It contains three modules, i.e., data processing (module 1), ARDE-N-BEATS (module 2), and traffic flow prediction (module 3). First, the entire data set is separated into a training, a validation, and a test subset. Second, ARDE-N-BEATS is then performed to optimize the N-BEATS hyperparameters using the training and validation data. Third, the traffic flow over the future time interval is predicted by the optimized N-BEATS model. For a better understanding of the difference between our ARDE-N-BEATS and the conventional frameworks, e.g., RSS and DE, Fig. 1 illustrates the workflows of the DE and RSS as well. As indicated in Fig. 1, the major differences between ARDE and DE are summarized as follows: 1) a reinitialization step is designed to retain the best genes for the subsequent procedures; 2) mutants are produced throughout generations using adaptative parameters in a more concentrated data space; and 3) an acceleration mechanism is devised to speed up convergence in the steps of reinitialization and selection. In this section, we will provide the details of

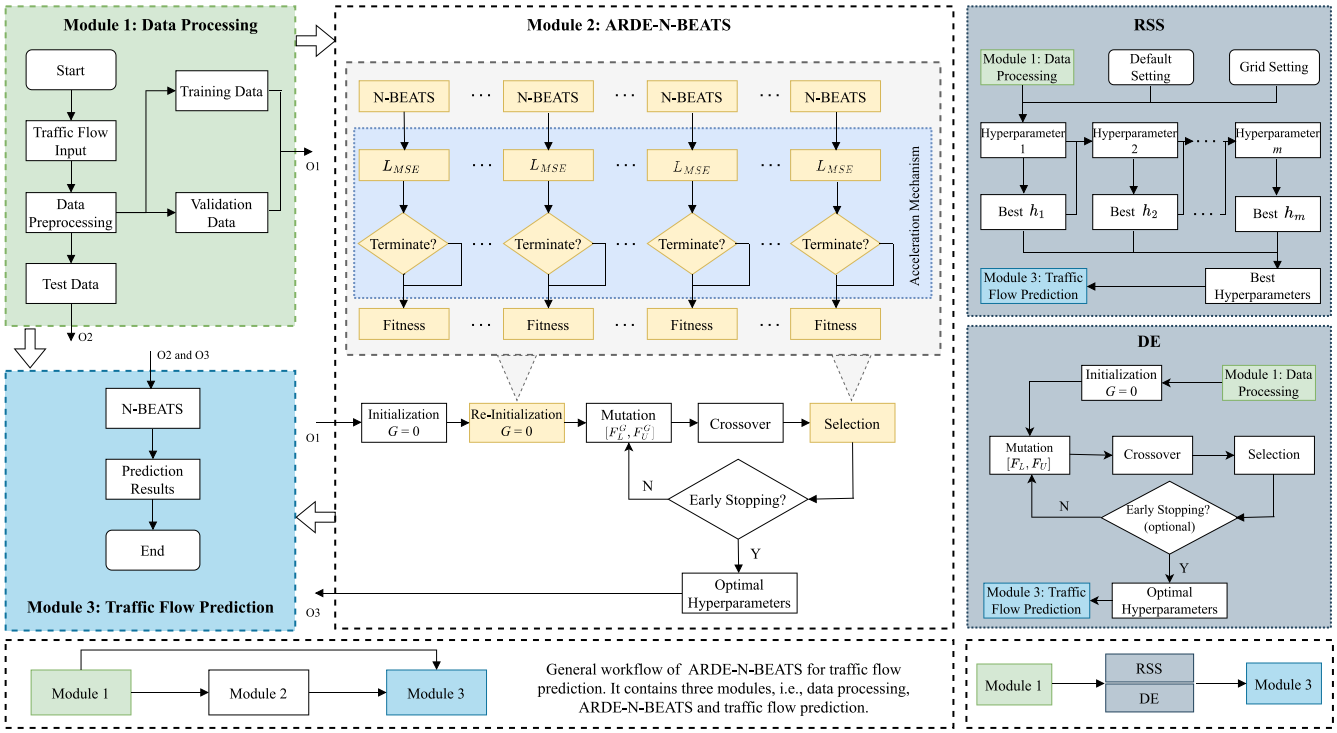


Fig. 1. Workflows of ARDE-N-BEATS as well as RSS and DE for traffic flow prediction.

N-BEATS, ARDE, and ARDE-N-BEATS framework for traffic flow forecasting, as well as some preliminary definitions and notions. To help readers better comprehend the equations in this section, a table of notations is provided in Table III.

A. Preliminaries

Definition 1 (Traffic Flow): Traffic flow time series is defined as a set of random variables $\{x_t, t \in T\}$, where random variable x_t is distributed according to some univariate probability distributions function P_t. T is a set of timestamps with equidistant time intervals Δt, and $T = \{1, 2, 3, \ldots\}$.

Definition 2 (Traffic Flow Prediction): Traffic flow prediction is to estimate future traffic volume. Given a set of fully observed traffic flow values at timestamps $\{1, 2, \ldots, t\}$ denoted as $\mathbf{x} = \{x_1, x_2, \ldots, x_t\}$, the objective of traffic flow prediction is to predict the traffic flow value at timestamp $t+1$, denoted by $\mathbf{y} = y_{t+1}$.

B. N-BEATS

N-BEATS is different from the existing RNN models that handle the time-series prediction as a seq2seq problem. It is treated as a nonlinear multivariate regression problem, and then a highly expressive NN with remarkable generalization skills is built to solve the problem [14].

N-NEATS is defined by stacking many layers of stacks and blocks with the residual principle, as displayed in Fig. 2. Assume that the N-BEATS network is comprised of s stacks, and each stack is constructed by b blocks. For the architecture of block, let $\mathbf{xb}_{n,m}$ denote the input of the mth block in stack n, we have output comprising of backcast $\hat{\mathbf{xb}}_{n,m}$ and forecast

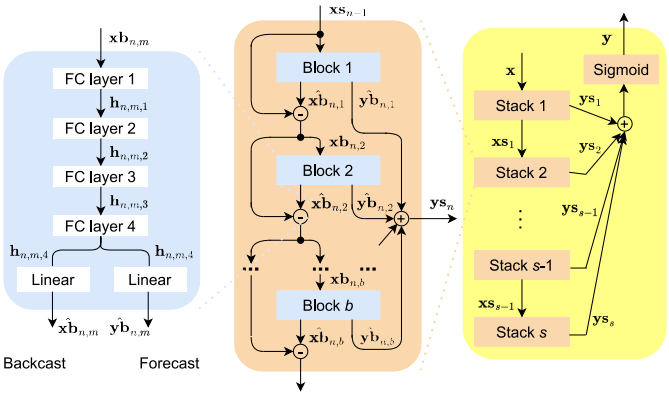


Fig. 2. Architecture of N-BEATS.

$\hat{\mathbf{yb}}_{n,m}$, which are derived by

$$\mathbf{h}_{n,m,1} = \mathrm{FC}_{n,m,1}\left(\mathbf{xb}_{n,m}\right), \mathbf{h}_{n,m,1} = \mathrm{ReLU}\left(\mathbf{h}_{n,m,1}\right) \quad (1)$$

and

$$\mathbf{h}_{n,m,2} = \mathrm{FC}_{n,m,2}\left(\mathbf{h}_{n,m,1}\right), \mathbf{h}_{n,m,2} = \mathrm{ReLU}\left(\mathbf{h}_{n,m,2}\right) \quad (2)$$

and

$$\mathbf{h}_{n,m,3} = \mathrm{FC}_{n,m,3}\left(\mathbf{h}_{n,m,2}\right), \mathbf{h}_{n,m,3} = \mathrm{ReLU}\left(\mathbf{h}_{n,m,3}\right) \quad (3)$$

and

$$\mathbf{h}_{n,m,4} = \mathrm{FC}_{n,m,4}\left(\mathbf{h}_{n,m,3}\right), \mathbf{h}_{n,m,4} = \mathrm{ReLU}\left(\mathbf{h}_{n,m,4}\right) \quad (4)$$

and

$$\hat{\mathbf{xb}}_{n,m} = \mathbf{B}_{n,m}\mathbf{h}_{n,m,4}, \hat{\mathbf{yb}}_{n,m} = \mathbf{F}_{n,m}\mathbf{h}_{n,m,4} \quad (5)$$

TABLE III
TABLE OF NOTATIONS

Notation	Definition
m	Sequence length
N	Number of observations
$\mathbf{X}$	Input data
$\mathbf{x}_j$	Single instance of input data
s, b	Number of stack and block of N-BEATS
$\mathbf{xb}_{n,m}$	Input of the mth block in stack n
$\hat{\mathbf{xb}}_{n,m}$	Backcast output of the mth block in stack n
$\hat{\mathbf{yb}}_{n,m}$	Forecast output of the mth block in stack n
$\mathbf{h}_{n,m,l}$	Output of the lth hidden layer
FC	Fully-connected layer
$\mathbf{B}_{n,m}$, $\mathbf{F}_{n,m}$	Weights
$\mathbf{ys}_n$	Output of the nth stack
$\mathbf{y}$	Output of N-BEATS
b_{L_j}, b_{U_j}	Lower and upper bounds
P^0	Population size of initialization
P^1	Population size of re-initialization
$\mathbf{z}^0$	Re-initialized population
f	Fitness function
ϵ	Fitness threshold
G	Generation
$\mathbf{v}^G$	Mutated population at the $G$the generation
F_i^G	Mutation factor
F_L^G, F_U^G	Lower and upper mutation bounds
$\mathbf{u}_i^G$	Trail vector at the $G$the generation
$\mathbf{z}_i^G$	Target vector at the $G$the generation
C_r	Crossover factor
f_G^*	The best fitness of the $G$the generation
$maxG$	Maximum generation
$\mathbf{z}^*$	Optimized hyperparameter vector
k	Patience parameter for early stopping
$\mathbf{yz}_i^G$, $\mathbf{yu}_i^G$	Output of N-BEATS under hyperparameters $\mathbf{z}_i^G$ or $\mathbf{u}_i^G$
L_{MSE}	Validation loss
$L_{threshold}$	Loss threshold
$\mathbf{w}_1$	Weights of N-BEATS at the first epoch
$\check{\mathbf{X}}$	Input from validation data
$\check{\mathbf{y}}$	Groundtruth output from validation data
L^*	Validation loss of hyperparameters with the globally best fitness
L_{margin}^0	Initial threshold
L_{margin}^G	Decay threshold at the Gth generation
$\hat{\mathbf{y}}$	Groundtruth output
g_{NBEATS}	N-BEATS model

where FC stands for the fully connected layer, and ReLU is the rectified linear unit activation function. $\mathbf{B}_{n,m}$ and $\mathbf{F}_{n,m}$ are both the weights.

The residual between the input and output of block m is then performed as the input of the block at a higher level (i.e., block $m+1$), as shown by

$$\mathbf{xb}_{n,m+1} = \mathbf{xb}_{n,m} - \hat{\mathbf{xb}}_{n,m}. \tag{6}$$

Following this, the cumulative forecast $\mathbf{ys}_n$ is taken as the output of stack n, which is defined by

$$\mathbf{ys}_n = \sum_{i=1}^{b} \hat{\mathbf{yb}}_{n,i} \tag{7}$$

then, the final output of the whole network can be obtained by summing the outputs of all stacks (i.e., s stacks), and a sigmoid function is appended for nonlinear transformation, as indicated by

$$\mathbf{y} = \text{sigmoid}\left(\sum_{i=1}^{s} \mathbf{ys}_i\right). \tag{8}$$

Algorithm 1 ARDE Algorithm

Parameters: P^0, P^1, ϵ, $\{F_L^i, i \in \{0, 1, \ldots, maxG-1\}\}$, $\{F_U^i, i \in \{0, 1, \ldots, maxG-1\}\}$, b_L, b_U, C_r, $maxG$, k
Input: f, $\mathbf{X}$
Output: $\mathbf{z}^*$

for G in $\{0, 1, \ldots, maxG-1\}$ **do**
  if $G == 0$ **then**
    $\mathbf{z}^0 \leftarrow initialization(P^0, b_L, b_U)$;
    $\mathbf{z}^0 \leftarrow re\text{-}initialization(\mathbf{z}^0, P^1, \epsilon)$;
    $\mathbf{v}^0 \leftarrow mutation(\mathbf{z}^0, F_L^0, F_U^0, b_L, b_U)$;
    $\mathbf{u}^0 \leftarrow crossover(\mathbf{v}^0, \mathbf{z}^0, C_r)$;
    $\mathbf{z}^1 \leftarrow selection(\mathbf{u}^0, \mathbf{z}^0, \mathbf{X}, f)$;
  else
    $\mathbf{v}^G \leftarrow mutation(\mathbf{z}^G, F_L^G, F_U^G, b_L, b_U)$;
    $\mathbf{u}^G \leftarrow crossover(\mathbf{v}^G, \mathbf{z}^G, C_r)$;
    $\mathbf{z}^{G+1} \leftarrow selection(\mathbf{u}^G, \mathbf{z}^G, \mathbf{X}, f)$;
    if $earlyStopping(k, \mathbf{X}, f)$ **then**
      break
    end if
  end if
end for
$\mathbf{z}^* \leftarrow optimalHyperParameters(f, \mathbf{z}^{G+1}, \mathbf{X})$
return $\mathbf{z}^*$

C. Adaptive Reinitialized Differential Evolution

The formulated ARDE algorithm (Algorithm 1) consists of six steps: 1) initialization; 2) reinitialization; 3) mutation; 4) crossover; 5) selection; and 6) early stopping.

1) Initialization: The step of initialization is only applicable for the first generation (iteration). Suppose there are totally D parameters to be optimized for a given data set $\mathbf{X}$. Let $[b_{L_j}, b_{U_j}]$ stand for the range of the jth parameter, $j = 1, 2, \ldots, D$, where L and U denote the lower and upper bounds, respectively. The population is initialized with the following equations:

$$\mathbf{z}_{ij}^0 = b_{L_j} + \text{rand}(0, 1) * \left(b_{U_j} - b_{L_j}\right) \tag{9}$$

and

$$\mathbf{z}_i^0 \in \mathbb{R}^{P^0} \tag{10}$$

where $i = 1, 2, \ldots, P^0$; P^0 is the initialized population size; rand(0, 1) is the function to generate a random number between 0 and 1 with a uniform probability distribution [2].

2) Reinitialization: The aforementioned step of initialization randomly scans a subset of the data space to form the initial gene and chromosome candidates. The stochasticity brought by the random selection might produce unfavorable genes. In order to keep the best genes for the following mutation and crossover steps, we reinitialize the selected population by introducing a fitness threshold ϵ. Let $f(\mathbf{z}_i^0)$, where

$i \in \{1, 2, \ldots, P^0\}$, stand for the fitness value of chromosome candidate $\mathbf{z}_i^0$, if

$$f\left(\mathbf{z}_i^0\right) < \epsilon \tag{11}$$

then, $\mathbf{z}_i^0$ will be picked out to form the reinitialized population. Once the reinitialized population size is over P^1, the reinitialization procedure will be terminated, as indicated by

$$\mathbf{z}_i^0 \in \mathbb{R}^{P^1} \tag{12}$$

where $i = 1, 2, \ldots, P^1$; P^1 is a preset parameter for determining the reinitialization population size.

3) Mutation: The step of mutation is to add a scaled vector difference between two randomly chosen chromosomes to a third chromosome [2]. The mutation is performed using

$$\mathbf{v}_{ij}^G = \mathbf{z}_{kj}^G + F_i^G * \left(\mathbf{z}_{lj}^G - \mathbf{z}_{mj}^G\right) \tag{13}$$

and

$$\mathbf{z}_{ij}^G \in \mathbb{R}^{P^1} \tag{14}$$

and

$$F_i^G = \text{random}\left(F_L^G, F_U^G\right) \tag{15}$$

where superscript G denotes the Gth generation of the algorithm; $1 \le i \ne k \ne l \ne m \le P^1$; and F_i^G is the mutation factor produced randomly from the uniform distribution on the interval $[F_L^G, F_U^G]$, where F_L^G and F_U^G are two adaptive parameters for the Gth generation. The two mutation bounds for the next generation (i.e., the $G+1$th generation) are updated according to

$$F_L^{G+1} < F_U^{G+1} <= F_L^G < F_U^G. \tag{16}$$

The aim of such implementation is to produce mutants in a more concentrated data space over generations.

4) Crossover: The purpose of crossover is to exchange genetic materials between the initialized and mutated chromosomes. Thus, it can produce new recombinant chromosomes, as referred to trial vectors as well. The step of crossover in the ARDE algorithm is formulated as

$$\mathbf{u}_{ij}^G = \begin{cases} \mathbf{v}_{ij}^G, & \text{if } c_{ij}^G \ge C_r \\ \mathbf{z}_{ij}^G, & \text{if } c_{ij}^G < C_r \end{cases} \tag{17}$$

and

$$c_{ij}^G = \text{rand}(0, 1) \tag{18}$$

where $\mathbf{u}_{ij}^G$ is the new trial vector after crossover; $C_r \in [0, 1]$ is the crossover factor; and rand(0, 1) generates a random value uniformly distributed on the interval between 0 and 1.

5) Selection: The selection operation aims to select the best genes for offspring. The selection is conducted by comparing the fitness values of the trial vector $\mathbf{u}_i^G$ and the target vector $\mathbf{z}_i^G$. If the fitness value of trial vector $\mathbf{u}_i^G$ is better than that of target vector $\mathbf{z}_i^G$, then replace the target vector with trial vector $\mathbf{u}_i^G$ for the offspring; otherwise, keep target vector $\mathbf{z}_i^G$ to the next step [2]. Thereby, the offspring is obtained by

$$\mathbf{z}_i^{G+1} = \begin{cases} \mathbf{u}_i^G, & \text{if } f\left(\mathbf{u}_i^G; \mathbf{X}\right) \le f\left(\mathbf{z}_i^G; \mathbf{X}\right) \\ \mathbf{z}_i^G, & \text{if } f\left(\mathbf{u}_i^G; \mathbf{X}\right) > f\left(\mathbf{z}_i^G; \mathbf{X}\right). \end{cases} \tag{19}$$

6) Early Stopping: Suppose the derived best fitness for the $G+1$th generation is represented by

$$f_{G+1}^* = \min\left(\left\{f\left(\mathbf{z}_i^{G+1}; \mathbf{X}\right) : i \in \left\{1, 2, \ldots, P^1\right\}\right\}\right). \tag{20}$$

In order to improve the time efficiency of the hyperparameter optimization process, we introduce a criterion of early stopping mechanism. First, a patience parameter k is given, if the best fitness of the $G+1$th generation (i.e., f_{G+1}^*) is the same as the best fitness of the previous $k-1$ generations, the early stopping mechanism will be triggered, as formulated by

$$f_{G+1}^* = f_G^* = \cdots = f_{G-k+2}^* \tag{21}$$

where $G \ge k-1$ and $G \le maxG - 1$.

The global optimal hyperparameter vectors $\mathbf{z}^*$ are determined using

$$\mathbf{z}^* = \underset{\mathbf{z} \in \mathbb{R}^D}{\arg\min}\left(\left\{f\left(\mathbf{z}_i^{G^*}; \mathbf{X}\right) : i \in \left\{1, 2, \ldots, P^1\right\}\right\}\right) \tag{22}$$

where G^* denotes the generation that the terminal condition is applied.

D. ARDE-N-BEATS for Traffic Flow Prediction

Suppose at timestamp $t+m$, we have observed the traffic flow data at timestamps $(t+1, t+2, \ldots, t+m)$. The input of the model is denoted as $\mathbf{X}$, which is represented by

$$\mathbf{x}_j = \left(x_{j1}, x_{j2}, \ldots, x_{jm}\right)^T \in \mathbb{R}^m \tag{23}$$

and

$$\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{N-1}, \mathbf{x}_N) \in \mathbb{R}^{m \times N} \tag{24}$$

where m stands for the sequence length (SL) and N is the number of observations.

In the steps of reinitialization and selection, we devise a novel acceleration mechanism to speed up the convergence of the ARDE algorithm. Let $\mathbf{yz}_i^G$ denote the output via the N-BEATS model under hyperparameters $\mathbf{z}_i^G$ at the step of initialization or selection. Accordingly, $\mathbf{yu}_i^G$ represents the output with hyperparameters derived from mutation (i.e., $\mathbf{u}_i^G$), as demonstrated by

$$\mathbf{yz}_i^G = g_{\text{NBEATS}}\left(\mathbf{X}, \mathbf{z}_i^G\right) \tag{25}$$

and

$$\mathbf{yu}_i^G = g_{\text{NBEATS}}\left(\mathbf{X}, \mathbf{u}_i^G\right) \tag{26}$$

where $\mathbf{X}$ stands for the input data; g_{NBEATS} represents the N-BEATS model with the hyperparameters vector $\mathbf{z}$ or $\mathbf{u}$.

Let $L_{\text{MSE}}(\mathbf{w}_1, \check{\mathbf{X}}, \check{\mathbf{y}}, \mathbf{z}_i^G)$ and $L_{\text{MSE}}(\mathbf{w}_1, \check{\mathbf{X}}, \check{\mathbf{y}}, \mathbf{u}_i^G)$ stand for the validation loss of the first epoch while training an N-BEATS model over hyperparameters $\mathbf{z}^G$ and $\mathbf{u}^G$, respectively. In this study, the mean square error (MSE) is adopted as the loss function. $\mathbf{w}_1$ stands for the network's weights learned after the first training epoch. $\check{\mathbf{X}}$ and $\check{\mathbf{y}}$ represents the input and ground-truth output, respectively, from the validation data. If the validation loss exceeds a dynamic threshold $L_{\text{threshold}}$, as shown by

$$L_{\text{MSE}}\left(\mathbf{w}_1, \check{\mathbf{X}}, \check{\mathbf{y}}, \mathbf{z}_i^G\right) > L_{\text{threshold}} \tag{27}$$

and

$$L_{\text{MSE}}\left(\mathbf{w}_1, \check{\mathbf{X}}, \check{\mathbf{y}}, \mathbf{u}_i^G\right) > L_{\text{threshold}} \tag{28}$$

we assume that the trial hyperparameter set is not favorable, and then the training process will be terminated after the first training epoch. This dynamic threshold is determined by

$$L^* = \arg\min_{L_{\text{MSE}}}\left(f\left(\left\{\mathbf{z}^G, \mathbf{u}^G\right\}, \mathbf{X}\right)\right) \tag{29}$$

and

$$L_{\text{threshold}} = L^* + L_{\text{margin}}^G \tag{30}$$

and

$$L_{\text{margin}}^G = L_{\text{margin}}^0 * \left(f_{\text{margin}}\right)^G \tag{31}$$

where L_{margin}^G is a decayed threshold depending on the generation G, a preset initial threshold L_{margin}^0, and a preset decay factor f_{margin}; and L^* is a dynamic variable that is updated according to the hyperparameters with the globally best fitness value along the whole optimization process.

Assume that the outputs $\mathbf{yz}_i^G$ and $\mathbf{yu}_i^G$ are comprised by

$$\mathbf{yz}_i^G = \left(yz_{i1}^G, yz_{i2}^G, \ldots, yz_{i(N-1)}^G, yz_{iN}^G\right) \in \mathbb{R}^N \tag{32}$$

and

$$\mathbf{yu}_i^G = \left(yu_{i1}^G, yu_{i2}^G, \ldots, yu_{i(N-1)}^G, yu_{iN}^G\right) \in \mathbb{R}^N \tag{33}$$

we use a minibatch stochastic gradient decent (SGD) together with an Adam optimizer to train the N-BEATS model. Suppose the ground truth of traffic flow is denoted as

$$\hat{\mathbf{y}} = \left(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{N-1}, \hat{y}_N\right) \in \mathbb{R}^N \tag{34}$$

we adopt the metric of mean absolute percentage error (MAPE) to determine the fitness (i.e., the f function in Section III-C) in this study, as defined by

$$f\left(\mathbf{z}_i^G, \hat{\mathbf{y}}\right) = \frac{1}{N}\sum_{j=1}^{N}\frac{\left|yz_{ij}^G - \hat{y}_j\right|}{\left|\hat{y}_j\right|} \cdot 100\% \tag{35}$$

and

$$f\left(\mathbf{u}_i^G, \hat{\mathbf{y}}\right) = \frac{1}{N}\sum_{j=1}^{N}\frac{\left|yu_{ij}^G - \hat{y}_j\right|}{\left|\hat{y}_j\right|} \cdot 100\%. \tag{36}$$

Following that, the offspring $\mathbf{z}_i^{G+1}$ can be obtained, as illustrated by the step of selection in Section III-C. In accordance with (22), the optimization problem is solved with the global optimal hyperparameters vector $\mathbf{z}^*$. Then, for a given input $\mathbf{x}$, the future traffic flow value is predicted by

$$\mathbf{y} = g_{\text{NBEATS}}\left(\mathbf{x}, \mathbf{z}^*\right). \tag{37}$$

The details of the aforementioned ARDE-N-BEATS framework for traffic flow forecasting are explained by Algorithm 2.

Algorithm 2 ARDE-N-BEATS for Traffic Flow Prediction

Parameters: L_{margin}^0, f_{margin}
Input: $\mathbf{X}$, $\hat{\mathbf{y}}$, $\check{\mathbf{X}}$, $\check{\mathbf{y}}$, $\mathbf{x}$
Output: y

for G in ARDE algorithm **do**
  if step is re-initialization **or** step is selection **then**
    $L_{margin}^G \leftarrow L_{margin}^0 * (f_{margin})^G$
    for $\mathbf{s} \in \{\mathbf{z}^G, \mathbf{u}^G\}$ **do**
      if $\mathbf{s}$ is the first hyperparameter set **then**
        $L_{MSE}(\mathbf{w}_1, \check{\mathbf{y}}, \check{\mathbf{X}}, \mathbf{s}) \leftarrow$ *first epoch validation loss*;
        $L^* \leftarrow L_{MSE}(\mathbf{w}_1, \check{\mathbf{y}}, \check{\mathbf{X}}, \mathbf{s})$;
      else
        $L_{MSE}(\mathbf{w}_1, \check{\mathbf{y}}, \check{\mathbf{X}}, \mathbf{s}) \leftarrow$ *first epoch validation loss*;
        if $L_{MSE}(\mathbf{w}_1, \check{\mathbf{y}}, \check{\mathbf{X}}, \mathbf{s}) > L^* + L_{margin}^G$ **then**
          terminate N-BEATS training;
        end if
        $f(\mathbf{s}, \hat{\mathbf{y}}) \leftarrow$ *calculate fitness value*;
        if $f(\mathbf{s}, \hat{\mathbf{y}})$ is the global minimum **then**
          $L^* \leftarrow L_{MSE}(\mathbf{w}_1, \check{\mathbf{y}}, \check{\mathbf{X}}, \mathbf{s})$;
        end if
      end if
    end for
  end if
  if step is early stopping **then**
    $\mathbf{z}^* \leftarrow optimalHyperParameters(f, \mathbf{z}^{G+1}, \mathbf{X})$;
  end if
end for
$\mathbf{y} \leftarrow bestModelPrediction(\mathbf{x}, \mathbf{z}^*)$;
return $\mathbf{y}$

IV. Experiments and Analysis

A. Data Sets

Three real-world traffic flow data sets were collected for evaluation. We choose roads with dense traffic flow for study because they are given much attention in both traffic management and research [1]. Two of them are about the traffic flow of two motorways in Dublin, Ireland, and the third is about a main road in San Francisco, USA. The two Irish data sets were downloaded from the database maintained by the transport infrastructure Ireland (TII),[1] and the San Francisco data set was obtained from the Caltrans Performance measurement system (PeMS) database.[2] Both databases have also been utilized for experiments by other relevant work [36], [46]. All the traffic flow data were originally collected at a 15-min interval continuously spanning one year from 1 April 2018 to 31 March 2019, using the average of all the inductive loop detectors in the corresponding road [2]. Following this, the 15-min format of data is transformed into a 30-min ($\Delta t = 30$), a 45-min ($\Delta t = 45$), and a 60-min ($\Delta t = 60$) interval format for different tasks of prediction. For the 15-min prediction task, the forecasting horizon is 15 min, and the sampling interval for the

[1] https://trafficdata.tii.ie
[2] http://pems.dot.ca.gov

TABLE IV
DATA SETS DESCRIPTION

Data Set	Road	Segment	Direction	Detector No
M50-N	M50	Balinteer to Finglas	Northbound	8
M1-N	M1	Airport to Swords	Northbound	2
I280-S	I280	Bernal Heights to Ingleside	Southbound	7

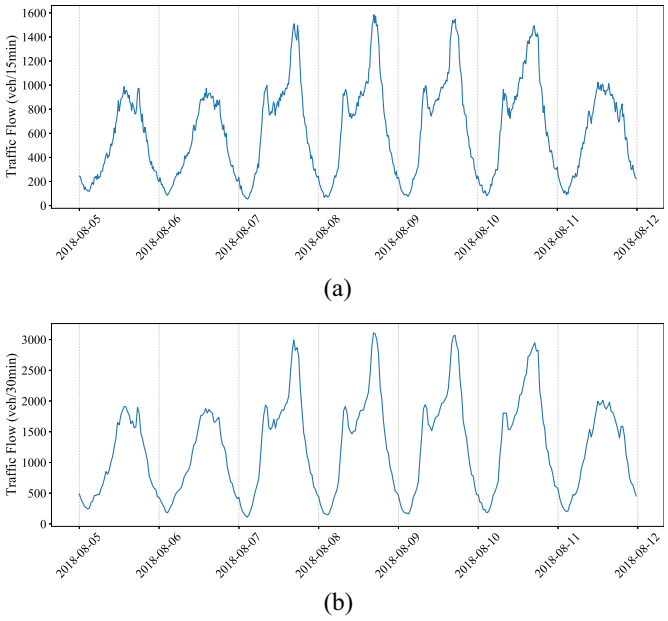


Fig. 3. Plots for the 15 and 30-min format traffic flow trends of the M1-N data set during a typical week (from 5 August 2018 to 11 August 2018). (a) M1-N (15 min). (b) M1-N (30 min).

time-series data is also 15 min. The rest prediction tasks, such as 30, 45, and 60-min prediction tasks, are specified in a similar way. Table IV provides more details about these data sets. Two examples of the average flow of the 15-min interval (unit: veh/15 m), as well as the 30-min interval (unit: veh/30 m) during a typical week have been depicted in Fig. 3.

B. Evaluation Metrics

Three metrics are used to evaluate the proposed method's and existing methods' prediction performance: mean absolute error (MAE), root MSE (RMSE), and MAPE. They are described as follows:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right| \tag{38}$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2} \tag{39}$$

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\frac{\left|y_i - \hat{y}_i\right|}{\hat{y}_i}\cdot 100\% \tag{40}$$

where n is the sample size of the test data, y_i denotes the predicted traffic flow during the predefined time interval, and $\hat{y}_i$ indicates the ground-truth traffic flow during that interval.

TABLE V
PERFORMANCE FROM THE N-BEATS MODEL WITH TWO DIFFERENT SETS OF HYPERPARAMETERS

Set	Fitness (%)	MAE	RMSE	MAPE (%)
Set 1	6.85	59.78	90.25	6.94
Set 2	13.86	98.20	136.60	11.79

C. Experimental Settings

1) Data Set Partition: All data sets are divided into three subsets: a training set, a validation set, and a test set. For each data set, the data from 18 January 2019 onward (i.e., approximately 20%) are taken as the test set. Accordingly. the training and validation sets are formed by the rest data. The training and validation sets are randomly sampled with a ratio of 8:2.

2) Hyperparameters: N-BEATS is a complicated architecture with lots of hyperparameters. In this study, we choose the nine most important hyperparameters of the N-BEATS model, including SL, block per stack (BPS), stake number (SN), hidden unit (HU), thetas dim (TD), batch size (BS), learning rate (LR), LR decay factor (LDF), and LR decay patience (LDP). SL, BPS, SN, HU, and TD are model hyperparameters for determining the architecture of N-BEATS, where SL, BPS, and SN are referred to as m, b, and s explained in Section III, respectively. While the rest are optimizer hyperparameters that control the training procedure. The selection of significant hyperparameters is essential, as different sets of hyperparameters exhibit different effects on the performance of N-BEATS model. Thereby, figuring out the best hyperparameters can maximally exploit the learning capability of N-BEATS model. An example of two different hyperparameters on the same data set (i.e., M50-N 30-min task) is demonstrated in Table V. For set 1, the hyperparameters are set as $\{\text{SL, BPS, SN, HU, TD, BS, LR, LDF, LDP}\} = \{8, 3, 2, 81, 6, 302, 0.0138, 0.1248, 7\}$. For set 2, we have a different hyperparameter set of $\{\text{SL, BPS, SN, HU, TD, BS, LR, LDF, LDP}\} = \{38, 2, 4, 23, 14, 44, 0.0243, 0.1488, 10\}$. From the forecasting performance reported in Table V, we can have an observation that the performance of these two sets of hyperparameters differs substantially even on the same data set.

3) Parameter Settings: As mentioned, we have a total of nine hyperparameters for optimization in this study (i.e., $D = 9$). The hyperparameter range is mainly determined based on the following principles: 1) ranges for some hyperparameters are recommended by a relevant work [47]; 2) we attempt to assign a higher upper bound to all hyperparameters to skip tuning the bound limits, and more importantly, to enhance the applicability of our constructed methods in practice. However, it is also important to strike a balance between predictive performance and computational efficiency; and 3) ranges for some hyperparameters are set empirically based on DNN training. Thus, the range of SL is set as $[1, 50]$; BPS and SN are both set as $[1, 10]$. HU is chosen from 1 to 500, and TD is produced within $[1, 20]$. BS is generated between 10 and 512. Note that BS below 10 is suggested to be discarded with the purpose of speeding up the training process [2]. Hyperparameters of LR, LDF, and LDP for

TABLE VI
LOWER AND UPPER BOUNDS OF EACH SELECTED HYPERPARAMETER

SL		BPS		SN		HU		TD		BS		LR		LDF		LDP	
b_{L_1}	b_{U_1}	b_{L_2}	b_{U_2}	b_{L_3}	b_{U_3}	b_{L_4}	b_{U_4}	b_{L_5}	b_{U_5}	b_{L_6}	b_{U_6}	b_{L_7}	b_{U_7}	b_{L_8}	b_{U_8}	b_{L_9}	b_{U_9}
1	50	1	10	1	10	1	500	1	20	10	512	0.0001	0.1	0	1	1	10

TABLE VII
PERFORMANCE COMPARISON BETWEEN DIFFERENT OPTIMIZATION METHODS

Data	Model	15-min			30-min			45-min			60-min		
		MAE	RMSE	MAPE (%)	MAE	RMSE	MAPE (%)	MAE	RMSE	MAPE (%)	MAE	RMSE	MAPE (%)
M50-N	RSS-N-BEATS	24.48	36.40	5.68	52.35	77.94	6.47	79.37	120.02	6.39	108.57	164.07	6.65
	BO-N-BEATS	21.97	33.34	**4.86**	47.89	73.37	5.63	70.94	110.67	5.46	98.86	156.41	5.89
	ACO-N-BEATS	23.38	36.01	5.13	50.59	80.90	5.68	70.13	108.24	5.30	99.10	**153.08**	5.65
	PSO-N-BEATS	24.32	36.39	5.47	48.26	74.39	5.57	69.08	108.46	4.98	99.09	154.07	5.50
	DE-N-BEATS	22.90	35.23	5.06	48.59	**73.27**	6.46	**67.26**	**108.19**	5.11	116.61	174.23	7.58
	ARDE-N-BEATS	**21.86**	**33.30**	5.04	**47.45**	73.49	**5.20**	67.82	109.58	**4.85**	**96.77**	157.11	**5.11**
M1-N	RSS-N-BEATS	32.88	47.39	7.15	59.18	91.55	6.75	91.88	140.66	7.23	109.67	157.84	6.53
	BO-N-BEATS	31.23	45.83	7.15	56.42	90.01	6.17	81.43	128.33	6.35	109.75	161.34	6.76
	ACO-N-BEATS	29.66	**43.83**	6.74	62.53	99.80	6.77	74.90	122.25	5.45	98.71	148.34	5.46
	PSO-N-BEATS	30.95	45.26	6.94	53.72	83.67	6.09	76.92	124.29	5.77	100.99	149.96	5.66
	DE-N-BEATS	30.82	45.69	6.88	54.24	86.47	6.39	81.52	132.36	6.27	104.93	158.51	6.17
	ARDE-N-BEAS	**28.92**	43.93	**6.19**	**52.58**	**83.11**	**5.64**	**74.45**	**119.63**	**5.15**	**96.06**	**147.64**	**5.38**
I280-S	RSS-N-BEATS	41.28	60.23	5.89	84.77	123.39	6.28	130.91	198.17	6.57	160.95	236.34	5.93
	BO-N-BEATS	39.58	57.82	5.68	77.07	115.01	5.56	**110.40**	174.28	**5.31**	152.86	225.84	5.88
	ACO-N-BEATS	41.10	59.88	5.83	76.23	113.73	5.42	111.16	**173.10**	5.46	**152.53**	**223.39**	5.77
	PSO-N-BEATS	40.96	60.12	5.82	75.68	114.18	**5.32**	112.57	176.83	5.51	166.22	245.65	6.11
	DE-N-BEATS	**39.32**	57.49	5.72	**74.64**	**111.77**	5.46	116.47	180.09	5.48	155.59	226.79	5.92
	ARDE-N-BEATS	39.40	**57.49**	**5.61**	76.23	113.09	5.59	114.09	180.43	5.51	155.18	230.59	**5.73**

Notes: For a fair comparison, the standard DE model also adopts the early stopping strategy as described in Subsection III-C6.

all prediction tasks are produced at the intervals [0.0001, 0.1], [0, 1] and [1, 10], respectively. Table VI lists the bounds of each hyperparameter.

For the rest parameters of ARDE-N-BEATS, we set $P^0 = 10,000$, $P^1 = 20$, $F_L = \{0.8, 0.6, 0.4, 0.2, 0, 0, 0, 0, 0, 0\}$, $F_U = \{1, 0.8, 0.6, 0.4, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2\}$, $C_r = 0.7$, $\max G = 10$, $k = 3$, $L^0_{\text{margin}} = 0.003$, and $f_{\text{margin}} = 0.8$ for all three data sets. The training epochs are all set as 150, and the best model with the minimum loss on the validation data is returned. Meanwhile, the parameter of ϵ is set as 0.07 for the M50-N and M1-N data sets, and 0.06 for the I280-S data set, which is determined with understanding and trials from the training and validation data.

4) Other Settings: The mean squared error (MSE) is employed as the loss function of N-BEATS for updating weights. All the models are implemented with Python and TensorFlow, and all the experiments were conducted on the server with Intel Xeon Gold 6240 CPU @ 2.60 GHz. The data and code are publicly available in GitHub repositories.[3]

D. Results and Comparisons

This section provides a comprehensive comparison and analyzes the findings of ARDE-N-BEATS in the following two perspectives: 1) comparison to other hyperparameter tuning methods and 2) comparison to existing baseline methods in traffic flow or time-series forecasting.

Table VII reveals the comparison of prediction errors on the test data between the N-BEATS with ARDE (i.e., ARDE-N-BEATS) and N-BEATS with other popular hyperparameter optimization approaches, including RSS over a fixed set (i.e., RSS-N-BEATS) [1], Bayesian optimization (BO) (i.e., BO-N-BEATS), ant colony optimization (i.e., ACO-N-BEATS), particle swarm optimization (i.e., PSO-N-BEATS), and standard DE optimization (i.e., DE-N-BEATS). All the five benchmarking methods are searched under the same data space as ARDE-N-BEATS. RSS-N-BEATS scans the hyperparameters of SL, BPS, SN, HU, TD, BS, LR, LDF, and LDP with steps of 1, 1, 1, 5, 1, 5, 0.0005, 0.1 and 1, respectively. For BO-N-BEATS and ACO-N-BEATS, the initial populations are both set to 30. The inertial weight (w), cognitive factor (c_1), social factor (c_2), and swarm size in PSO-N-BEATS are set to 0.5, 0.5, 0.9, and 60, respectively. For DE-N-BEATS, the mutation factors are generated using the interval [0.5, 1], the population is set as 30 as well, and the remaining parameters are the same as ARDE-N-BEATS, such as the crossover factor (C_r), $maxG$, k, and so on. Overall, the N-BEATS with BO, ACO, PSO, DE and ARDE have a lower prediction error than RSS-N-BEATS, indicating that they have better searching capabilities than RSS. Overall, ARDE-N-BEATS can achieve better or very competitive performance than the benchmarking approaches, since it defeats others regarding 21 of the 36 evaluation metrics (i.e., 3 metrics × 3 data sets × 4 tasks). DE-N-BEATS and ACO-N-BEATS, outperforming the rest in terms of six and five metrics, respectively, are the second-best and third-best performing approaches.

ARDE-N-BEATS, on the other hand, has a significantly lower computational cost than the benchmarking optimization approaches, as depicted in Fig. 4. Such considerable enhancement of computational efficiency is expected because of the designed acceleration mechanism in the reinitialization and

[3]https://github.com/Xiaocai-Zhang/ARDE-N-BEATS

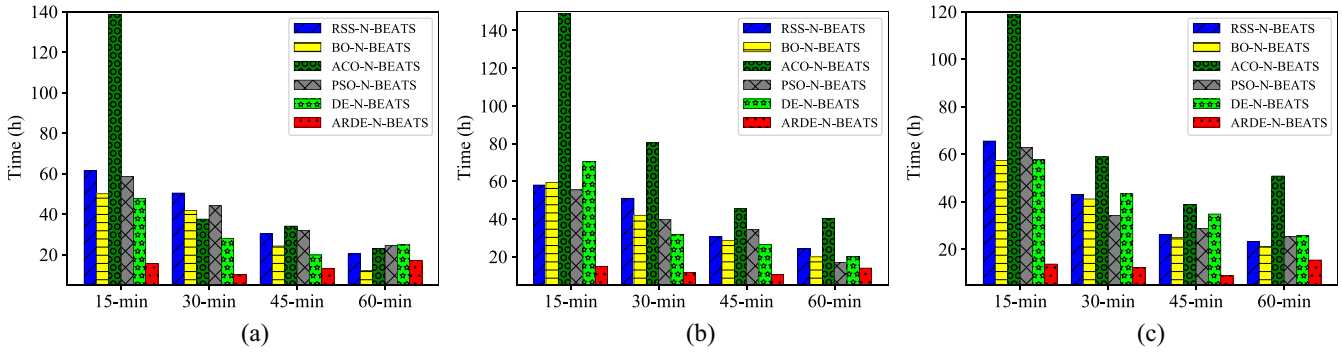


Fig. 4. Computational time comparison between different hyperparameters optimization methods. ARDE-N-BEATS takes the least computational cost for most of the prediction tasks with the exception of one task (i.e., M50-N 60-min task). However, BO-N-BEATS spends the least time on that task. One reason is probably that the averaged BS during the optimization process of BO is much larger for the M50-N data set (293.2 for M50-N versus 257.2 for M1-N versus 260.8 for I280-S). (a) M50-N. (b) M1-N. (c) I280-S.

TABLE VIII
PERFORMANCE COMPARISON BETWEEN ARDE-N-BEATS AND EXISTING BASELINES

Data	Model	15-min			30-min			45-min			60-min		
		MAE	RMSE	MAPE (%)	MAE	RMSE	MAPE (%)	MAE	RMSE	MAPE (%)	MAE	RMSE	MAPE (%)
M50-N	BPNN	28.43	41.27	6.53	79.43	115.61	8.77	134.38	194.14	11.24	250.09	352.04	13.99
	DBN[1]	29.67	43.03	6.47	73.18	108.36	7.61	118.53	174.06	9.52	133.58	195.01	8.21
	SAE[7]	25.07	38.13	5.34	55.10	85.08	5.94	77.99	121.53	6.17	130.62	212.33	7.82
	LSTM[3]	25.38	38.43	5.43	53.33	83.11	5.72	78.67	123.51	6.42	115.75	172.78	7.37
	LSTM_BILSTM[29]	28.73	40.30	8.37	66.57	95.24	9.89	105.37	148.97	11.16	131.00	189.03	9.41
	GRU	25.42	38.24	5.58	53.17	82.54	5.97	78.90	124.73	6.51	112.54	170.13	7.57
	CNN[48]	34.57	46.56	10.17	61.62	93.06	7.18	113.83	175.50	8.39	177.83	242.06	11.93
	TCN[35]	29.14	42.37	6.28	76.06	111.68	7.73	135.88	196.63	9.14	201.97	290.02	10.78
	seq2seq-LSTM[30]	23.42	34.66	5.65	52.54	79.18	6.40	78.38	122.46	6.40	112.36	172.97	6.62
	ARDE-N-BEATS	**21.86**	**33.30**	**5.04**	**47.45**	**73.49**	**5.20**	**67.82**	**109.58**	**4.85**	**96.77**	**157.11**	**5.11**
M1-N	BPNN	36.16	51.66	7.68	69.95	103.47	8.20	101.64	147.38	8.22	145.45	201.51	9.56
	DBN[1]	37.07	52.74	8.02	71.37	107.78	7.56	100.93	151.11	7.88	133.00	185.84	8.19
	SAE[7]	32.40	47.78	6.94	62.05	98.29	6.78	85.66	142.32	6.83	124.05	176.45	7.69
	LSTM[3]	32.79	48.18	7.08	62.45	110.34	6.80	82.89	132.31	6.52	118.12	170.34	6.99
	LSTM_BILSTM[29]	34.13	49.37	8.48	67.73	99.06	8.53	106.34	154.18	9.19	126.10	177.28	8.14
	GRU	33.10	48.95	7.05	64.45	110.57	6.98	88.13	136.03	6.84	114.80	166.06	7.03
	CNN[48]	32.22	48.98	6.77	76.37	103.96	12.14	94.34	136.40	8.35	120.94	166.58	8.21
	TCN[35]	37.11	53.28	7.94	79.07	118.78	8.24	122.58	183.15	9.07	171.82	242.42	11.13
	seq2seq-LSTM[30]	31.30	46.49	6.78	57.26	87.46	6.18	86.56	135.43	6.72	110.92	165.71	6.23
	ARDE-N-BEATS	**28.92**	**43.93**	**6.19**	**52.58**	**83.11**	**5.64**	**74.45**	**119.63**	**5.15**	**96.06**	**147.64**	**5.38**
I280-S	BPNN	43.69	62.99	6.21	95.69	133.46	7.48	143.74	204.69	7.80	199.06	270.27	8.95
	DBN[1]	44.28	63.82	6.36	98.46	145.85	6.75	156.53	236.04	7.26	210.61	298.37	8.10
	SAE[7]	43.50	63.60	6.08	89.75	131.51	6.26	135.83	209.51	6.58	177.31	259.03	6.62
	LSTM[3]	43.54	63.07	6.10	86.78	129.42	6.11	131.05	201.37	6.29	175.59	254.86	6.75
	LSTM_BILSTM[29]	44.44	62.63	7.03	85.27	120.73	6.67	139.34	200.11	7.43	185.68	262.74	7.45
	GRU	43.95	64.94	6.20	87.48	130.74	6.26	129.30	200.10	6.40	172.99	251.39	6.69
	CNN[48]	57.65	80.66	8.54	118.82	156.60	10.01	231.54	302.91	13.03	203.21	287.03	7.63
	TCN[35]	44.53	64.24	6.32	95.70	139.81	6.79	163.02	237.82	7.78	230.58	319.05	8.75
	seq2seq-LSTM[30]	42.37	64.82	6.03	80.25	118.16	5.82	130.56	190.58	6.36	174.70	259.35	6.35
	ARDE-N-BEATS	**39.40**	**57.49**	**5.61**	**76.23**	**113.09**	**5.59**	**114.09**	**180.43**	**5.51**	**155.18**	**230.59**	**5.73**

selection steps. The computational time of ARDE is not sensitive to the volume of training data, since it does not exhibit the trend as the other five algorithms (i.e., RSS, BO, ACO, PSO, and DE) that time increases as data volume enlarges. This is because we introduce the step of reinitialization, which requires a minimum computational cost for searching for significant populations. While the acceleration mechanism devised helps get rid of unnecessary computational costs when the data volume is large. ARDE-N-BEATS spends the least amount of time for optimization, with the only exception of the 60-min task on the M50-N data set that BO-N-BEATS takes the least time. The probable reason is that BO scans candidates with a much larger average BS for the M50-N data set (i.e., 293.2 for M50-N versus 257.2 for M1-N versus 260.8 for I280-S). Although ACO-N-BEATS obtains good performance in terms of above metrics, however, it spends the most computational expense in average. The computational superiority of ARDE over other benchmarking methods is more prominent in short-term traffic flow prediction (e.g., 15 and 30-min prediction tasks). When compared to the standard DE optimization, our ARDE algorithm saves the computational time with the highest at 78.90% (i.e., 14.89 h versus 70.56 h for the M1-N 15-min prediction task), and the lowest at 30.05% (i.e., 13.92 h versus 19.90 h for the M1-N 60-min task). Such improvements are dramatic and encouraging. Furthermore, the prediction errors suggest that

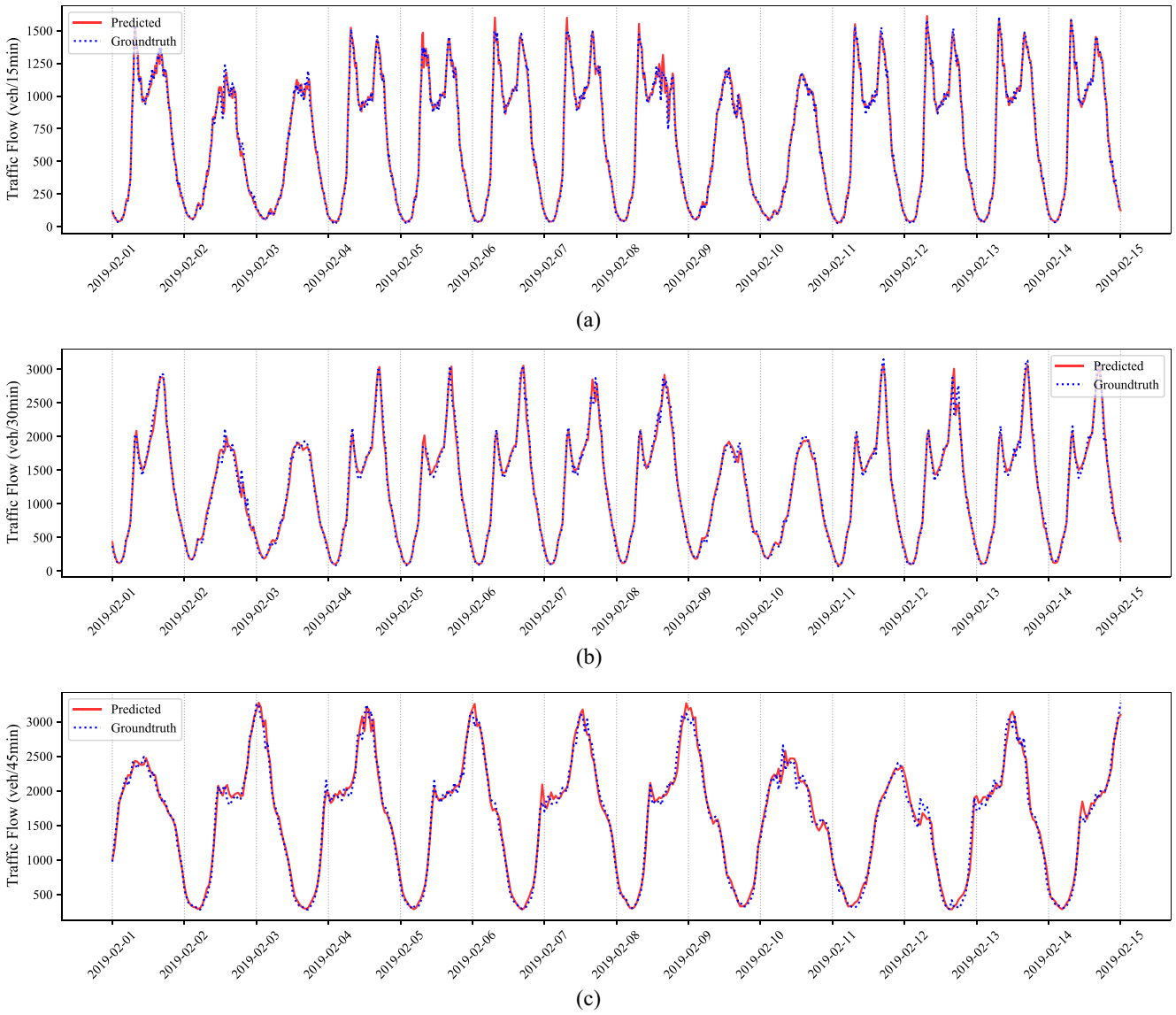


Fig. 5. Comparison between the predicted traffic flow and the ground-truth traffic flow on three selected tasks during a period of two weeks. The results obtained by our method are depicted by red solid lines. The ground-truth flow is plotted by blue dot lines. The trends of the predicted flow by ARDE-N-BEATS closely match with the ground-truth flow. (a) M50-N (15 min). (b) M1-N (30 min). (c) I280-S (45 min).

ARDE has a more robust and reliable searching ability which could aid deep learning models in realizing their extraordinary representation learning potential.

The nine baseline machine learning algorithms in traffic flow or time-series modeling, i.e., BPNN, DBN [1], SAE [7], LSTM [3], LSTM_BILSTM [29], GRU, CNN [48], TCN [35], and seq2seq-LSTM [30], are chosen as benchmarking methods for performance evaluation. Eight of them are utilizing deep architecture, as deep learning has been shown to outperform standard machine learning and statistical methods in related work [3], [7], [35], which will be compared comprehensively with different state-of-the-art architectures in this study. The forecasting results for ARDE-N-BEATS as well as benchmarking methods are exhibited in Table VIII. BPNN is a conventional NN with a simple fully connected structure. Both the DBN and the SAE employ a greedy layerwise unsupervised pretraining and fine-tuning for network training, showing remarkable improvements in studies [1] and [7]. LSTM is a typical RNN architecture, which is specially constructed to process the sequential data in time order. Furthermore, LSTM_BILSTM is an improved LSTM architecture combining the structures of vanilla LSTM and bidirectional LSTM, which has exhibited enhanced stability and reliability for longer time-series data modeling [29]. GRU is a variation of the LSTM with fewer trainable parameters and faster convergence speed. CNN is notable for its advantages that it does not require heavy preprocessing and has a very strong feature mapping ability [48]. TCN enables exponential expansion of the receptive field without loss of coverage, and is able to capture long-range patterns from sequential data, which has also demonstrated high accuracy in short-term traffic flow prediction [35]. Seq2seq-LSTM incorporates frameworks of seq2seq structure and LSTM module, indicating a significant improvement in real-world traffic flow prediction in study [30].

The mean accuracy (1-MAPE) of ARDE-N-BEATS exceeds 94% for all prediction tasks except the 15-min prediction on the M1-N data set (i.e., 93.81%). These encouraging findings imply that the forecasting accuracy obtained by ARDE-N-BEATS is high, stable and robust. The significant improvements indicate the strong nonlinear mapping capability of ARDE-N-BEATS as well. For all three data sets, ARDE-N-BEATS outperforms all the baselines with regard to all the three evaluation indices. Furthermore, baselines, such as SAE and RNN-based methods (i.e., LSTM, GRU, and seq2seq-LSTM) exhibit a very competitive accuracy to ARDE-N-BEATS. SAE can extract deeply hidden features with the assistance of layerwise pretraining and fine-tuning mechanisms introduced [49]. Meanwhile, RNN-based deep architectures are capable of learning the long-term interdependencies that lie behind the sequential data [50]. When compared to the most four competitive baselines, including SAE, LSTM, GRU, and seq2seq-LSTM, ARDE-N-BEATS improves the average accuracy of all prediction tasks by 1.17%, 1.05%, 1.17%, and 0.88%, respectively. DBN performs a bit worse than SAE, and the LSTM_BILSTM architecture cannot outperform the individual LSTM or GRU structure. The traditional BPNN can get a good accuracy for short-term prediction (i.e., 15-min prediction), however, its MAPE becomes larger as the prediction horizon increases. The prediction performance from both CNN and TCN is not robust and stable, as they obtain poor predictions for some tasks. The predicted curve and the ground-truth curve over a two-week period are depicted in Fig. 5, demonstrating that the predicted curves by ARDE-N-BEATS are well matched to the real curves without any explicit noise.

V. Conclusion

This article proposed a deep learning framework for traffic flow prediction. The core concept was to use an improved DE method to determine the globally optimized hyperparameters for an N-BEATS network. This hyperparameter optimization approach is advanced to the grid search or the random search, which has been frequently adopted by deep learning models in traffic flow prediction. Our method was named ARDE-N-BEATS (ARDE-based N-BEATS). The proposed ARDE-N-BEATS model was evaluated on three real-world data sets to confirm its superior prediction performance in comparison with the baseline methods, including BPNN, DBN, SAE, LSTM, LSTM_BILSTM, GRU, CNN, TCN, and seq2seq-LSTM. For practically all of the prediction tasks, ARDE-N-BEATS has attained at least 94% accuracy. This result is expected, as an N-BEATS-based model with optimized hyperparameters is exhibited with stronger nonlinear mapping ability than other deep learning architectures. A comprehensive comparison between the proposed ARDE algorithm and other hyperparameters optimization methods (i.e., RSS, BO, ACO, PSO, and DE) also confirmed the advantages of ARDE in terms of forecasting accuracy as well as computational efficiency.

For satisfying the need for rapid deployment of ITS and IoT, researchers are still working on developing an accurate and dependable approach for urban traffic flow prediction. In future work, we will refine our model by investigating additional features, such as day type as well as weather conditions, which might have a significant impact on traffic flow characteristics.

References

[1] W. Huang, G. Song, H. Hong, and K. Xie, "Deep architecture for traffic flow prediction: Deep belief networks with multitask learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 5, pp. 2191–2201, Oct. 2014.

[2] X. Zhang, "Deep learning for traffic time series data analysis," Ph.D. dissertation, Dept. Faculty Eng. Inf. Technol., Univ. Technol. Sydney, Sydney, NSW, Australia, 2021.

[3] Y. Tian and L. Pan, "Predicting short-term traffic flow by long short-term memory recurrent neural network," in *Proc. IEEE Int. Conf. Smart City/SocialCom/SustainCom (SmartCity)*, 2015, pp. 153–158.

[4] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, and C. Chen, "Data-driven intelligent transportation systems: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1624–1639, Dec. 2011.

[5] L. Xiao, D. Jiang, D. Xu, W. Su, N. An, and D. Wang, "Secure mobile crowdsensing based on deep learning," *China Commun.*, vol. 15, no. 10, pp. 1–11, Oct. 2018.

[6] L. Qu, W. Li, W. Li, D. Ma, and Y. Wang, "Daily long-term traffic flow forecasting based on a deep neural network," *Expert Syst. Appl.*, vol. 121, pp. 304–312, May 2019.

[7] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: A deep learning approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 865–873, Apr. 2015.

[8] T. N. Duc, C. T. Minh, T. P. Xuan, and E. Kamioka, "Convolutional neural networks for continuous QoE prediction in video streaming services," *IEEE Access*, vol. 8, pp. 116268–116278, 2020.

[9] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, Nov. 2020.

[10] J. Tang, F. Liu, Y. Zou, W. Zhang, and Y. Wang, "An improved fuzzy neural network for traffic speed prediction considering periodic characteristic," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 9, pp. 2340–2350, Sep. 2017.

[11] X. Zhang, Z. Zhao, Y. Zheng, and J. Li, "Prediction of taxi destinations using a novel data embedding method and ensemble learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 1, pp. 68–78, Jan. 2020.

[12] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Feb. 2012.

[13] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting," in *Proc. 10th Int. Conf. Learn. Represent. (ICLR)*, 2020, pp. 1–31.

[14] B. N. Oreshkin, G. Dudek, P. Pełka, and E. Turkina, "N-BEATS neural network for mid-term electricity load forecasting," *Appl. Energy*, vol. 293, Jul. 2021, Art. no. 116918.

[15] M. S. Ahmed and A. R. Cook, "Analysis of freeway traffic time-series data by using box-Jenkins techniques," *Transp. Res. Rec.*, no. 722, pp. 1–9, 1979.

[16] S. V. Kumar and L. Vanajakshi, "Short-term traffic flow prediction using seasonal ARIMA model with limited input data," *Eur. Transp. Res. Rev.*, vol. 7, no. 3, pp. 1–9, 2015.

[17] Q. Y. Ding, X. F. Wang, X. Y. Zhang, and Z. Q. Sun, "Forecasting traffic volume with space-time ARIMA model," *Adv. Mater. Res.*, vols. 156–157, pp. 979–983, Jan. 2011.

[18] S. Lee and D. B. Fambro, "Application of subset autoregressive integrated moving average model for short-term freeway traffic volume forecasting," *Transp. Res. Rec.*, vol. 1678, no. 1, pp. 179–188, 1999.

[19] W. Min and L. Wynter, "Real-time road traffic prediction with spatio-temporal correlations," *Transp. Res. C, Emerg. Technol.*, vol. 19, no. 4, pp. 606–616, 2011.

[20] M.-C. Tan, S. C. Wong, J.-M. Xu, Z.-R. Guan, and P. Zhang, "An aggregation approach to short-term traffic flow prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 1, pp. 60–69, Mar. 2009.

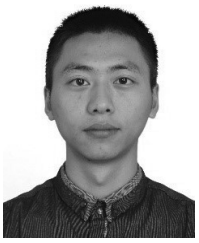
[21] J. Guo, W. Huang, and B. M. Williams, "Adaptive Kalman filter approach for stochastic short-term traffic flow rate prediction and uncertainty quantification," *Transp. Res. C, Emerg. Technol.*, vol. 43, no. 1, pp. 50–64, Jun. 2014.

[22] J. Wang, Q. Shi, and H. Lu, "The study of short-term traffic flow forecasting based on theory of chaos," in *Proc. IEEE Intell. Veh. Symp.*, 2005, pp. 869–874.

[23] G. A. Davis and N. L. Nihan, "Nonparametric regression and short-term freeway traffic forecasting," *J. Transp. Eng.*, vol. 117, no. 2, pp. 178–188, 1991.
[24] M. Castro-Neto, Y.-S. Jeong, M.-K. Jeong, and L. D. Han, "Online-SVR for short-term traffic flow prediction under typical and atypical traffic conditions," *Expert Syst. Appl.*, vol. 36, no. 3, pp. 6164–6173, 2009.
[25] S. Sun, C. Zhang, and G. Yu, "A Bayesian network approach to traffic flow forecasting," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 1, pp. 124–132, Mar. 2006.
[26] J. Wang, W. Deng, and Y. Guo, "New Bayesian combination method for short-term traffic flow forecasting," *Transp. Res. C, Emerg. Technol.*, vol. 43, no. 1, pp. 79–94, 2014.
[27] H.-F. Yang, T. S. Dillon, and Y.-P. P. Chen, "Optimized structure of the traffic flow forecasting model with a deep learning approach," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2371–2381, Oct. 2017.
[28] X. Luo, D. Li, Y. Yang, and S. Zhang, "Spatiotemporal traffic flow prediction with KNN and LSTM," *J. Adv. Transp.*, vol. 2019, pp. 1–11, Feb. 2019.
[29] C. Ma, G. Dai, and J. Zhou, "Short-term traffic flow prediction for urban road sections based on time series analysis and LSTM_BILSTM method," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 5615–5624, Jun. 2022.
[30] Z. Wang, X. Su, and Z. Ding, "Long-term traffic prediction based on LSTM encoder–decoder architecture," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 10, pp. 6561–6571, Oct. 2021.
[31] A.-S. Mihaita, H. Li, Z. He, and M.-A. Rizoiu, "Motorway traffic flow prediction using advanced deep learning," in *Proc. IEEE 22nd Int. Conf. Intell. Transp. Syst. (ITSC)*, 2019, pp. 1683–1690.
[32] Y. Wu and H. Tan, "Short-term traffic flow forecasting with spatial-temporal correlation in a hybrid deep learning framework," 2016, *arXiv:1612.01022*.
[33] Y. Tian, K. Zhang, J. Li, X. Lin, and B. Yang, "LSTM-based traffic flow prediction with missing data," *Neurocomputing*, vol. 318, pp. 297–305, Nov. 2018.
[34] B. Yang, S. Sun, J. Li, X. Lin, and Y. Tian, "Traffic flow prediction using LSTM with feature enhancement," *Neurocomputing*, vol. 332, pp. 320–327, Mar. 2019.
[35] W. Zhao, Y. Gao, T. Ji, X. Wan, F. Ye, and G. Bai, "Deep temporal convolutional networks for short-term traffic flow forecasting," *IEEE Access*, vol. 7, pp. 114496–114507, 2019.
[36] Y. Liu, J. Q. James, J. Kang, D. Niyato, and S. Zhang, "Privacy-preserving traffic flow prediction: A federated learning approach," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7751–7763, Aug. 2020.
[37] B. Yan, G. Wang, J. Yu, X. Jin, and H. Zhang, "Spatial-temporal Chebyshev graph neural network for traffic flow prediction in IoT-based its," *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9266–9279, Jun. 2022.
[38] X. Zhang, C. Huang, Y. Xu, and L. Xia, "Spatial-temporal convolutional graph attention networks for citywide traffic flow forecasting," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manag.*, 2020, pp. 1853–1862.
[39] T. Qi, G. Li, L. Chen, and Y. Xue, "ADGCN: An asynchronous dilation graph convolutional network for traffic flow prediction," *IEEE Internet Things J.*, vol. 9, no. 5, pp. 4001–4014, Mar. 2022.
[40] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *Proc. 6th Int. Conf. Learn. Represent. (ICLR)*, 2018, pp. 1–16.
[41] X. Ye, S. Fang, F. Sun, C. Zhang, and S. Xiang, "Meta graph transformer: A novel framework for spatial–temporal traffic prediction," *Neurocomputing*, vol. 491, pp. 544–563, Jun. 2022.
[42] H.-X. Li, J.-L. Yang, G. Zhang, and B. Fan, "Probabilistic support vector machines for classification of noise affected data," *Inf. Sci.*, vol. 221, pp. 60–71, Feb. 2013.
[43] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *J. Inf. Security Appl.*, vol. 50, Feb. 2020, Art. no. 102419.
[44] X. Zhang, X. Fu, Z. Xiao, H. Xu, and Z. Qin, "Vessel trajectory prediction in maritime transportation: Current approaches and beyond," *IEEE Trans. Intell. Transp. Syst.*, early access, Jul. 27, 2022, doi: 10.1109/TITS.2022.3192574.
[45] R. Ketabi, M. Al-Qathrady, B. Alipour, and A. Helmy, "Vehicular traffic density forecasting through the eyes of traffic cameras; a spatio-temporal machine learning study," in *Proc. 9th ACM Symp. Design Anal Intell. Veh. Netw. Appl.*, 2019, pp. 81–88.
[46] J. M. G. Sopeña, V. Pakrashi, and B. Ghosh, "Interval prediction for short-term traffic forecasting using hybrid mode decomposition models," in *Proc. IEEE Int. Intell. Transp. Syst. Conf. (ITSC)*, 2021, pp. 3246–3251.
[47] L. Peng, S. Liu, R. Liu, and L. Wang, "Effective long short-term memory with differential evolution algorithm for electricity price prediction," *Energy*, vol. 162, pp. 1301–1314, Nov. 2018.
[48] Y. Geng and X. Luo, "Cost-sensitive convolutional neural networks for imbalanced time series classification," *Intell. Data Anal.*, vol. 23, no. 2, pp. 357–370, 2019.
[49] X. Zhang, Y. Zheng, Z. Zhao, Y. Liu, M. Blumenstein, and J. Li, "Deep learning detection of anomalous patterns from bus trajectories for traffic insight analysis," *Knowl.-Based Syst.*, vol. 217, Apr. 2021, Art. no. 106833.
[50] J. Liu, A. Shahroudy, D. Xu, A. C. Kot, and G. Wang, "Skeleton-based action recognition using spatio-temporal LSTM network with trust gates," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 3007–3021, Dec. 2018.

Xiaocai Zhang received the B.S. degree in navigation technology and the M.S. degree in traffic and transportation engineering from Dalian Maritime University, Dalian, China, in 2013 and 2016, respectively, and the Ph.D. degree in computer science from the University of Technology Sydney, Ultimo, NSW, Australia, in 2021.

He is currently a Research Scientist with the Institute of High Performance Computing, A*STAR, Singapore. He was a Research Assistant with the Data Science Institute, University of Technology Sydney. His research interests include data mining, deep learning, urban computing, time-series analysis, and intelligent transportation systems.



Zhixun Zhao received the B.E. degree in electronic science and technology from the University of Electronic Science and Technology of China, Chengdu, China, in 2013, the M.E. degree in microelectronics technology from the National University of Defense Technology, Changsha, China, in 2015, and the Ph.D. degree in computer science from the University of Technology Sydney, Sydney, NSW, Australia, in 2020.

He is currently working in Data Analytics and Cloud Computing with the National Key Laboratory of Science and Technology on Blind Signal Processing, Chengdu. His research interests include data mining, machine learning, and cloud computing.

Jinyan Li received the bachelor of science degree in applied mathematics from the National University of Defense Technology, Changsha, China, in 1991, the master of engineering degree in computer engineering from Hebei University of Technology, Tianjin, China, in 1994, and the Ph.D. degree in computer science from the University of Melbourne, Parkville, VIC, Australia, in 2001.

He is currently a Professor of Data Science with the Data Science Institute and a Core Member of the Centre for Health Technologies, Faculty of Engineering and Information Technology, University of Technology Sydney, Ultimo, NSW, Australia. He is also the Bioinformatics Program Leader. He joined the University of Technology Sydney in 2011 after ten years of research and teaching work with the Institute for Infocomm Research, Nanyang Technological University, Singapore, and National University of Singapore, Singapore. He is widely known for his pioneering and theoretical research work on Emerging Patterns that has spawned numerous follow-up research interests in data mining, machine learning, and bioinformatics and made an enduring contribution to these fields.