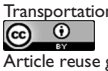
TRR
JOURNAL OF THE TRANSPORTATION RESEARCH BOARD

Research Article

Transportation Research Record
2021, Vol. 2675(10) 211–227


Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/03611981211011170
journals.sagepub.com/home/trr
SAGE

DistTune: Distributed Fine-Grained Adaptive Traffic Speed Prediction for Growing Transportation Networks

Ming-Chang Lee[1], Jia-Chun Lin[1], and Ernst Gunnar Gran[1,2]

Abstract

Over the past decade, many approaches have been introduced for traffic speed prediction. However, providing fine-grained, accurate, time-efficient, and adaptive traffic speed prediction for a growing transportation network where the size of the network keeps increasing and new traffic detectors are constantly deployed has not been well studied. To address this issue, this paper presents DistTune based on long short-term memory (LSTM) and the Nelder-Mead method. When encountering an unprocessed detector, DistTune decides if it should customize an LSTM model for this detector by comparing the detector with other processed detectors in the normalized traffic speed patterns they have observed. If a similarity is found, DistTune directly shares an existing LSTM model with this detector to achieve time-efficient processing. Otherwise, DistTune customizes an LSTM model for the detector to achieve fine-grained prediction. To make DistTune even more time-efficient, DisTune performs on a cluster of computing nodes in parallel. To achieve adaptive traffic speed prediction, DistTune also provides LSTM re-customization for detectors that suffer from unsatisfactory prediction accuracy due to, for instance, changes in traffic speed patterns. Extensive experiments based on traffic data collected from freeway I5-N in California are conducted to evaluate the performance of DistTune. The results demonstrate that DistTune provides fine-grained, accurate, time-efficient, and adaptive traffic speed prediction for a growing transportation network.

Traffic speed is a key indicator for measuring the efficiency of a transportation network. Accurate traffic speed prediction is therefore crucial to achieve proactive traffic management and control for transportation networks. During the past decade, many approaches and methods have been introduced for traffic speed prediction. Each of them can be summarized as learning a mapping function between input variables and output variables. These methods can be classified into two main types: parametric and nonparametric. Parametric approaches simplify the mapping function to a known form, i.e., they require a predefined model. A typical example is the autoregressive integrated moving average approach (ARIMA) (*1*). By contrast, nonparametric approaches do not require a predefined model structure. Typical examples include the k-nearest neighbors (k-NN) method (*2*, *3*), artificial neural network (ANN) (*4*), and recurrent neural network (RNN) (*5*).

However, providing fine-grained, accurate, time-efficient, and adaptive traffic speed prediction for a growing transportation network where the size of the network keeps increasing and new traffic detectors are constantly deployed on the network has not been well studied. To address this issue, this paper proposes a solution based on long short-term memory (LSTM) (*6*), which is a special type of RNN. Prior studies such as Ma et al. (*7*), Yu et al. (*8*), and Zhao et al. (*9*) have shown that LSTM is superior in time series prediction and provides better prediction accuracy than many existing approaches and neural networks, including Elman NN (*7*), Time-delayed NN (*7*), Nonlinear Autoregressive NN (*7*), support vector machine (*7*), ARIMA (*1*), and the Kalman Filter approach (*10*). Therefore, LSTM is chosen as our building block.

However, several challenges exist and several issues must be addressed to achieve the above-mentioned goal (i.e., fine-grained, accurate, time-efficient, and adaptive traffic speed prediction for a growing transportation network). For instance, detectors such as loop sensors and

[1]Department of Information Security and Communication Technology, Norwegian University of Science and Technology, Gjøvik, Norway
[2]Simula Research Laboratory, Fornebu, Norway

Corresponding Author:
Jia-Chun Lin, jia-chun.lin@ntnu.no

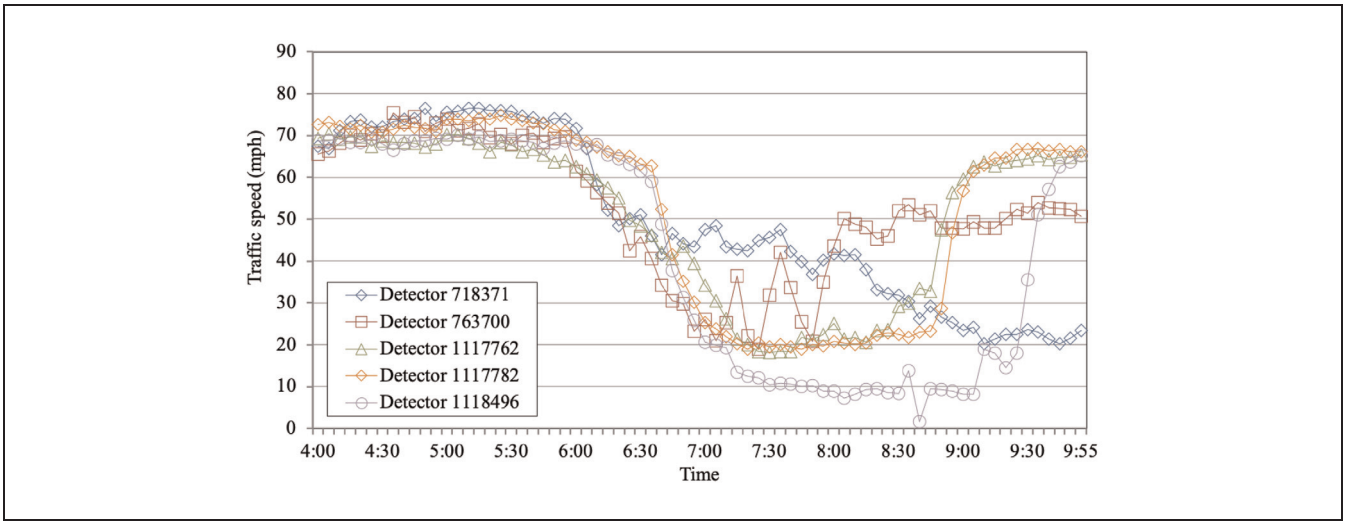


Figure 1. The traffic speed collected by five detectors on freeway I5-N in California between 4 a.m. and 10 a.m. in a typical weekday (*12*).

traffic cameras in a transportation network are deployed in different places to collect and monitor traffic data. Depending on the density of nearby population and other factors, the traffic speed observed/collected by detectors at different locations may not be the same. For example, Figure 1 illustrates the traffic speed observed by five detectors deployed on freeway I5-N in California (*11*) in a typical weekday. We can see that the observed traffic speeds were similar to each other between 4 a.m. and 6 a.m. However, the pattern of the traffic speed became different and diverse from 6 a.m. Therefore, we recommend that each detector should have its own LSTM model to predict the traffic speed of its coverage so as to provide fine-grained traffic speed prediction and achieve better transportation management and services. However, such an approach is expensive and impractical because training an LSTM model for each individual detector in the growing transportation network is required and that each training process is in general time-consuming.

Another issue is how to achieve satisfactory prediction accuracy and continuously maintain satisfactory accuracy over time for every single detector in the growing transportation network. It is well-known that the success of LSTM in achieving satisfactory prediction accuracy relies on an appropriately configured set of hyperparameters, which are parameters whose values are set before the training process of LSTM starts. These hyperparameters include the learning rate and the number of hidden layers. Determining appropriate values for LSTM hyperparameters is usually done manually by trial and error, which might be time-consuming, and may not be able to guarantee good prediction accuracy. In addition, an LSTM model might not be able to keep offering satisfactory prediction accuracy because the traffic speed pattern observed by the corresponding detector may change over time.

To summarize, this paper attempts to address the following challenges:

1. How can we automatically customize an LSTM for each single detector (i.e., appropriately configuring LSTM hyperparameters) in a growing transportation network such that the corresponding LSTM model provides satisfactory prediction accuracy?
2. How can we time-efficiently perform automatic LSTM customization for the increasing number of detectors in growing transportation networks?
3. How can we keep maintaining satisfactory prediction accuracy for every single detector in a time-efficient way?

To address the above challenges, in this paper, we propose DistTune, which is a distributed scheme to automatically customize LSTM models and constantly provide satisfactory prediction accuracy for every single detector in a growing transportation network. DistTune customizes an LSTM model for a detector by automatically tuning LSTM hyperparameter values and training the corresponding LSTM models based on the Nelder-Mead method (NMM) (*13*), a commonly applied method used to find the minimum or maximum of an objective function in a multidimensional space. The reason why NMM is employed will be explained later in this paper.

However, simply using NMM to gradually customize an LSTM model for every single detector is not a scalable solution since the number of detectors in a growing transportation network might be very large and increasing. To enhance scalability and efficiency, DistTune

allows detectors to share the same LSTM model if these detectors have observed similar traffic patterns. In addition, DistTune is designed in an incremental, distributed, and parallel manner, and runs on a cluster of computing nodes to accelerate all required LSTM customization processes.

Whenever DistTune encounters an unprocessed detector D_i, it checks if the traffic speed pattern observed by D_i is similar to that observed by any other detector whose LSTM model has been customized by DistTune. If the answer is positive and the traffic speed pattern observed by D_i is similar to that observed by detector D_j, DistTune directly shares D_j's LSTM model with D_i without customizing an LSTM model for D_i. However, if the answer is negative, DistTune requests an available computing node from the cluster to customize an LSTM model for D_i. To guarantee and maintain satisfactory prediction accuracy, DistTune keeps track of the prediction accuracy of all detectors. If any LSTM model is unable to achieve the desired prediction accuracy because of a change in the traffic pattern or another reason, DistTune requests an available computing node to re-customize an LSTM model for the corresponding detector.

To demonstrate the performance of DistTune, we conducted three extensive experiments on an Apache Hadoop YARN cluster using the traffic data collected by detectors on freeway I5-N in California. In the first experiment, we designed several scenarios to evaluate the auto-tuning effectiveness of DistTune. The results show that auto-tuning LSTM hyperparameters leads to higher prediction accuracy than manually configuring LSTM hyperparameters. In the second experiment, we evaluated the auto-sharing LSTM performance of DistTune. The results indicate that DistTune significantly reduces the time for LSTM customization. In the last experiment, we evaluated the LSTM re-customization performance of DistTune under two types of re-customization. One is to start from scratch. The other one is based on Transfer Learning (*14*). Surprisingly, the results show that following the concept of Transfer Learning does not bring significant benefits if time consumption and prediction accuracy are both considered. The contributions of this paper are as follows:

1. Two well-known hyperparameter optimization approaches are evaluated and compared through empirical experiments: NMM and the Bayesian optimization approach (BOA) (*15*). The results suggest that the former method suits DistTune better because of its efficiency in finding appropriate LSTM hyperparameter values for achieving the desired prediction accuracy.
2. The proposed DistTune enables detectors to share the same LSTM model, and it provides an LSTM re-customization service for all detectors when any of them fails to provide satisfactory prediction accuracy. The design of DistTune addresses the time-consuming and computation-intensive issues of LSTM customization for the tremendous number of detectors in a growing transportation network.
3. The performance of DistTune is carefully evaluated through extensive experiments based on traffic data collected by detectors on freeway I5-N in California. The results suggest that DistTune provides fine-grained, accurate, time-efficient, and adaptive traffic speed prediction for a growing transportation network.

The rest of the paper is organized as follows: the next section provides background and related work. We then evaluate NMM and BOA, and explain why NMM is chosen by DistTune to autotune LSTM hyperparameters. We then introduce the details of DistTune and evaluate the performance of DistTune, respectively. This is followed by a discussion of the training data set we chose for conducting our experiments. The paper concludes with an outline of future work.

Background and Related Work

In this section, we briefly introduce LSTM and discuss related work.

LSTM and its Hyperparameters

LSTM is a special type of RNN. Its architecture is similar to that of RNN except that the nonlinear units in the hidden layers are memory blocks. Each memory block consists of memory cells, an input gate, a forget gate, and an output gate. The input gate decides if the input should be stored in the memory cells. The forget gate determines if current memory contents should be deleted. The output gate decides if current memory contents should be output. This design enables LSTM to preserve information over long time lags, thereby addressing the vanishing gradient problem (*16*).

According to Lee et al. (*12*), the prediction performance of LSTM greatly depends on configuring appropriate values for the following hyperparameters:

- Learning rate
- The number of hidden layers
- The number of hidden units
- Epochs

The learning rate controls the amount the weights of LSTM are updated during training. The lower the value,

the less is the possibility of missing any local minima, but it might prolong the training process. A hidden layer is where hidden units take in a set of weighted inputs and produce an output through an activation function. More hidden layers are usually required to learn a large and complex training data set. A hidden unit is a neuron in a hidden layer. An inappropriate number of hidden units might cause either overfitting or underfitting. An epoch is defined as one forward pass and one backward pass of all the training data. Too few epochs may underfit the training data, whereas too many epochs might overfit the training data.

It is clear that all the above-mentioned hyperparameters are important to the learning performance and computation efficiency of LSTM. Therefore, DistTune takes all these hyperparameters into consideration when conducting LSTM customization.

Related Work

Over the past two decades, many traffic prediction approaches have been proposed. As briefly mentioned in the introduction, they can be classified into two categories: parametric and nonparametric approaches. In parametric approaches, a model structure (i.e., the mapping function between input variables and output variables) needs to be determined beforehand based on some theoretical assumptions. Usually the model parameters can be derived from empirical data. The ARIMA model is a typical and widely used parametric approach (*17*). Many ARIMA-based approaches were also introduced to improve prediction accuracy, including Lee and Fambro (*18*), Williams (*19*), and Williams and Hoel (*20*).

Unlike parametric approaches, nonparametric approaches do not require a predefined model structure (i.e., there is no need to make assumptions about the mapping function). Typical examples include k-NN, ANN, RNN, hybrid approaches, and so forth. The k-NN method was used in Davis and Nihan (*2*) to forecast freeway traffic. Several variants of the k-NN and RNN methods were then introduced for traffic prediction, such as Bustillos and Chiu (*3*) and Cui et al. (*21*). Le et al. (*22*) addressed traffic speed prediction using big traffic data obtained from static sensors and proposed local Gaussian processes to learn and make predictions for correlated subsets of data. Jiang and Fei (*23*) introduced a data-driven vehicle speed prediction method based on Hidden Markov models. Ma et al. (*7*) employed LSTM to forecast traffic speed using remote microwave sensor data. However, all the above approaches focus on predicting traffic on a fix-length road section or a fix-sized region. Unlike these approaches, DistTune is a flexible scheme for a growing transportation network since it is able to incrementally customize LSTM models for any detectors that are newly deployed on a transportation network. This property makes DistTune a flexible and scalable solution for a growing transportation network.

Ma et al. (*24*) used deep learning theory to predict traffic congestion evolution in large-scale transportation networks. Furthermore, Ma et al. (*25*) predicted traffic speed in large-scale transportation networks by representing network traffic as images and employing convolutional neural networks to make predictions. However, both of these methods require the scale of the target transportation network to be fixed and specified in advance, which is not required when DistTune is employed. DistTune can handle an increasing number of detectors on the fly without requiring the scale of the target transportation networks to be specified beforehand.

More recently, Lee and Lin (*26*) formulated the problem of customizing an LSTM model into a finite Markov decision process and then introduced a distributed approach called DALC to automatically customize LSTM models for detectors in large-scale transportation networks. However, DALC only focuses on two LSTM hyperparameters (the number of hidden layers and epochs) using a fixed state transition graph. Unlike DALC, our DistTune considers two more LSTM hyperparameters without using a finite Markov decision process. DistTune is, therefore, more flexible than DALC. In 2020, Lee et al. (*12*) introduced DistPre to provide find-grained traffic speed prediction for large-scale transportation networks based on LSTM customization and distributed computing. However, the LSTM models generated by DistPre cannot be updated to adapt to traffic pattern changes. Therefore, DistPre might not be able to maintain satisfactory prediction accuracy for every single detector in a growing transportation network over time.

Evaluation of BOA and NMM

Hyperparameter optimization is the process of finding appropriate values for the hyperparameters of a training algorithm such that the algorithm can achieve desirable results. Existing approaches include NMM, the Bayesian optimization approach (BOA), grid search, random walk, random search, genetic algorithm, greedy search, simulated annealing, particle swarm optimization, etc. The evaluation of these approaches can be found in, for example, Matuszyk et al. (*27*) and Thornton et al. (*28*).

Among these approaches, BOA has gained great popularity in recent years in a wide range of areas as a result of its power and efficiency. BOA was designed to find the optimal value of a black-box objective function. In our context, the black-box objective function refers to an LSTM model, and the optimal value refers to a hyperparameter setting (i.e., assigning a value to each LSTM hyperparameter we consider). With this optimal

Table 1. Four Hyperparameters and their Domains

Hyperparameter	R_{Learn}	N_{Layer}	N_{Unit}	ep
Domain	[0.01, 0.2]	[1, 10]	[2, 40]	[100, 1000]
Discrete with step	0.01	1	2	20

Note: ep = number of epochs.

hyperparameter setting, the LSTM model is able to provide satisfactory prediction accuracy.

In BOA, the uncertainty of the objective function across not-yet evaluated values is modeled as a prior probability distribution (called prior), which captures our beliefs about the behavior of the function. After gathering the function evaluations (i.e., the prediction accuracy of an LSTM model under a particular hyperparameter setting), the prior is updated to form the posterior distribution over the objective function. The posterior distribution is then used to construct an acquisition function that will select the most promising value (i.e., the most promising hyperparameter setting) for next evaluation. The above process iterates toward an optimum. Examples of acquisition functions include, for instances, probability of improvement, expected improvement, and Bayesian expected losses (*29*). All of them try to use and balance exploration and exploitation to minimize the number of function queries. This is why BOA is suitable for functions that are expensive to evaluate. An in-depth review of BOA can be found in Brochu et al. (*30*).

NMM is another popular optimization method for nonlinear functions. This method does not require any derivative information, making it suitable for problems with non-smooth functions. NMM minimizes an objective function by generating an initial simplex based on a predefined vertex and then performing a function evaluation at each vertex of the simplex. Note that a simplex has $n + 1$ vertices in R^n where n is the number of dimensions of the parameter space. For instance, the simplex is a triangle when n is 2. In our context, a vertex is a set of values assigned to LSTM hyperparameters, the predefined vertex is a default LSTM hyperparameter setting, and the function evaluation is to derive the prediction error of an LSTM model trained with a certain data set under a hyperparameter setting.

A sequence of transformations is then performed iteratively on the simplex, aiming to decrease the function values at its vertices. Possible transformations include reflection, expansion, contraction, and shrinking. The above process terminates when the sample standard deviation of the function values of the current simplex fall below some tolerance. We refer readers to Singer and Nelder (*31*) for more details about these transformations.

In this paper, we focus on evaluating BOA and NMM to see which of them suits DistTune better, i.e., which of them can more time-efficiently find LSTM hyperparameters for detectors such that the resulting LSTM models are able to achieve the desired prediction accuracy. In this experiment, BOA and NMM focus on auto-tuning the four above-mentioned LSTM hyperparameters: learning rate (denoted by R_{Learn}), the number of hidden layers (denoted by N_{Layer}), the number of hidden units (denoted by N_{Unit}), and the number of epochs (denoted by ep). Table 1 presents the domains of these hyperparameters.

To fairly evaluate and compare BOA and NMM, we selected the five detectors shown in Figure 1 to be their targets. It is clear from Figure 1 that the traffic speed patterns observed by detectors 1117762 and 1117782 are similar to each other, and the traffic speed patterns observed by the other three detectors are diverse. These phenomena reflect the traffic speed patterns observed by detectors in a real transportation network. This is why we chose these five detectors to evaluate BOA and NMM. If one of these two approaches is able to more time-efficiently autotune LSTM hyperparameters for these five detectors than the other such that the corresponding models achieve the desired prediction accuracy, it is likely that this approach can offer better time efficiency than the other when it is adopted by DistTune in a growing transportation network.

In this evaluation, each of these five detectors has the same size of training data (i.e., traffic speed values from five working days with the collection interval of every 5 min). Two metrics were used for the comparison: average absolute relative error (AARE) and LSTM customization time (LCT). AARE is a well-known measure for the prediction accuracy of a forecast method (*32*), which is defined as follows:

$$\mathrm{AARE} = \frac{1}{W}\sum_{w=1}^{W}\frac{|s_w - \widehat{s_w}|}{s_w} \tag{1}$$

where

W is the total number of data points considered for comparison,

w is the index of data point,

s_w is the actual traffic speed value at w, and

$\widehat{s_w}$ is the forecast traffic speed value at w.

The lower AARE is, the higher prediction accuracy the forecast method has. The second metric (i.e., LCT) is the

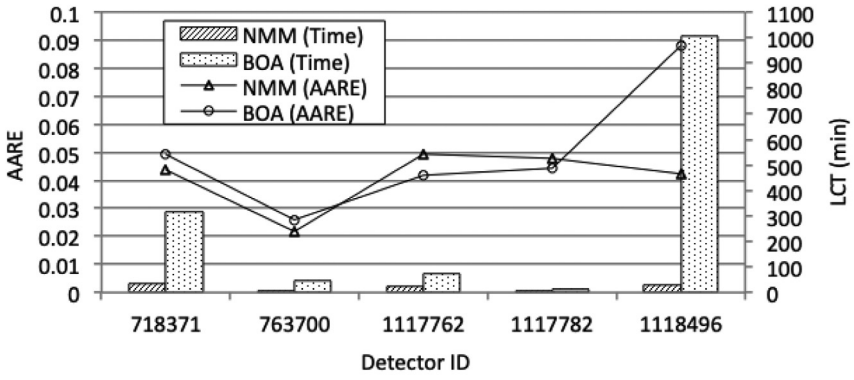


Figure 2. The performance of NMM and BOA on customizing LSTM models for five detectors that observe different traffic speed patterns.
Note: NMM = Nelder-Mead method; BOA = Bayesian optimization approach; AARE = average absolute relative error; LCT = LSTM customization time.

time taken by BOA and NMM to individually find an LSTM hyperparameter setting for a detector such that the prediction accuracy of the corresponding LSTM model satisfies a predefined AARE threshold, which is 0.05 in this paper. Note that this value is considered satisfactory according to Lee et al. (*12*), Lee and Lin (*26*), and Xia et al. (*33*).

In this experiment, both approaches start with the following default hyperparameter setting, taken from Lee et al. (*12*):

$$<R_{Learn} = 0.01, N_{Layer} = 1, N_{Unit} = 2, ep = 100>.$$

As soon as they find a hyperparameter setting with which the corresponding LSTM model has an AARE less than or equal to the AARE threshold, they automatically terminate. However, if they have iterated 20 times without being able to find such a hyperparameter setting, they will be forcibly terminated. This experiment was performed on a laptop running MacOS 10.13.1 with 2.5 GHz Quad-Core Intel Core i7 and 16 GB 1,600 MHz DDR3.

Figure 2 illustrates the performance of BOA and NMM in terms of AARE (presented as marked lines) and LCT (presented as clustered columns). We can see that all the LSTMs customized by NMM satisfy the AARE threshold because their AARE values are all lower than 0.05. However, not all the LSTMs customized by BOA achieve the same good result, especially for detector 1118496. After BOA iterates 20 times, the best AARE value for this detector is still higher than 0.05, and the LCT taken by BOA has reached 1,006 min.

Based on the above results, we conclude that NMM is more suitable than BOA for DistTune because NMM has better efficiency and effectiveness in finding appropriate values for LSTM hyperparameters such that the desired prediction accuracy for detectors can be satisfied. Even though BOA is able to find optimal hyperparameter settings that might be able to reach higher prediction accuracy than NMM, its high time consumption is unaffordable for DistTune. This is why NMM is adopted by DistTune to autotune LSTM hyperparameters.

Methodology of DistTune

In this section, we firstly introduce all algorithms of DistTune and then suggest a customization protocol for practitioners to implement DistTune on their growing transportation networks.

Algorithms. DistTune is designed and implemented as an incremental LSTM auto-tuning, sharing, and re-customization system on a cluster consisting of a master server and a set of worker nodes. The master server determines whether to customize, share, or re-customize LSTM models for individual detectors. Each worker node waits for an instruction from the master server to conduct LSTM customization/re-customization for a given detector.

Figure 3 illustrates the LSTM auto-tuning and sharing algorithm running on the master server, and Figure 4 shows the high-level flowchart of the algorithm. Let $G = D_1, D_2, \ldots, D_x$ be a list of detectors that have their own LSTM models customized by DistTune. Note that G is empty before DistTune is employed and launched. Whenever encountering an unprocessed detector (denoted by U_i), the algorithm first normalizes a list of

Input: An unprocessed detector U_i
Output: A decision to share an existing LSTM model with U_i or to customize an LSTM model for U_i
Procedure:

```
1   Let f be a boolean variable and let f be false;
2   Let G = {D1, D2, ..., Dx} be a list of detectors having their
3   own customized LSTM models;
4   Let Li = {vi,1, vi,2, ..., vi,T} be a list of traffic-speed values
5   previously collected by Ui;
6   Normalize Li into Nori by dividing each value in Li by V;
7   for j = 1 to x{  // x is the total number of detectors in G;
8     Calculate AARDi,j based on Equation 2;
9     if AARDi,j < thdAARD {
10      Share Dj's LSTM model with Ui;
11      Let f be true;
12      break;}}
13  if f==false {
14    Ask a worker node to customize an LSTM model for Ui;
15    Append Ui to the end of G;}
```

Figure 3. The LSTM auto-tuning and sharing algorithm performed by the master server.
Note: LSTM = long short-term memory.

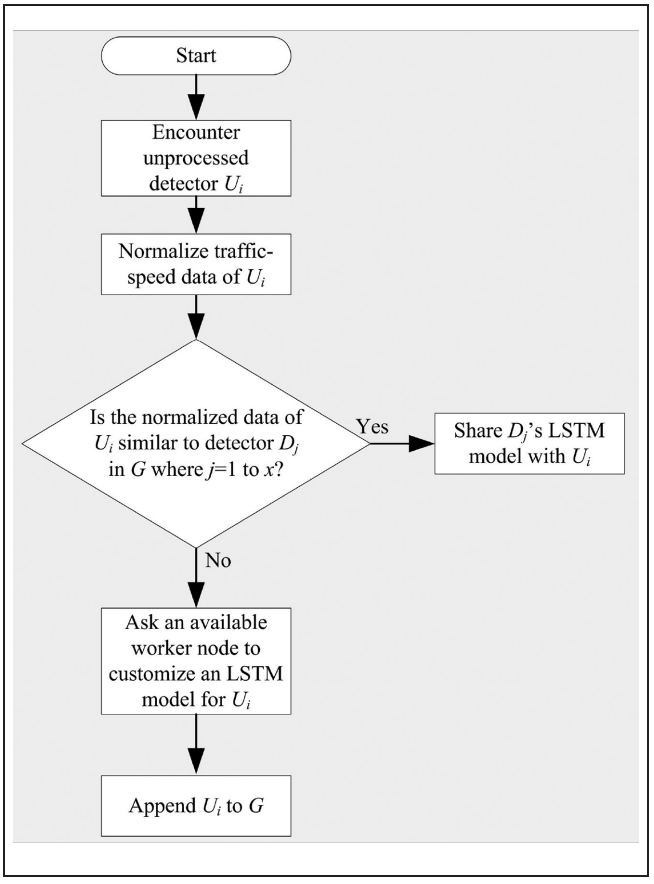


Figure 4. The flowchart of the LSTM auto-tuning and sharing algorithm.
Note: LSTM = long short-term memory.

traffic speed values observed and collected by U_i. Let L_i be the list, and $L_i = \{v_{i,1}, v_{i,2}, \ldots, v_{i,T}\}$ where $v_{i,t}$ is the traffic speed value collected by U_i at time t, $t = 1,2,\ldots,T$. The normalization process divides $v_{i,t}$ by V, which is a predefined fixed value (e.g. 70 to represent the speed limit in mph). Let Nor_i be the normalization result, i.e., $Nor_i = \{n_{i,1}, n_{i,2}, \ldots, n_{i,T}\}$ where $n_{i,T} = \frac{v_{i,T}}{V}$.

The master server decides if it should customize an LSTM model for U_i by sequentially comparing U_i with every detector (denoted by D_j, $j = 1,2,\ldots,x$) in G in terms of their normalized traffic speed pattern based on the following Equation (*12*):

$$\mathrm{AARD}_{i,j} = \frac{1}{T}\sum_{i=1}^{T}\frac{|n_{i,t} - n_{j,t}|}{n_{i,t}} \qquad (2)$$

where $AARD_{i,j}$ is the average absolute relative difference between the traffic speed patterns collected by U_i and D_j, and $n_{j,t}$ is the normalized traffic speed value collected by D_j at time t, where $n_{j,t} = \frac{v_{j,t}}{V}$.

In fact, Equation 2 is similar to the AARE equation shown in Equation 1. Recall that a smaller AARE value indicates that what the prediction method predicts is more similar to the actual one. We follow the same concept to propose Equation 2 and use it to measure if two detectors have observed similar normalized traffic speed patterns. If $AARD_{i,j}$ is less than a predefined threshold thd_{AARD} (implying that U_i and D_j observe a similar normalized traffic speed pattern), the master server directly shares D_j's LSTM model with U_i, without customizing an LSTM model for U_i (see Figure 3: line 9 to line 12).

However, if the master server is unable to find any detector that has observed a similar normalized traffic speed pattern with U_i (i.e., line 13 holds), the master server asks an available worker node to customize an LSTM model for U_i, and then appends U_i to the end of G to indicate that U_i has its own customized LSTM model. Based on how each detector is appended to G, it is clear that every detector in G must have observed a distinct traffic speed pattern.

Several factors might stop an LSTM model from providing satisfactory prediction accuracy. For example, the traffic pattern collected by a detector has changed and no longer follows the previous pattern that is used to train the detector's LSTM model. In order to guarantee fine-grained and satisfactory prediction accuracy for every detector in a growing transportation network, the master server keeps track of every single detector by using the tracking algorithm shown in Figure 5.

Let A be a list of detectors that have been processed by DistTune, and $A = D_1, D_2, \ldots, D_y$ where $y \geq x$. In other words, each detector in A has an LSTM model either inherited from another detector or customized by

```
Input: All detectors in A
Output: Re-customization decisions for all the detectors
Procedure:
Let A = {D1, D2, ..., Dy} be a list of processed detectors;
Let Q = 0;
Let τ be current time;
Let I be a fixed time interval for periodical tracking;
while (τ − Q ≥ I){
  for k = 1 to y{  // y is the total number of detectors in A;
    if the AARE of the LSTM used by detector Dk> thdAARE{
      Ask a worker node to re-customize a LSTM model for Dk;
      if Dk is not in G {
        Append Dk to the end of G;}}}
  Let Q = τ;}
```

Figure 5. The tracking algorithm performed by the master server.
Note: LSTM = long short-term memory.

```
Input: An LSTM (re-)customization request from the master server
Output: An LSTM model that satisfies thdAARE
Procedure:
Let Dφ be the target detector;
Let C be a boolean variable and let C be false;
Employ NMM to create the initial simplex using the default
LSTM hyperparameter setting;
Train the LSTM model with the traffic-speed data previously
collected by Dφ;
if the AARE of the LSTM model is smaller or equal to thdAARE{
  Assign the LSTM model to Dφ;}
else{
  while (C == false){
    Use NMM to transform the simplex by replacing the worst
    vertex with a better one;
    Train the corresponding LSTM model;
    if the corresponding AARE is not bigger than thdAARE{
      Assign the LSTM model to Dφ;
      Let C be true;}}}
```

Figure 6. The LSTM (re-)customization algorithm performed by every worker node.
Note: LSTM = long short-term memory; AARE = average absolute relative error.

DistTune. Periodically, the master server checks if the prediction accuracy of every detector in A is still satisfactory. If a detector (denoted by D_k, $k = 1,2,\ldots,y$) has an AARE value higher than a predefined threshold thd_{AARE}, the master server requests a worker node to re-customize an LSTM model for D_k. After that, the master server appends D_k to the end of G if this is the first time that D_k gets its own LSTM model.

Whenever a worker node receives a customization or re-customization request for detector D_φ (where $D_\varphi \in$ A) from the master server, the worker node conducts Figure 6. First, the worker node employs NMM to create the initial simplex based on the default LSTM hyperparameter setting (i.e., $R_{Learn} = 0.01$, $N_{Layer} = 1$, $N_{Unit} = 2$, $ep = 100$) since this setting introduces less computational cost (*12*). With this setting, the worker node trains a LSTM model with the traffic speed data previously collected by D_φ. If the AARE of this LSTM model is less than or equal to thd_{AARE}, the worker node stops NMM and outputs this LSTM because the desired prediction accuracy has already been reached.

However, if the AARE is higher than thd_{AARE}, the worker node keeps using NMM to transform the simplex by replacing the worst vertex with a better one and repeats the same procedure until the function evaluation at a vertex of the simplex is satisfactory. In other words, the worker node terminates when it finds an LSTM model of which the AARE is less than or equal to thd_{AARE}. The corresponding LSTM model is the final output, and it will be used to predict future traffic speed collected by D_φ.

Customization Protocol. To apply DistTune to a growing transportation network, the following steps should be conducted beforehand:

1. Choose a target transportation network.
2. Collect the traffic speed data observed by all detectors in the target transportation network, and then determine the sizes of training data set and testing data set for LSTM customization.
3. Determine thd_{AARD}. If the value of thd_{AARD} is low, it means that detectors need to have high similarity in their normalized traffic speed patterns to be able to share a single LSTM model.
4. Determine thd_{AARE}. The lower the value of thd_{AARE} is set, the higher prediction accuracy will be achieved.

Once one has finished the above steps, DistTune can be launched to initiate its traffic speed prediction service.

Performance Evaluation

To evaluate DistTune, we conducted three extensive experiments using traffic data from the Caltrans performance measurement system (*34*), which is a database of traffic data collected by detectors placed on state highways throughout California. Freeway I5-N was chosen. This is a major route from the Mexico–United States border to Oregon with a total length of 796.432 miles. In these experiments, DistTune incrementally provides its service until 110 detectors on I5-N are covered to show that DistTune is able to be employed in a growing transportation network and to incrementally customize LSTM models for any detectors that are newly deployed to the network. Note that the distance between two consecutive detectors of the 110 detectors is around 5 miles,

and each detector collected traffic data every 5 min. For each detector, we crawled its traffic data for six continuous working days, and then split it into a training data set (the first 5 days) and a testing data set (the last day). In other words, the training data set and testing data set for each detector consist of 1,440 and 288 data points, respectively. Since all the traffic data was aggregated at 5-min intervals, DistTune follows the same interval for prediction. The reason why the training data set is five continuous working days will be discussed in the next section.

In all the experiments, $thd_{AARE} = 0.05$ and $thd_{AARD} = 0.1$ (*12*). In this paper, if a detector is able to provide 95% prediction accuracy, we consider it satisfactory. This is why we set thd_{AARE} to be 0.05. The same reason for thd_{AARD}: if two detectors have 90% similarity in their normalized traffic speed patterns, we consider these patterns similar. This is why we set thd_{AARD} to be 0.1. In fact, these two thresholds are configurable if one wants to change the degree of similarity or achieve a different level of prediction accuracy.

Three widely used performance metrics (*32*), which are average absolute error (AAE), AARE, and root mean square error (RMSE), were employed in all our experiments to evaluate LSTM prediction accuracy. Please see Equation 1 for AARE. The equations to calculate AAE and RMSE are as follows:

$$\text{AAE} = \frac{1}{X}\sum_{q=1}^{X}|s_q - \widehat{s_q}| \tag{3}$$

$$\text{RMSE} = \sqrt{\frac{1}{X}\sum_{q=1}^{X}(s_q - \widehat{s_q})^2} \tag{4}$$

where
X is the total number of data points for comparison,
q is the index of a data point,
s_q is the actual traffic speed value at q, and
$\widehat{s_q}$ is the forecast traffic speed value at q.
Low values for AAE, AARE, and RMSE indicate that the corresponding LSTM model has high prediction accuracy.

Experiment 1

In this experiment, we designed five scenarios (see Table 2) to evaluate the auto-tuning effectiveness of DistTune by temporarily disabling the LSTM sharing function. In other words, every detector in this experiment will always get its own LSTM model customized by DistTune. In Scenario 1, DistTune only autotunes two hyperparameters. The other two hyperparameters are assigned with small values. In Scenario 2, DistTune autotunes one more hyperparameter. In Scenario 3, all of the four hyperparameters are automatically tuned by DistTune. Scenarios 4 and 5 are similar to Scenarios 1 and 2, respectively, but with an equal or a higher value for each non-autotuned hyperparameter. The goal is to see the impact of these increased values on DistTune.

Table 2. The Hyperparameter Settings in Five Scenarios

Scenario	1	2	3	4	5
R_{Learn}	Tune	Tune	Tune	Tune	Tune
N_{Layer}	1	Tune	Tune	2	Tune
N_{Unit}	10	6	Tune	10	10
ep	Tune	Tune	Tune	Tune	Tune

Note: *ep* = number of epochs.

Four performance metrics were used in this experiment:

- Cumulative LSTM customization time: the summation of the time taken by DistTune to customize an LSTM model for each detector such that the corresponding LSTM satisfies thd_{AARE}.
- Average AAE: the average AAE of all the LSTMs customized by DistTune. See Equation 5.
- Average AARE: the average AARE of all the LSTMs customized by DistTune. See Equation 6.
- Average RMSE: The average RMSE of all the LSTMs customized by DistTune. See Equation 7.

$$\text{Average AAE} = \frac{\sum_{r=1}^{Z}\text{AAE}_r}{Z} \tag{5}$$

$$\text{Average AARE} = \frac{\sum_{r=1}^{Z}\text{AARE}_r}{Z} \tag{6}$$

$$\text{Average RMSE} = \frac{\sum_{r=1}^{Z}\text{RMSE}_r}{Z} \tag{7}$$

In the above three equations, Z is the total number of LSTMs (in this experiment, Z equals 110), and r is the index number of an LSTM, $r = 1,2,...,Z$.

Figure 7 illustrates the cumulative LSTM customization time taken by DistTune in the five scenarios. From the results of the first three scenarios, it seems that the cumulative LSTM customization time increases when more hyperparameters are automatically tuned by DistTune. However, this is not true when the last two scenarios are further considered. We can see that Scenarios 4 and 5 cost more time than Scenario 3, even though they have fewer autotuned hyperparameters than Scenario 3. In other words, increasing the number of autotuned LSTM hyperparameters does not mean that the corresponding LSTM customization time will always increase. The reason why Scenario 4 resulted in the longest LSTM customization time is that the network

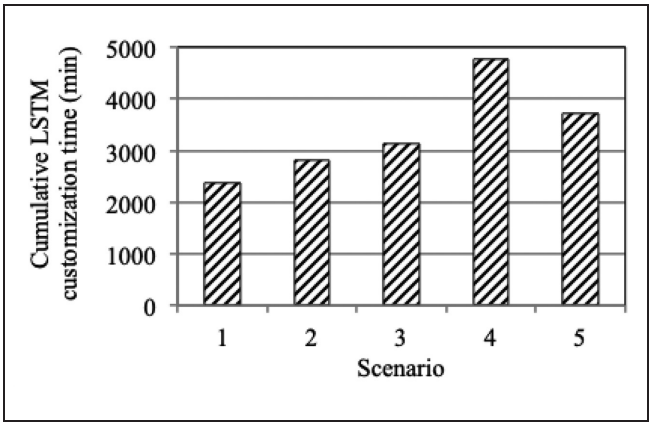


Figure 7. The cumulative LSTM customization time consumed by DistTune in five scenarios when the LSTM sharing function is disabled.
Note: LSTM = long short-term memory.

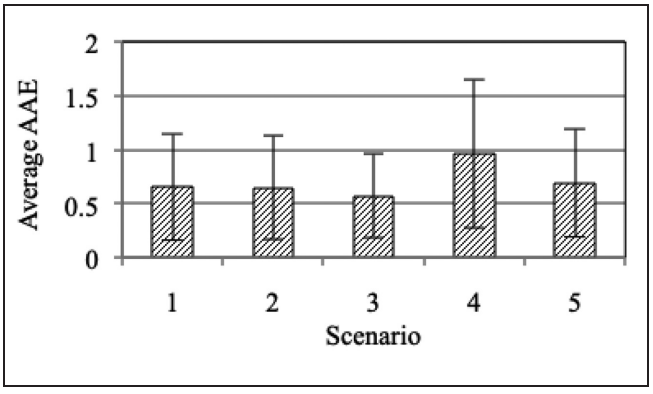


Figure 8. The average AAE results in five scenarios when the LSTM sharing function of DistTune is disabled.
Note: LSTM = long short-term memory; AAE = average absolute error.

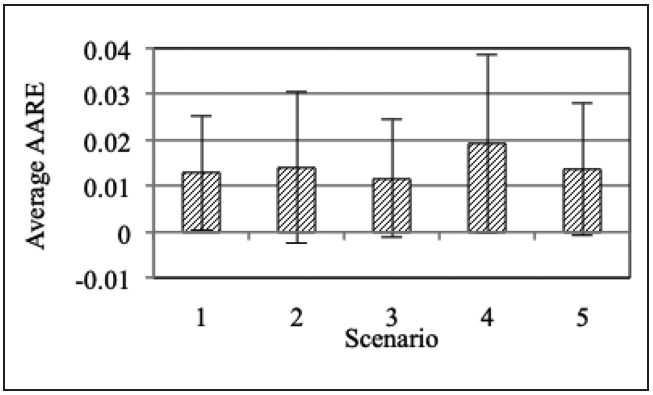


Figure 9. The average AARE results in five scenarios when the LSTM sharing function of DistTune is disabled.
Note: LSTM = Long short-term memory; AARE = Average absolute relative error.

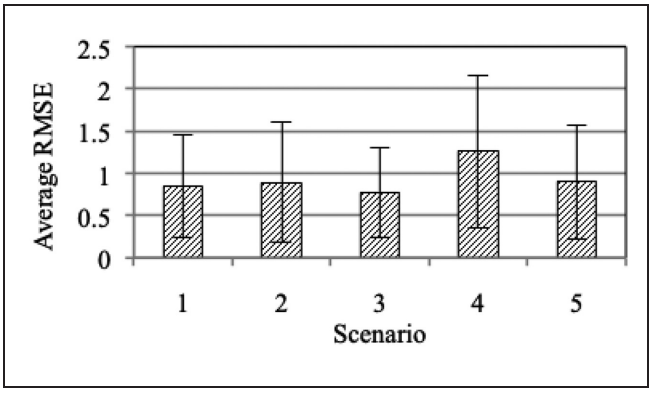


Figure 10. The average RMSE results in five scenarios when the LSTM sharing function of DistTune is disabled.
Note: LSTM = long short-term memory; RMSE = root mean square error.

structure of each LSTM model in Scenario 4 is more complicated than that in all the other scenarios. Therefore, using DistTune to autotune N_{Layer} and N_{Unit} would be more appropriate.

Figures 8 to 10 show the prediction performance of DistTune in the five scenarios when the LSTM sharing function is disabled. It is clear that Scenario 3 has the highest prediction accuracy since its average AAE value, average AARE value, average RMSE value, and the corresponding standard deviation values are the smallest among the five scenarios. The results confirm that autotuning all the considered hyperparameters leads to the best prediction accuracy. However, if the cumulative LSTM customization time shown in Figure 7 is further considered, it appears that Scenario 3 is more computationally expensive than Scenarios 1 and 2 due to its significant increase in computation time, compared with its slight improvement in prediction accuracy. This is because DistTune in this experiment disables the LSTM sharing function. DisTune needs to individually customize an LSTM model for every single detector, so the cumulative LSTM customization time of DistTune cannot have a significant improvement. In the next experiment, we will show how DistTune mitigates this issue by enabling the LSTM sharing function.

Experiment 2

In this experiment, we studied the impact of the LSTM sharing function and the number of worker nodes on the performance of DistTune. To do this, we designed four cases as listed in Table 3. In Cases 1 and 2, the cluster running DistTune has a single worker node. However, the LSTM sharing function of DistTune was disabled in Case 1, whereas it was enabled in Case 2. In Cases 3 and

Table 3. The Details of Four Cases

Case no.	1	2	3	4
Number of worker nodes	1	1	30	30
The LSTM sharing function	Disabled	Enabled	Disabled	Enabled

Note: LSTM = long short-term memory.

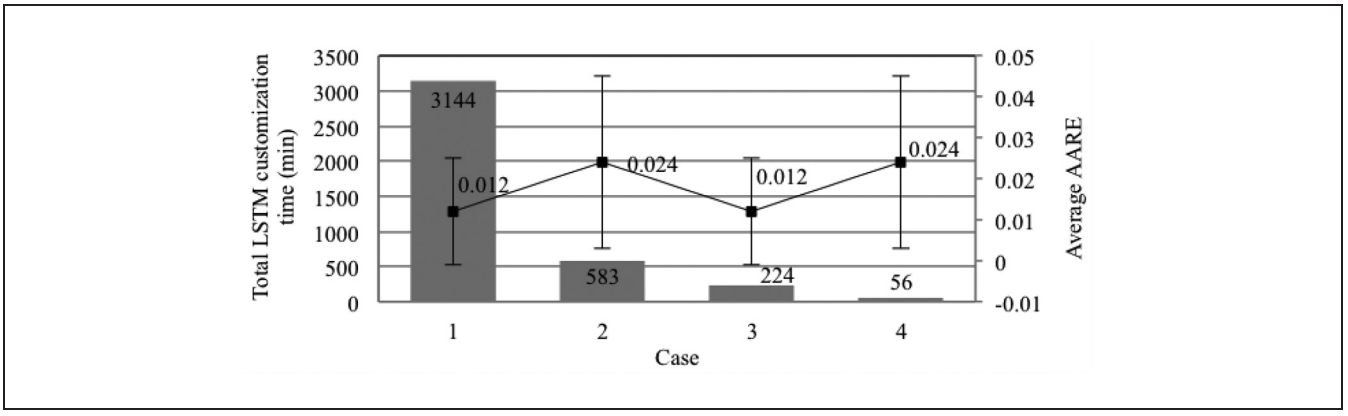


Figure 11. The performance of DistTune in four cases.
Note: LSTM = long short-term memory; AARE = average absolute relative error.

4, the cluster has 30 worker nodes, and the sharing function was disabled and enabled, respectively. Two performance metrics were employed in this experiment: total LSTM customization time and average AARE. Note that the former is the total elapsed time from when DistTune is launched until all 110 detectors have obtained LSTM models.

Figure 11 shows the performance results of DistTune in the four cases. Case 1 is the most time-consuming of the four cases. This is because only one worker node was employed to individually and sequentially customize LSTM models for each of the 110 detectors. This situation turns better in Case 2 since the sharing function was enabled. The total LSTM customization duration reduces by $81.46\% = \frac{3144-583}{3144}$ from Case 1 to Case 2, implying that allowing LSTM models to be shared between detectors significantly reduces the number of LSTM models that DistTune needs to customize, even though there is only one worker node in the cluster supporting DistTune.

When the 30 worker nodes were used and the sharing function was disabled, i.e. Case 3, the total LSTM customization time reduces to 224 min, meaning that increasing the scale of the cluster and using parallel computing also help improve the performance of DistTune. By further enabling detectors to share their LSTM models without individual customization, i.e. Case 4, the total time drops to only 56 min. The reduction is around $75\% = \frac{224-56}{224}$ compared with Case 3. This significant performance improvement is mainly a result of two factors. First, by enabling the LSTM sharing function, DistTune only needed to customize 31 LSTM models for the 110 detectors. Second, the task of LSTM customization was shared by the 30 worker nodes and performed by them in parallel.

On the other hand, from the perspective of average AARE, both Cases 1 and 3 lead to the same result (around 0.012 as shown in Figure 11). This is because the algorithm used by DistTune to customize LSTM models (i.e., NMM) is deterministic. In other words, the result generated by NMM for a given detector is always identical no matter which worker node performs the task.

For the same reason, the average AARE results in Cases 2 and 4 are identical, but they are a little higher than those in Cases 1 and 3. The reason is that not all the detectors in Cases 2 and 4 have their own customized LSTM models.

In Cases 1 and 3, all the 110 detectors require no re-customization since each of them has a customized LSTM model. However, in Cases 2 and 4, seven out of the 110 detectors require re-customization because the LSTM models that they inherited from other detectors are unable to satisfy thd_{AARE}. Nevertheless, the re-customization ratio is low ($6.36\% = \frac{7}{110}$), suggesting that setting thd_{AARD} as 0.1 seems appropriate.

Based on all the above results, we conclude that DistTune in Case 4 provides the best trade-off between time efficiency and prediction accuracy, implying that enabling LSTM sharing and running DistTune on a large cluster are both important for DistTune.

Table 4. The Hyperparameter Setting in Type 1 and Type 2

Re-customization type	R_{Learn}	N_{Layer}	N_{Unit}	ep
Type 1	0.01	1	2	100
Type 2	0.05	1	10	180

Note: ep = number of epochs.

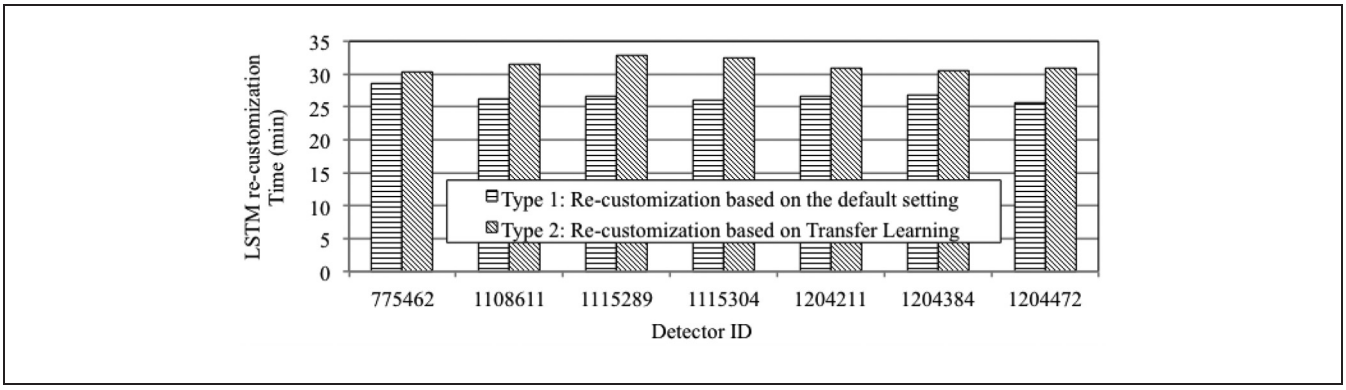


Figure 12. The required LSTM re-customization time for the seven detectors.
Note: LSTM = long short-term memory.

Experiment 3

In this experiment, we evaluated the LSTM re-customization performance of DistTune by re-customizing LSTM models for the seven detectors that require re-customization in Cases 2 and 4 in Experiment 2. Two types of LSTM re-customization were considered:

- Type 1: based on the default hyperparameter setting.
- Type 2: based on Transfer Learning.

In Type 1, DistTune re-customizes an LSTM model for each of the seven detectors by using the default hyperparameter setting (i.e., $R_{Learn} = 0.01$, $N_{Layer} = 1$, $N_{Unit} = 2$, $ep = 100$) to generate the initial simplex for NMM. In Type 2, DistTune re-customizes an LSTM model for detector i by using the LSTM hyperparameter setting of detector j to generate the initial simplex if the LSTM model currently used by detector i is shared by detector j.

Table 4 lists the hyperparameter settings separately used in Types 1 and 2 for the seven detectors. In fact, all the detectors inherited their LSTM models from the same detector (i.e., detector 1118333) since the traffic speed patterns they observed are similar to the one observed by detector 1118333. This is why these detectors have the same hyperparameter setting in Type 2.

Figure 12 shows the LSTM re-customization time for the seven detectors. Surprisingly, Type 2 consumes more time than Type 1 for every detector. The key reason is that the hyperparameter setting in Type 2 contains higher values for N_{Unit} and ep, therefore prolonging the LSTM training time and increasing the required LSTM re-customization time for each detector.

Figures 13 to 15 illustrate the prediction performance of the seven detectors before and after the two types of re-customization were individually performed. Both Types 1 and 2 are able to customize an appropriate LSTM model for each of these detectors and considerably reduce all the AAE, AARE, and RMSE values. It is clear that Type 2 leads to slightly better prediction accuracy than Type 1, but Type 2 consumes more processing time than Type 1. Therefore, choosing Type 1 seems to be more economic. For this reason, the re-customization approach of DistTune is based on Type 1, i.e., the default hyperparameter setting.

Discussion

In this section, we discuss and explain why we chose to use the traffic speed data of five continuous working days to be the training data set for every detector in all our experiments, rather than using a longer period of data. According to our observation, not all detectors observe similar traffic speed patterns all the time. To demonstrate this point, we used one detector, i.e. detector 718086, from freeway I5-N as an example. Figures 16 to 19 illustrate the traffic speed data collected by this detector on freeway I5-N between 4 a.m. and 10 a.m. for 1-week

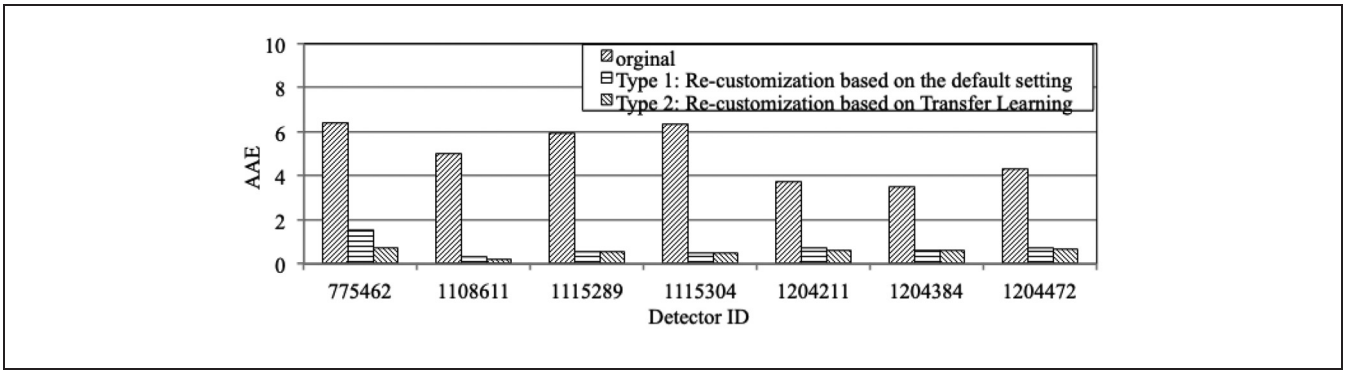


Figure 13. The AAE results before and after two types of LSTM re-customization are individually performed.
Note: LSTM = long short-term memory; AAE = average absolute error.

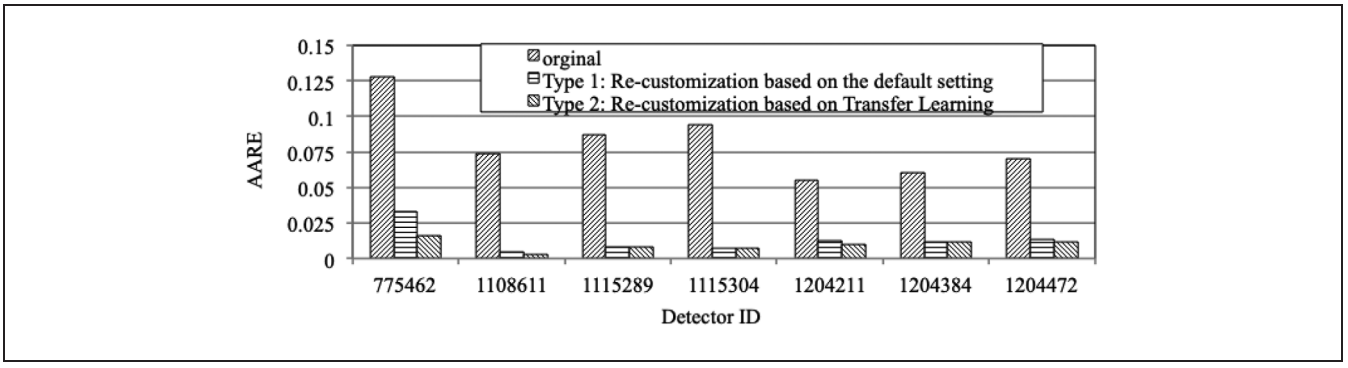


Figure 14. The AARE results before and after two types of LSTM re-customization are individually performed.
Note: LSTM = long short-term memory; AARE = average absolute relative error.

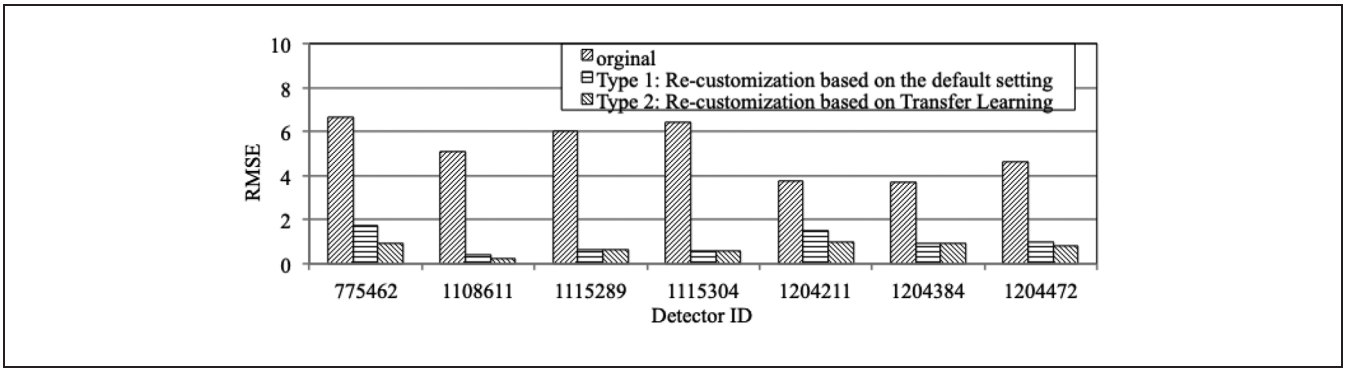


Figure 15. The RMSE results before and after two types of LSTM re-customization are individually performed.
Note: LSTM = long short-term memory; RMSE = root mean square error.

working days, 4-week working days, 8-week working days, and 12-week working days, respectively. When the observation period is one week, we can see from Figure 16 that the traffic speed patterns of these five working days have some deviation. This situation gets worse when we increased the observation time. Please see Figures 17 to 19. It is clear that the traffic speed pattern collected by this detector has more variation as the observation period prolongs. Note that this phenomenon not only appears for this detector, we found that it also happens for many other detectors.

To show the effect of different lengths of training data, we evaluate the corresponding LSTM training time and prediction accuracy in terms of AAE, AARE, and

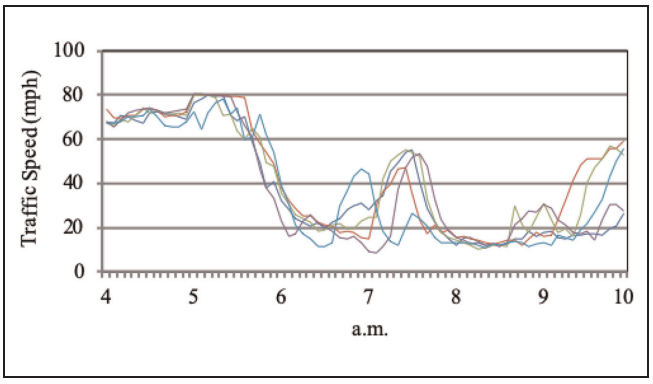


Figure 16. The traffic speed pattern of detector 718086 between 4 a.m. and 10 a.m. for 1-week consecutive working days (from Oct 16, 2017 to Oct. 20, 2017).

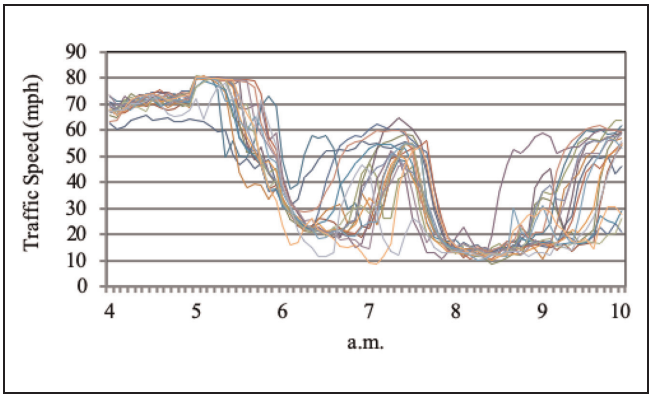


Figure 17. The traffic speed pattern of detector 718086 between 4 a.m. and 10 a.m. for 4-week consecutive working days (from Sept. 25, 2017 to Oct. 20, 2017, that is, the five working days of four consecutive weeks).

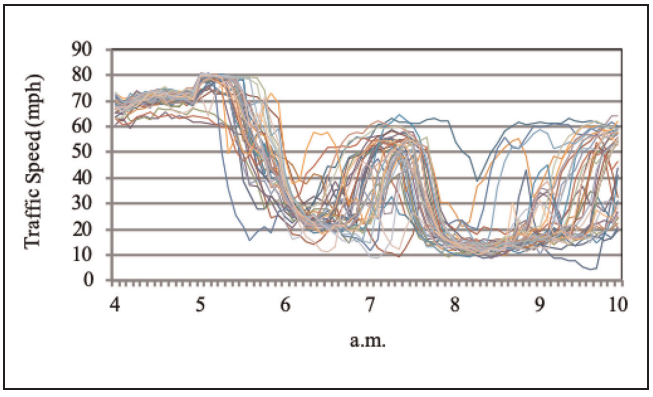


Figure 18. The traffic speed pattern of detector 718086 between 4 a.m. and 10 a.m. for 8-week consecutive working days (from Aug. 28, 2017 to Oct. 20, 2017, that is, the five working days of eight consecutive weeks).

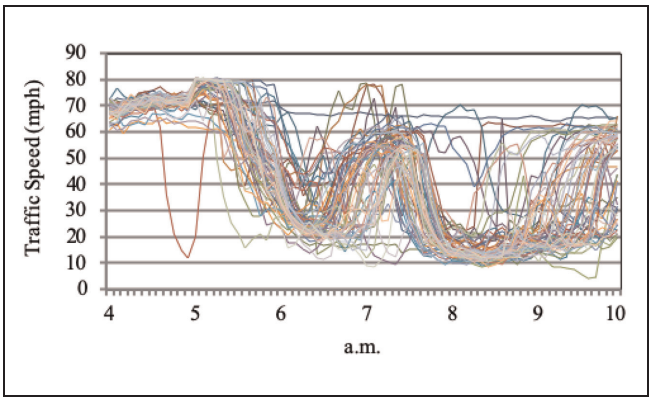


Figure 19. The traffic speed pattern of detector 718086 between 4 a.m. and 10 a.m. for 12-week consecutive working days (from July 31, 2017 to Oct. 20, 2017, that is, the five working days of 12 consecutive weeks).

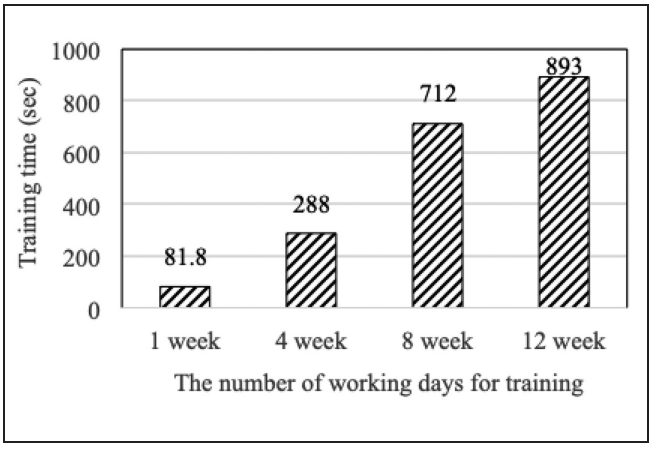


Figure 20. The required training time given different lengths of training data.

RMSE based on the default hyperparameter setting (i.e., $R_{Learn} = 0.01$, $N_{Layer} = 1$, $N_{Unit} = 2$, $ep = 100$). As shown in Figure 20, the 1-week scenario requires the shortest LSTM training time (around 81.8 s) because the size of the training data is only 5 days, which is the shortest among all the four scenarios. The time is only 28.4%($= \frac{81.8}{228}$), 11.5%($= \frac{81.8}{712}$), and 9.2%($= \frac{81.8}{893}$) compared with the training time required in the 4-week, 8-week, and 12-week scenarios, respectively.

On the other hand, from Figure 21, we can also see that the 1-week scenario outperforms the other three scenarios when it comes to prediction accuracy. Apparently, the 1-week scenario leads to the smallest value in AAE, AARE, and RMSE, implying that it leads to the best prediction accuracy among the four scenarios.

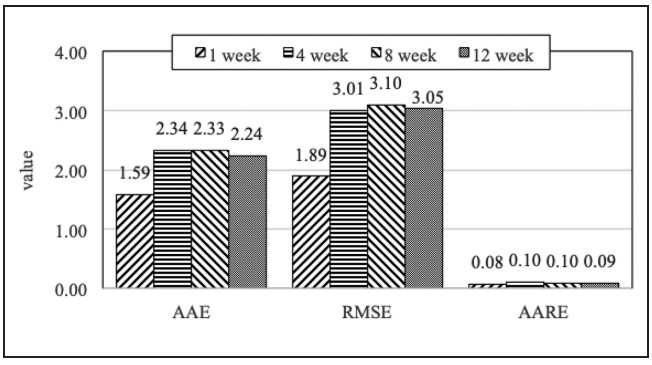


Figure 21. The prediction accuracy under different lengths of training data.
Note: AAE = average absolute error; RMSE = root mean square error; AARE = average absolute relative error.

Based on the above results, we confirm that choosing 1-week traffic speed data as the training data size of LSTM not only saves time, but also achieves higher prediction accuracy because of less deviation in the traffic speed patterns. These two properties are essential since DistTune is designed to provide time-efficient and accurate traffic speed prediction.

Conclusion and Future Work

In this paper, we have introduced DistTune, a distributed scheme to achieve fine-grained, accurate, time-efficient, and adaptive traffic speed prediction for the increasing number of detectors deployed in a growing transportation network. DistTune automatically customizes LSTM models for detectors based on NMM, which was chosen over BOA based on our empirical comparison and evaluation. By allowing LSTM models to be shared between different detectors and employing parallel processing, DistTune successfully addresses the scalability issue, and enables fine-grained and efficient traffic speed prediction. Furthermore, DistTune keeps monitoring detectors and re-customizes their LSTM models when necessary to make sure that their prediction accuracy can always be achieved and guaranteed. Our extensive experiments based on traffic data collected by the Caltrans performance measurement system demonstrate the performance of DistTune and confirm that DistTune provides fine-grained, accurate, time-efficient, and adaptive traffic speed prediction for a growing transportation network.

In this paper, DistTune is implemented on a cluster consisting of only one master server and a set of worker nodes. Such a cluster might have the following risks or limitations. The master server is a single point of failure (SPOF), and it might crash or fail for diverse reasons. In addition, its computation resources might not be able to support the operation of DistTune if DistTune is employed in a large-scale transportation network. On the other hand, the number of worker nodes decides the execution performance of DistTune. In this paper, the number of worker nodes is fixed for simplicity. However, it should be dynamically adjusted over time according to the workload of DistTune.

Therefore, in our future work, we would like to further address the SPOF issue and suggest a highly scalable and elastic solution based on a cloud or a multi-cloud environment for DistTune. Furthermore, we plan to improve the performance of DistTune by considering proper scheduling approaches, such as Lee et al. (*35*) and Lin and Lee (*36*). In addition, we plan to investigate Early Stopping (*37*) and study its impact on the performance of DistTune. Furthermore, we would like to integrate DistTune with other novel techniques such as vehicle trajectory extraction (*38*) to provide better traffic managements and services for growing transportation networks.

Acknowledgments

The authors want to thank the anonymous reviewers for their reviews and valuable suggestions to this paper.

Author Contributions

The authors confirm contribution to the paper as follows: study conception and design: Ming-Chang Lee and Jia-Chun Lin; data collection: Ming-Chang Lee; analysis and interpretation of results: Ming-Chang Lee and Jia-Chun Lin; draft manuscript preparation: Ming-Chang Lee, Jia-Chun Lin, and Ernst Gunnar Gran. All authors reviewed the results and approved the final version of the manuscript.

Declaration of Conflicting Interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the eX^3 project—*Experimental Infrastructure for Exploration of Exascale Computing*, funded by the Research Council of Norway under contract 270053, and the scholarship under project number 80430060 supported by Norwegian University of Science and Technology.

References

1. Ahmed, M. S., and A. R. Cook. Analysis of Freeway Traffic Time-Series Data by Using Box-Jenkins Techniques. *Transportation Research Record: Journal of the Transportation Research Board*, 1979. 722: 1–9.

2. Davis, G. A., and N. L. Nihan. Nonparametric Regression and Short-Term Freeway Traffic Forecasting. *Journal of Transportation Engineering*, Vol. 117, No. 2, 1991, pp. 178–188.
3. Bustillos, B., and Y. C. Chiu. Real-Time Freeway-Experienced Travel Time Prediction Using N-Curve and k Nearest Neighbor Methods. *Transportation Research Record: Journal of the Transportation Research Board*, 2011. 2243: 127–137.
4. Chan, K. Y., T. S. Dillon, J. Singh, and E. Chang. Neural-Network-Based Models for Short-Term Traffic Flow Forecasting using a Hybrid Exponential Smoothing and Levenberg–Marquardt Algorithm. *IEEE Transactions on Intelligent Transportation Systems*, Vol. 13, No. 2, 2012, pp. 644–654.
5. van Lint, J. W. C., S. P. Hoogendoorn, and van Zuylen, H. J. Freeway Travel Time Prediction with State-Space Neural Networks: Modeling State-Space Dynamics with Recurrent Neural Networks. *Transportation Research Record: Journal of the Transportation Research Board*, 2002. 1811: 30–39.
6. Hochreiter, S., and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, Vol. 9, No. 8, 1997, pp. 1735–1780.
7. Ma, X., Z. Tao, Y. Wang, H. Yu, and Y. Wang. Long Short-Term Memory Neural Network for Traffic Speed Prediction Using Remote Microwave Sensor Data. *Transportation Research Part C: Emerging Technologies*, Vol. 54, 2015, pp. 187–197.
8. Yu, R., Y. Li, C. Shahabi, U. Demiryurek, and Y. Liu. Deep Learning: A Generic Approach for Extreme Condition Traffic Forecasting. *Proc., 2017 SIAM International Conference on Data Mining, Society for Industrial and Applied Mathematics*, Texas, 2017, pp. 777–785.
9. Zhao, Z., 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, 2017, pp. 68–75.
10. Okutani, I., and Y. J. Stephanedes. Dynamic Prediction of Traffic Volume through Kalman Filtering Theory. *Transportation Research Part B: Methodological*, Vol. 18, No. 1, 1984, pp. 1–11.
11. Wikipedia contributors. Interstate 5 in California—Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/Interstate_5_in_California. Accessed March 20, 2021.
12. Lee, M.-C., J.-C. Lin, and E. G. Gran. Distributed Fine-Grained Traffic Speed Prediction for Large-Scale Transportation Networks Based on Automatic LSTM Customization and Sharing. *Proc., 26th International European Conference on Parallel and Distributed Computing*, 2020, pp. 234–247. https://arxiv.org/abs/2005.04788.
13. Nelder, J. A., and R. Mead. A Simplex Method for Function Minimization. *The Computer Journal*, Vol. 7, No. 4, 1965, pp. 308–313.
14. Torrey, L., and J. Shavlik. Transfer Learning. In *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques* (E. S. Olivas, J. D. Guerrero, M. Martinez-Sober, J. R. Magdalena-Benedito, and L. Serrano, eds.), IGI Global, Hershey, PA, 2010, pp. 242–264.
15. Mockus, J. *Bayesian Approach to Global Optimization: Theory and Applications*, Vol. 37, Springer Science & Business Media, Cham, 2012.
16. Hochreiter, S. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, Vol. 6, No. 2, 1998, pp. 107–116.
17. Box, G. E., G. M. Jenkins, G. C. Reinsel, and G. M. Ljung. *Time Series Analysis: Forecasting and Control*, 5th ed. John Wiley & Son, Hoboken, NJ, 2015.
18. Lee, S., and D. B. Fambro. Application of Subset Autoregressive Integrated Moving Average Model for Short-Term Freeway Traffic Volume Forecasting. *Transportation Research Record: Journal of the Transportation Research Board*, 1999. 1678: 179–188.
19. Williams, B. Multivariate Vehicular Traffic Flow Prediction: Evaluation of ARIMAX Modeling. *Transportation Research Record: Journal of the Transportation Research Board*, 2001. 1776: 194–200.
20. Williams, B. M., and L. A. Hoel. Modeling and Forecasting Vehicular Traffic Flow as a Seasonal ARIMA Process: Theoretical Basis and Empirical Results. *Journal of Transportation Engineering*, Vol. 129, No. 6, 2003, pp. 664–672.
21. Cui, Z., K. Henrickson, R. Ke, and Y. Wang. Traffic Graph Convolutional Recurrent Neural Network: A Deep Learning Framework for Network-Scale Traffic Learning and Forecasting. *IEEE Transactions on Intelligent Transportation Systems*, Vol. 21, No. 11, 2019, pp. 4883–4894.
22. Le, T. V., R. Oentaryo, S. Liu, and H. C. Lau. Local Gaussian Processes for Efficient Fine-Grained Traffic Speed Prediction. *IEEE Transactions on Big Data*, Vol. 3, No. 2, 2017, pp. 194–207.
23. Jiang, B., and Y. Fei. Vehicle Speed Prediction by Two-Level Data Driven Models in Vehicular Networks. *IEEE Transactions on Intelligent Transportation Systems*, Vol. 18, No. 7, 2017, pp. 1793–1801.
24. Ma, X., H. Yu, Y. Wang, and Y. Wang. Large-Scale Transportation Network Congestion Evolution Prediction Using Deep Learning Theory. *PLoS One*, Vol. 10, No. 3, 2015, p. e0119044.
25. Ma, X., Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang. Learning Traffic as Images: A Deep Convolutional Neural Network for Large-Scale Transportation Network Speed Prediction. *Sensors*, Vol. 17, No. 4, 2017, p. 818.
26. Lee, M.-C., and J.-C. Lin. DALC: Distributed Automatic LSTM Customization for Fine-Grained Traffic Speed Prediction. *Proc., 34th International Conference on Advanced Information Networking and Applications*, 2020, pp. 164–175. https://arxiv.org/abs/2001.09821.
27. Matuszyk, P., R. T. Castillo, D. Kottke, and M. Spiliopoulou. A Comparative Study on Hyperparameter Optimization for Recommender Systems. In Workshop on Recommender Systems and Big Data Analytics, Graz, Austria, 2016, pp. 13–21.
28. Thornton, C., F. Hutter, H. H. Hoos, and K. Leyton-Brown. Auto-WEKA: Combined Selection and Hyperparameter

Optimization of Classification Algorithms. *Proc., 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery*, New York, NY, 2013, pp. 847–855.
29. Hoffman, M. D., E. Brochu, and N. de Freitas. Portfolio Allocation for Bayesian Optimization. *In UAI, Citeseer*, 2011, pp. 327–336.
30. Brochu, E., V. M. Cora, and N. De Freitas. A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning. *arXiv Preprint arXiv:1012.2599*, 2010.
31. Singer, S., and J. Nelder. Nelder-Mead Algorithm. *Scholarpedia*, Vol. 4, No. 7, 2009, p. 2928.
32. Zou, N., J. Wang, G. L. Chang, and J. Paracha. Application of Advanced Traffic Information Systems: Field Test of a Travel-Time Prediction System with Widely Spaced Detectors. *Transportation Research Record: Journal of the Transportation Research Board*, 2009. 2129: 62–72.
33. Xia, D., B. Wang, H. Li, Y. Li, and Z. Zhang. A Distributed Spatial–Temporal Weighted Model on MapReduce for Short-Term Traffic Flow Forecasting. *Neurocomputing*, Vol. 179, 2016, pp. 246–263.
34. California Department of Transportation. PeMS. http://pems.dot.ca.gov/. Accessed March 20, 2021.
35. Lee, M.-C., J.-C. Lin, and R. Yahyapour. Hybrid Job-Driven Scheduling for Virtual MapReduce Clusters. *IEEE Transactions on Parallel and Distributed Systems*, Vol. 27, No. 6, 2016, pp. 1687–1699.
36. Lin, J.-C., and M.-C. Lee. Performance Evaluation of Job Schedulers under Hadoop YARN. *Concurrency and Computation: Practice and Experience*, Vol. 28, No. 9, 2016, pp. 2711–2728.
37. Deeplearning4j. What is Early Stopping? https://deeplearning4j.konduit.ai/tuning-and-training/early-stopping. Accessed March 20, 2021.
38. Chen, X., Z. Li, Y. Yang, L. Qi, and R. Ke. High-Resolution Vehicle Trajectory Extraction and Denoising from Aerial Videos. *IEEE Transactions on Intelligent Transportation Systems*, 2020, pp. 1–13. https://doi.org/10.1109/tits.2020.3003782.