Consensus Ensemble System for Traffic Flow Prediction

Hongyuan Zhan[1], Gabriel Gomes[2], Xiaoye S. Li[3],
Kamesh Madduri[1], Alex Sim[3], Kesheng Wu[3]

[1] Penn State University, University Park, PA, USA
[2] University of California at Berkeley, Berkeley, CA, USA
[3] Lawrence Berkeley National Laboratory, Berkeley, CA, USA

BERKELEY LAB

Consensus Ensemble System for Traffic Flow Prediction

Hongyuan Zhan, Gabriel Gomes, Xiaoye S. Li,
Kamesh Madduri, Alex Sim, *Member, IEEE*, and Kesheng Wu, *Senior Member, IEEE*

***Abstract*—Traffic flow prediction is a key component of an intelligent transportation system. Accurate traffic flow prediction provides a foundation to other tasks such as signal coordination and travel time forecasting. There are many known methods in literature for the short-term traffic flow prediction problem, but their efficacy depends heavily on the traffic characteristics. It is difficult, if not impossible, to pick a single method that works well over time. In this work, we present an automated framework to address this practical issue. Instead of selecting a single method, we combine predictions from multiple methods to generate a consensus traffic flow prediction. We propose an ensemble learning model that exploits the temporal characteristics of the data, and balances the accuracy of individual models and their mutual dependence through a covariance-regularizer. We additionally use a pruning scheme to remove anomalous individual predictions. We apply our proposed model to multi-step-ahead arterial roadway flow prediction. In tests, our method consistently outperforms recently published ensemble prediction methods based on Ridge Regression and Lasso. Our method also produces steady results even when the standalone models and other ensemble methods make wildly exaggerated predictions.**

***Index Terms*—Ensemble learning, model-combination, machine learning, traffic flow prediction.**

I. Introduction

There are many useful applications for short-term (up to one hour) prediction of traffic state. Speed predictions are used by traveler information systems to forecast travel times along routes. Traffic management centers in large urban areas increasingly employ real-time traffic prediction for decision support [1]. The available techniques for making these forecasts fall into two broad categories: those that employ physical models (e.g., the cell-transmission model [2]) in their calculations, and those that do not. Applications in traffic management typically fall into the former category, since the traffic control algorithms being tested can be expressed naturally in terms of the parameters of a physical model. Travel information systems usually employ non-physical models and use techniques of statistical learning to train their parameters.

In this work, we focus on the problem of forecasting traffic flows, as measured by fixed pavement sensors. There is a large body of work on the use of time-series, non-parametric, and other methods for predicting traffic flow. Lv et al. [3] provide a good summary of literature. Early work on this topic includes that of Nicholson and Swann [4], who used spectral analysis to extract trends from measured flow through the Mersey Queensway tunnel in Liverpool, England. Okutani and Stephanedes [5] used a Kalman filter to estimate the parameters of a linear prediction model. Many authors have applied time-series techniques to this problem. Hamed et al. [6] fitted an Autoregressive integrated moving average (ARIMA) model to traffic data in Amman, Jordan. Van Der Voort et al. [7] used a self-organizing neural network to cluster data prior to fitting it with ARIMA. Lee and Fambro [8] used the Aikaike information criterion for ARIMA model selection. Ghosh et al. [9] have noted the importance of spatial correlations and applied multivariate techniques. Stephanedes et al. [10] developed a state-space model for predicting flows on a network. Williams [11] compared seasonal ARIMA and ARIMAX and found improvement by using upstream flows as an external input. Wu et al. [12] introduced historical measurements as the external input to an ARMAX model. Pascale and Nicoli [13] used an adaptive Bayesian network to more generally capture the spatial and temporal correlation amongst flows on a freeway. Recently, Coogan et al. [14] used principal components analysis and partial least squares to extract trends from flows through an intersection.

The problem of traffic flow prediction can be viewed as a subcomponent of either a statistical or a physical predictor of traffic state. To predict trip times, for example, the flows can be directly translated into speeds by means of an empirical speed-flow curve [15]. Traffic forecasts based on physical models require a prediction of the model *input*. This input can be expressed either as an intensity of flows entering the network at its boundaries (aggregate demand modeling), or as a matrix of trip counts from each origin to each destination (disaggregate OD flows). The usual approach in the disaggregate case is to compute the OD matrix from predictions of internal flows, as in [16]. In the aggregate case, the boundary demands are assumed to be unaffected by the internal state of the network. Hence, a methodology for predicting sensor flows is useful in both the aggregate and disaggregate forms of physical model-based prediction.

When prediction results are provided by multiple models, traffic controllers have to either trust the prediction from one of the methods, or make some consensus judgement from the available results. The present paper focuses on an ensemble learning approach for consensus traffic flow prediction, which aims to combine different base predictions to produce more accurate and stable results.

Our proposed ensemble model extends the concept of stacking [17], [18], in which the combined model learns from the

H. Zhan and K. Madduri are with the department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA, USA.

G. Gomes is with the Partners for Advanced Transportation Technology, University of California at Berkeley, Berkeley, CA, USA.

X. S. Li, A. Sim, and K. Wu are with the Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA, USA.

mistakes made by base models in the past. Model combination could be viewed as a process to assign weights for each sub-model and generate a weighted average prediction. For example, given M base predictions $\{f_m\}_{m=1}^{M}$, we could take the simple average $\frac{1}{M}\sum_{m=1}^{M} f_m$. Despite tremendous success of ensemble learning in other areas, this approach is relatively new for transportation applications. Bagging predictors, an ensemble strategy, was previously studied by Sun [19] for short-term traffic prediction. Hou et al. [20] recently used random forests for traffic prediction in urban work zones. Both of these methods rely on bootstrapping the training set and train the model based on different sub-samples. The two methods mentioned above are not in the multi-model combination paradigm considered in this paper, since they use a same simple model, but trained on different data. In contrast, our approach belongs to the category of multi-model ensemble learning, which combines the forecast from multiple sub-models. The advantage of our method is to benefit from model diversity and achieve more robust results. Recently, other traffic researchers also suggested using multi-model methods to handle uncertainties. Li et al. [21] applied Ridge regression and Lasso regression to combine the output from several traffic flow simulation models. Model aggregation by neural network were also examined in [19], [22]. In neural network models, frequent parameter updates are infeasible due to the high computational cost during the training stage, whereas our proposed method reduces to a convex quadratic program, which could be solved efficiently. A fuzzy rule-based system with Genetic Algorithm was proposed in [23].

In this work, we study a consensus combination of five representative methods from time series modeling and machine learning, for the multi-step-ahead arterial roadway flow forecasting problem. We conduct a systematic performance evaluation of the base models and the ensemble model. The experimental procedure aims to mimic the application scenario. Our ensemble learning method robustly combines the sub-models. We found that our method consistently outperforms simple average combination and two related multi-model based traffic flow prediction methods [21], and improves the sub-model forecast results even in the cases when the other ensemble methods fail. We demonstrate how to make automated consensus predictions from base methods without resorting to intervention by traffic controllers after the system has being employed.

II. Base Methods

In the traffic flow forecasting setting, flow data forms a time series. In this work, we assume each flow time series is univariate: flow forecasts for each detector are made separately. Let $[y_1, y_2, \cdots, y_{\hat{t}}]$ denote a time series of historical observations of the traffic flow. The measurement stream arrives in a batch of l most recent observations in every l steps. Predictions for the upcoming batch of measurements $[y_{\hat{t}+1}, y_{\hat{t}+2}, \cdots, y_{\hat{t}+l}]$ are of interest. We assume that there is an underlying autoregressive function f such that

$$y_{t+1} = f(\mathbf{x}_t) + \epsilon_{t+1}$$
$$\text{where } \mathbf{x}_t = [y_{t-p+1}, y_{t-p+2}, \cdots, y_t]^T \in \mathbb{R}^p$$

The measurement y_{t+1} is a mapping from past values with additive i.i.d. Gaussian white noise $\epsilon_{t+1} \sim \mathcal{N}(0, \sigma_\epsilon^2)$. The function f and its order p is unknown. A forecasting model aims to approximate f and make an appropriate choice of p.

There are many traffic flow forecasting models proposed in the literature. Our goal is to integrate existing methodologies and produce a consensus prediction. We do not have restrictions on the specific base methods being used in the consensus system. Nevertheless, we would like to include the best methods from a variety of different types. We aim to produce a consensus model that will be consistently better than these base methods. In this section, we described several base prediction methods used to test our system. The chosen base models are representative from the following categories: 1. time series models; 2. latent variable models; 3. maximum-margin machine learning methods; 4. kernel machine learning methods; and 5. Bayesian models.

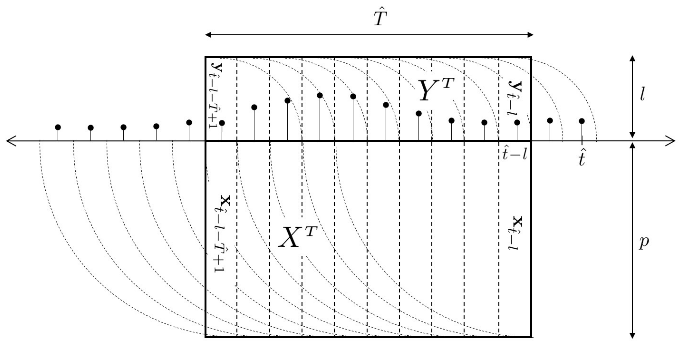

Fig. 1. Illustration of the time-series notation. This is a stem-plot of data samples up to time $\hat{t}$. The two boxes represent the training data matrices, with observations X and responses Y. The curved lines indicate the samples that are collected in each of the columns. For example, $\mathbf{x}_{\hat{t}-l}$ comprises samples from $\hat{t}-l-p+1$ to $\hat{t}-l$, whereas $\mathbf{y}_{\hat{t}-l}$ comprises samples from $\hat{t}-l+1$ to $\hat{t}$.

In the following sections, let

$$\begin{aligned}
\mathbf{x}_t &= [y_{t-p+1}, \cdots, y_t]^T \in \mathbb{R}^p \\
X &= [\mathbf{x}_{\hat{t}-l-\hat{T}+1}, \mathbf{x}_{\hat{t}-l-\hat{T}+2}, \cdots, \mathbf{x}_{\hat{t}-l}]^T \in \mathbb{R}^{\hat{T}\times p} \\
\mathbf{y}_t &= [y_{t+1}, y_{t+2}, \cdots, y_{t+l}]^T \in \mathbb{R}^l \\
Y &= [\mathbf{y}_{\hat{t}-l-\hat{T}+1}, \mathbf{y}_{\hat{t}-l-\hat{T}+2}, \cdots, \mathbf{y}_{\hat{t}-l}]^T \in \mathbb{R}^{\hat{T}\times l}.
\end{aligned}$$

(X, Y) comprises the training data for each base model. A row of X uses the p past observations as the explanatory variables, and the responses are collected in Y. $\hat{T}$ denotes the number of flow samples used for training. Also let $\{X_k\}_{k=1,\cdots,p} \in \mathbb{R}^{\hat{T}}$ be columns of X, and $\{Y_k\}_{k=1,\cdots,l} \in \mathbb{R}^{\hat{T}}$ be columns of Y. The notations for time indices used in the subsequent sections are given in Table I.

A. ARMAX

The ARMAX model for traffic flow prediction was studied in [12]. The ARMAX model describes the evolution of traffic flows over time via the stochastic difference equation

$$A(q^{-1})y_t = B(q^{-1})u_t + C(q^{-1})w_t, \tag{1}$$

TABLE I
NOTATION USED FOR DIFFERENT TIME INDICES. ALL VARIABLES ARE POSITIVE INTEGERS.

Variable	Description
t	Indices for time
$\hat{t}$	Present time
l	Verification and prediction horizon: number of time steps before the new batch of observations arrived
p	Dimension of $\mathbf{x}_t$ in the base models
$\hat{T}$	Number of past observations for training base models

where y_t is the traffic flow at time step t, u_t is the historical sample average flow value at the same time of day. w_t is assumed to be a zero-mean innovation sequence such that $E(w_t w_{t-j}) = 0$ for all $0 \leq j \leq t$. Here q^{-1} is the backward shift operator defined by $q^{-1}y_t = y_{t-1}$, $A(q^{-1}), B(q^{-1}), C(q^{-1})$ are scalar polynomials in the backward shift operators

$$\begin{aligned} A(q^{-1}) &= 1 + a_1 q^{-1} + \cdots + a_{n_a} q^{-n_a} \\ B(q^{-1}) &= 1 + b_1 q^{-1} + \cdots + b_{n_b} q^{-n_b} \\ C(q^{-1}) &= 1 + c_1 q^{-1} + \cdots + c_{n_c} q^{-n_c}, \end{aligned}$$

where the order n_a, n_b, n_c are hyperparameters. Coefficients of the polynomial are estimated via the recursive least squares adaptation algorithm [12], [24].

B. Partial Least Squares

Coogan et al. [14] recently applied the Partial Least Squares (PLS) technique to short-term traffic flow prediction. The key idea of PLS is to maximally exploit the covariance between flows in the prediction horizon and flows in the memory window. Let $\bar{\mathbf{x}} \in \mathbb{R}^p$ be the sample mean in X, and $\bar{\mathbf{y}} \in \mathbb{R}^l$ be the sample mean in Y. Subtract $\bar{\mathbf{x}}^T$ from each row of X and denote the result as $\tilde{X} \in \mathbb{R}^{\hat{T} \times p}$. Similarly, denote $\tilde{Y} \in \mathbb{R}^{\hat{T} \times l}$ the matrix obtained by removing the sample mean $\bar{\mathbf{y}}^T$ from each row of Y. PLS exploits covariance by finding a pair of vectors $(\mathbf{r}^*, \mathbf{s}^*) \in \mathbb{R}^p \times \mathbb{R}^l$, such that

$$(\mathbf{r}^*,\ \mathbf{s}^*) = \underset{(\mathbf{r},\ \mathbf{s})}{\operatorname{argmax}}\, \mathbf{r}^T(\tilde{X}^T\tilde{Y})\mathbf{s}, \quad \text{s.t.} \quad \|\mathbf{r}\|_2^2 = \|\mathbf{s}\|_2^2 = 1, \tag{2}$$

Notice that $\tilde{X}^T\tilde{Y} \in \mathbb{R}^{p \times l}$ is the sample covariance matrix of flows across different times. Intuitively, we seek a pair of projection directions $(\mathbf{r}^*,\ \mathbf{s}^*)$ in Eq. (2), which maximizes the sample covariance after the projection. The optimization problem in Eq. (2) could be solved by a partial SVD of $\tilde{X}^T\tilde{Y}$; the optimal projection direction $(\mathbf{r}^*,\ \mathbf{s}^*)$ are the first left and right singular vectors respectively. Define $\mathbf{w} := \tilde{X}\mathbf{r}^* \in \mathbb{R}^{\hat{T}}$. The orthogonal projection of column vector $\tilde{X}_k$ and $\tilde{Y}_k$ onto $\mathbf{w}$ are

$$\begin{aligned} p_k &:= \frac{\langle \tilde{X}_k, \mathbf{w} \rangle}{\|w\|_2^2}, \qquad k = 1, \cdots, p \\ c_k &:= \frac{\langle \tilde{Y}_k, \mathbf{w} \rangle}{\|w\|_2^2}, \qquad k = 1, \cdots, l \end{aligned} \tag{3}$$

respectively. Collectively we have $\mathbf{p} = \frac{\tilde{X}^T\mathbf{w}}{\|w\|_2^2}$ and $\mathbf{c} = \frac{\tilde{Y}^T\mathbf{w}}{\|w\|_2^2}$, which are called the first predictor component and first prediction component respectively [14]. The outer-product $\mathbf{w}\mathbf{p}^T$ provides a *rank-one* approximation to $\tilde{X}$, and similarly $\mathbf{w}\mathbf{c}^T$ is a rank-one approximation to $\tilde{Y}$. Next, $\tilde{X}$ and $\tilde{Y}$ are deflated to remove the effects contributed by the rank-one matrices,

$$\tilde{X} \leftarrow \tilde{X} - \mathbf{w}\mathbf{p}^T, \qquad \tilde{Y} \leftarrow \tilde{Y} - \mathbf{w}\mathbf{c}^T. \tag{4}$$

Equation (2) and (3) and the deflation (4) is repeatedly applied until we get N predictor and prediction components, i.e., a predictor component matrix $P \in \mathbb{R}^{p \times N}$ and a prediction component matrix $C \in \mathbb{R}^{l \times N}$. To make predictions for time $\hat{t} + k$, where $1 \leq k \leq l$, first project flows in the memory window onto the latent component matrix P:

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \|\mathbf{x}_{\hat{t}} - \bar{\mathbf{x}} - P\mathbf{w}\|_2^2. \tag{5}$$

The PLS predicted flow is then computed by

$$[f_{\text{pls},\hat{t}+1}, \cdots, f_{\text{pls},\hat{t}+l}]^T = \bar{\mathbf{y}} + C\hat{\mathbf{w}} \in \mathbb{R}^l, \tag{6}$$

The formulation we apply here is essentially the NIPALS-based PLS [25].

C. Support Vector Regression

Support vector machine (SVM) is one of the most successful machine learning methods. The variant of SVM for the regression setting is called the Support Vector Regression (SVR) [26], [27]. Traffic flow prediction with SVR has been previously studied in [28], [29].

The regression model is given by

$$f_{\text{svr}}(\mathbf{x}_t) = \phi(\mathbf{x}_t)^T\mathbf{w} + b, \tag{7}$$

where ϕ is a user-defined function that maps the flows during the memory window into features in higher dimension, and b is a bias term. We trained the SVR models separately for each prediction time-step $\{\hat{t} + k\}_{k=1}^{l}$. The SVR objective function at time $\hat{t} + k$ is

$$\min_{\mathbf{w},b} \sum_{t=\hat{t}-l-\hat{T}+1}^{\hat{t}-l} L_\epsilon(f_{\text{svr}}(\mathbf{x}_t) - y_{t+k}) + \lambda\|\mathbf{w}\|^2. \tag{8}$$

This is called the structural risk minimization framework [30], where the term $L_\epsilon(f_{\text{svr}}(\mathbf{x}_t) - y_{t+k})$ is the empirical loss we want to minimize from the training data, $\lambda\|\mathbf{w}\|^2$ controls the complexity of the model to avoid overfitting. $\lambda \in \mathbb{R}_+$ is a hyperparameter balancing the two terms.

The loss function used in SVR is defined by

$$L_\epsilon(f_{\text{svr}}(\mathbf{x}) - y) = \begin{cases} 0, & \text{if } |f_{\text{svr}}(\mathbf{x}) - y| < \epsilon \\ |f_{\text{svr}}(\mathbf{x}) - y| - \epsilon & \text{otherwise.} \end{cases} \tag{9}$$

Therefore, using L_ϵ, we allow the learned model to deviate from the true data by a margin ϵ without penalty, where $\epsilon \geq 0$ is supplied by the user. Equation (8) can be transformed into a quadratic programming formulation by introducing slack variables [26], [31]. Many state-of-the-art solvers for SVR use sequential minimal optimization (SMO)-type algorithms [32], [33]. In our experiments, we used the MATLAB SMO solver and the MATLAB SVR default hyperparameter values for λ and ϵ. After the optimal $\mathbf{w}^*$ and b^* for time $\hat{t} + k$ is learned, the SVM predicted flow is produced by

$$f_{\text{svr},\hat{t}+k} = \phi(\mathbf{x}_{\hat{t}})^T\mathbf{w}^* + b^*. \tag{10}$$

D. Kernel Ridge Regression

In kernel ridge regression (KRR), traffic flows in the memory window are first transformed by a mapping $\mathbf{x}_t \mapsto \phi(\mathbf{x}_t)$, then future flows $\{y_{t+k}\}_{k=1}^{l}$ are modeled by linear transformation of $\phi(\mathbf{x}_t)$. For simplicity, we assume that $\mathbf{x}_t$ and y_t are mean-centered by subtracting the sample average flows from each data point. Ridge regression balances the squared error and model complexity by solving

$$\min_{\mathbf{w}} \sum_{t=\hat{t}-l-\hat{T}+1}^{\hat{t}-l} \left(y_{t+k} - \phi(\mathbf{x}_t)^T \mathbf{w}\right)^2 + \lambda \|\mathbf{w}\|_2^2, \tag{11}$$

where $\lambda \in \mathbb{R}_+$ is a hyperparameter. The regularization term $\lambda\|\mathbf{w}\|_2^2$ prevents overfitting of the model. By the Representer theorem [34], there is a vector $\boldsymbol{\alpha} \in \mathbb{R}^{\hat{T}}$, such that the optimal solution vector $\mathbf{w}^*$ for Eq. (11) can be expressed as

$$\mathbf{w}^* = \sum_{t=\hat{t}-l-\hat{T}+1}^{\hat{t}-l} \alpha_t \phi(\mathbf{x}_t) = \Phi^T \boldsymbol{\alpha}, \tag{12}$$

where $\Phi^T = [\phi(\mathbf{x}_{\hat{t}-l}), \phi(\mathbf{x}_{\hat{t}-l+1}), \cdots, \phi(\mathbf{x}_{\hat{t}-l-\hat{T}+1})]$. Substituting the weight representation in Eq. (12) into Eq. (11), we have

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} \sum_{t=\hat{t}-l-\hat{T}+1}^{\hat{t}-l} \left(y_{t+k} - \phi(\mathbf{x}_t)^T \Phi^T \boldsymbol{\alpha}\right)^2 + \lambda \boldsymbol{\alpha} \Phi \Phi^T \boldsymbol{\alpha} \\ &= \min_{\boldsymbol{\alpha}} \ \|Y_k - \Phi\Phi^T \boldsymbol{\alpha}\|_2^2 + \lambda \boldsymbol{\alpha} \Phi\Phi^T \boldsymbol{\alpha} \\ &= \min_{\boldsymbol{\alpha}} \ \|Y_k - K\boldsymbol{\alpha}\|_2^2 + \lambda \boldsymbol{\alpha} K \boldsymbol{\alpha}, \end{aligned} \tag{13}$$

where $Y_k \in \mathbf{R}^{\hat{T}}$ is a vector of flows $[y_{t+k}]_{t=\hat{t}-l-\hat{T}+1}^{\hat{t}-l}$, and $K := \Phi\Phi^T$. Notice that we can avoid explicitly constructing the transformed explanatory variables $\phi(\mathbf{x}_t)$ in equation (13) by specifying a kernel function k such that $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$. In addition, Eq. (13) is unconstrained and convex, which allows an analytic solution. Setting the gradient of the objective function with respect to $\boldsymbol{\alpha}$ to zero, the optimal solution is given by

$$\boldsymbol{\alpha}^* = \left(K + \lambda I\right)^{-1} Y_k. \tag{14}$$

After $\boldsymbol{\alpha}^*$ is obtained, the optimal solution to equation (11) can be computed as $\mathbf{w}^* = \Phi^T \boldsymbol{\alpha}^*$. The time $\hat{t}+k$ prediction is

$$f_{\text{krr},\hat{t}+k} = \phi(\mathbf{x}_{\hat{t}})^T \left(\Phi^T \boldsymbol{\alpha}^*\right) = \sum_{t=\hat{t}-l-\hat{T}+1}^{\hat{t}-l} \alpha_t^* k(\mathbf{x}_{\hat{t}}, \mathbf{x}_t). \tag{15}$$

Again, the mapping ϕ does not come into play directly, the computation can be entirely done via the kernel.

E. Gaussian Process Regression

Gaussian process regression (GPR) is a non-parametric Bayesian method closely related to kernel ridge regression. Xie et al. applied Gaussian process regression for inter-state highway flow prediction [35]. GPR differs from kernel ridge regression mainly from the model derivation procedure and the use of Bayesian posterior distribution. In this work, we use the Gaussian process regression model described in [31], [36]. For time $\hat{t}+k$, the flow is modeled by

$$\begin{aligned} y_{\hat{t}+k} &= f_{\text{gpr},\hat{t}+k}(\mathbf{x}_{\hat{t}}) + \epsilon \\ f_{\text{gpr},\hat{t}+k} &\sim \mathcal{GP}\Big(0, k(\mathbf{x}, \mathbf{x}')\Big) \\ \epsilon &\overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2) \end{aligned} \tag{16}$$

where $\mathcal{GP}\Big(0, k(\mathbf{x}, \mathbf{x}')\Big)$ denotes a Gaussian process with covariance matrix parametrized by the kernel function $k(\mathbf{x}, \mathbf{x}')$. We assume the residual ϵ is independent of $f_{\text{gpr},\hat{t}+k}$. The zero-mean Gaussian process is used here, since, without loss of generality, the sample mean of flow values can be subtracted from y_t [35]. Under model (16), the covariance between traffic flows at t and t' is

$$\text{cov}(y_t, y_{t'}) = \sigma_f k(\mathbf{x}_t, \mathbf{x}_{t'}) + \sigma^2 \delta_{tt'}, \tag{17}$$

where $\delta_{tt'}$ is a Kronecker delta function which equals 1 if $t = t'$, and 0 otherwise. The joint distribution between historical flows Y_k and the modeled flow $f_{\text{gpr},\hat{t}+k}(\mathbf{x}_{\hat{t}})$ is

$$\begin{bmatrix} Y_k \\ f_{\text{gpr},\hat{t}+k} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} k(X,X) + \sigma^2 I & k(X, \mathbf{x}_{\hat{t}}) \\ k(\mathbf{x}_{\hat{t}}, X) & k(\mathbf{x}_{\hat{t}}, \mathbf{x}_{\hat{t}}) \end{bmatrix}\right). \tag{18}$$

The posterior predictive distribution [36] of $f_{\text{gpr},\hat{t}+k}$, conditional on $\mathbf{x}_{\hat{t}}$ and historical flows, is

$$p\Big(f_{\text{gpr},\hat{t}+k} | \mathbf{x}_{\hat{t}}, Y_k, X\Big) = \mathcal{N}\Big(\mu_{\text{gpr},\hat{t}+k}, \text{cov}\left(f_{\text{gpr},\hat{t}+k}\right)\Big)$$

There are closed-form formulas to compute the posterior mean $\mu_{\text{gpr},\hat{t}+k}$ and posterior covariance $\text{cov}\left(f_{\text{gpr},\hat{t}+k}\right)$ [36]. We use the posterior mean $\mu_{\text{gpr},\hat{t}+k}$ as Gaussian process point estimation for flows at time k, i.e., $f_{\text{gpr},\hat{t}+k} := \mu_{\text{gpr},\hat{t}+k}$.

III. Ensemble Learning

Use $\{f_{mt}\}_{m=1}^{M}$ to denote a collection of forecasts from M models at time t. In practice, it will often be necessary to select a single forecast. Therefore, combining the results from individual predictors will be valuable in practice - a consensus outcome potentially improves robustness and prediction accuracy. We propose a new consensus ensemble model with the following algorithmic contributions:

1. a time-dependent loss function exploiting the temporal data characteristics;
2. a new covariance-based regularizer to balance model diversity and accuracy to learn the parameters;
3. a pruning scheme to safe-guard against prediction anomaly.

Traditionally, consensus ensemble methods build a meta-model by convex combination of the base models.

$$\bar{f} = \sum_{m=1}^{M} \beta_m f_m, \quad \sum_{m=1}^{M} \beta_m = 1, \quad \beta_m \geq 0 \ \ \forall m$$

Stack regression is a classical consensus ensemble methods in machine learning for computing the weights $\{\beta_m\}_{m=1}^{M}$. In many machine learning applications, stack regression implicitly assumes that samples in the training set and test set are independently distributed. The training data is shuffled and

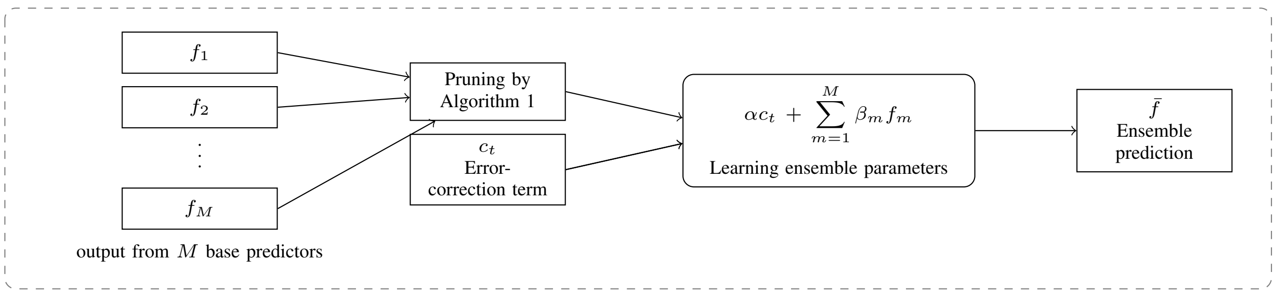
f_1
f_2
$\vdots$
f_M
output from M base predictors
Pruning by Algorithm 1
c_t Error-correction term
$\alpha c_t + \sum_{m=1}^{M} \beta_m f_m$
Learning ensemble parameters
$\bar{f}$ Ensemble prediction

Fig. 2. Overview of the consensus ensemble method proposed in this paper. α and $\{\beta_m\}_{m=1}^M$ are ensemble parameters learned from data.

partitioned. Parts of the training set is used for fitting the sub-models, and the left-out training data is used for computing the ensemble weights. In the traffic flow time series prediction setting, due to inherent seasonality and other temporal correlations, the shuffling and leave-out operations change the empirical distribution. It is not reasonable to remove some observations $y_{t<\hat{t}}$ and train the base models using samples before and after the removed observations. Therefore, it is necessary to modify stack regression by building the meta-training set in a sequential manner, without data shuffling. We describe a rolling training procedure in more detail in section IV-B. In addition, recent data might be more representative than older ones in a temporal prediction task. Finally, the prediction errors may be correlated over time. With these concerns in mind, we now describe our ensemble model for consensus traffic prediction. Figure 2 provides an overview of our proposed ensemble learning method.

A. Unsupervised Pruning

The proposed system has four stages as outlined in Figure 2. In the first stage, the predictions from individual models are produced. Each base prediction method has its own degree of robustness against noise and corruptions in the data. Some base learners may occasionally produce unexpected abnormal predictions due to observation noise. A base model may behave well in training, but make an anomalous prediction due to noise or corruptions in the most recently collected data. Therefore, if an anomalous prediction is included in the ensemble model in the third and fourth stages, the final consensus prediction will be affected. Hence, we propose a rule-based pruning step as a safe-guard against prediction outliers before ensemble learning steps.

Algorithm 1 Pruning prediction outliers

```
1: function PRUNING($\gamma, \{f_{mt}\}_{m=1}^M$)
2:     $f_{\max} := \max\{f_{mt}, m = 1, \cdots, M\}$
3:     $f_{\min} := \min\{f_{mt}, m = 1, \cdots, M\}$
4:     $f_{\text{median}} := \text{median}\{f_{mt}, m = 1, \cdots, M\}$
5:     if $f_{\max} > \gamma * f_{\text{median}}$ then
6:         remove $f_{\max}$ at this timestep
7:     else if $f_{\min} < (1/\gamma) * f_{\text{median}}$ then
8:         remove $f_{\min}$ at this timestep
```

The pruning (algorithm 1) takes a threshold γ as input. It then discards predicted values that are outside of an interval about the median with size proportional to γ. This pruning scheme is similar to scoring rules used in many sports. For example in synchronized swimming, the highest score and the lowest score are cancelled, and the team's final score is based on the average of the remaining. In our experiments, we show the ensemble predictions are much more robust with the help of pruning.

B. Ensemble Model

The model we propose exploits possible temporally correlated prediction errors. Denote $\bar{f}_t$ the consensus forecast at time t. Recall that a new batch of measurements is provided every l steps. Let t_v (v for verification) be the time at which y_t is provided. Define the error-correction term

$$c_t := \frac{\sum_{t'=0}^{T'-1} w(t';\theta)\left(y_{t_v-l-t'} - \bar{f}_{t_v-l-t'}\right)}{\sum_{t'=0}^{T'-1} w(t';\theta)}, \qquad (19)$$

where T' is the number of time-steps used to compute c_t from historical predictions and observations. In equation (19), the *decay-function* $w(\cdot;\theta)$ is a monotonically non-increasing function in the first argument, which down-weights the difference $y_\tau - \bar{f}_\tau$ for order τ. θ is the *decay-rate* hyperparameter which controls how fast $w(\cdot;\theta)$ decreases. There are many decay-functions proposed in the literature, for example, the exponential decay defined by $w_{\exp}(\tau;\theta) := \exp(-\tau\theta)$ and the polynomial decay $w_{\text{poly}}(\tau;\theta) := (1+\tau)^{-\theta}$ [37]. For both decay-functions, setting $\theta = 1$ is equivalent to placing equal weights for all samples, whereas larger θ discriminate against older ones. The proposed ensemble model is parametrized by

$$\bar{f}_t = \alpha c_t + \sum_{m=1}^{M} \beta_m f_{mt}. \qquad (20)$$

c_t is a removing average of differences between actual flow values and ensemble predictions from the most recent T' observations, which serves as a error correction term. $\alpha, \{\beta_m\}$ are parameters to be optimized in our model. We proposed the **T**ime **D**ecay **E**rror-**C**orrection Ensemble to learn the model parameters $\alpha, \{\beta_m\}$

$$\begin{aligned}&\min_{\alpha,\{\beta_m\}} \sum_{t=0}^{T-1} w(t;\theta)\left(y_{\hat{t}-1-t} - \alpha c_{\hat{t}-1-t} - \sum_{m=1}^{M}\beta_m f_{m\hat{t}-1-t}\right)^2 \\ &+\lambda\left(\sum_m \beta_m^2 \widehat{\mathrm{var}}(f_m) + \sum_m \sum_{m'\neq m} \beta_m\beta_{m'}\widehat{\mathrm{cov}}(f_m, f_{m'})\right) \\ &\text{subject to } \sum_{m=1}^{M}\beta_m = 1, \quad \beta_m \geq 0 \;\; \forall m, \quad L \leq \alpha \leq U. \end{aligned} \tag{TDEC}$$

Here T is another user-given hyperparameter to control the number of training samples supplied to the ensemble model. The loss function in TDEC also weights the training samples by a decay term $w(\cdot;\theta)$. The minimization spells more on the loss due to recent data. The term involving λ in the objective is a regularizer. Intuitively, we are seeking for $\{\beta_m\}$ to balance between the weighted $l2$ loss and variance of the ensemble model. $\widehat{\mathrm{cov}}(f_m, f_{m'})$ is the estimated covariance between model m and m'. The choice of $\widehat{\mathrm{cov}}$ is important, but accurate estimation of the covariance is difficult. We described the principle behind the covariance-regularizer and our choice of $\widehat{\mathrm{cov}}$ in the section III-C. The bounds L and U prevent overfitting by the error-correction term. The procedure to select the hyperparameters θ and λ, and the number of time-steps T' in the error-correction term is discussed in section IV.

C. Bias-Variance-Covariance Decomposition

In this section, we explain the intuition behind the covariance-regularizer in TDEC. The goal of statistical learning is to select a function $\hat{f}$ to minimize the expected generalization error of a loss function L,

$$\min_{\hat{f}} \mathbb{E}\left(L(y, \hat{f}(\mathbf{x}))\right).$$

The expectation here is averaged over all possible randomness, including the unknown data distribution $(y, \mathbf{x})$ and random training set $\mathcal{T}$. In regression problems, when $L(y, \hat{f}(\mathbf{x})) = (y - \hat{f}(\mathbf{x}))^2$, the generalization error conditional on an input $\mathbf{x}$ could be decomposed by

$$\begin{aligned}&\mathbb{E}_{\epsilon,\mathcal{T}}\left((y_t - \hat{f}(\mathbf{x}))^2\right) \\ =&\sigma_\epsilon^2 + \left(\mathbb{E}_{\epsilon,\mathcal{T}}\hat{f}(\mathbf{x}) - f(\mathbf{x})\right)^2 + \mathbb{E}_{\epsilon,\mathcal{T}}\left(\hat{f}(\mathbf{x}) - \mathbb{E}_{\epsilon,\mathcal{T}}(\hat{f}(\mathbf{x}))\right)^2 \\ =&\sigma_\epsilon^2 + \mathrm{bias}^2(\hat{f}(\mathbf{x})) + \mathrm{var}(\hat{f}(\mathbf{x})),\end{aligned} \tag{21}$$

where subscripts under the expectation operator denote the random variables. This is called the bias-variance decomposition [38] and it holds for all data distributions and estimated models $\hat{f}$. For an ensemble model given by $\bar{f} = \sum_m \beta_m f_m$, the variance term reduces to

$$\mathrm{var}(\bar{f}(\mathbf{x})) = \sum_m \beta_m^2 \,\mathrm{var}(f_m) + \sum_m \sum_{m'\neq m} \beta_m\beta_{m'}\,\mathrm{cov}(f_m, f_{m'}). \tag{22}$$

Therefore the purpose of the regularizer in TDEC is to strike the right balance between the bias and variance of the ensemble model and achieve a lower expected generalization error. Since we do not know about the true data generating distribution $(y, \mathbf{x})$, we replaced the minimization of expected $l2$ error by empirical weighted $l2$ error from the past T time steps in TDEC. Similarly, we need to estimate $\mathbb{E}_{\mathbf{x}}\left(\mathrm{var}\,\bar{f}(\mathbf{x})\right)$. Recall that the $\mathrm{cov}(f_m(\mathbf{x}), f_{m'}(\mathbf{x}))$ considers randomness in the training set which produced f_m. Therefore, a straightforward estimation of the covariance between base learners is to retrain the model using *different training data* and compute the sample covariance of predictions for each $\mathbf{x}$. However, this approach is computationally expensive. Note that we do not consider the error-correction term as a predictor and the coefficient α does not enter covariance-regularizer and the sum-to-one equality constraint.

D. Covariance Matrix

Based on the above discussion, we now describe the form of the estimated covariance matrix used in this paper. Let $\hat{t}$ be the current time,

$$\widehat{\mathrm{cov}}(f_m, f_{m'}) = \frac{\sum_{t=0}^{T-1} w(t;\theta)(f_{m\hat{t}-1-t} - \mu_m)(f_{m'\hat{t}-1-t} - \mu_{m'})}{\sum_{t=0}^{T-1} w(t;\theta)}, \tag{23}$$

where μ_m is an extension of the definition of simple mean, defined by

$$\mu_m = \frac{\sum_{t=0}^{T-1} w(t;\theta) f_{m\hat{t}-1-t}}{\sum_{t=0}^{T-1} w(t;\theta)}. \tag{24}$$

This definition of $\widehat{\mathrm{cov}}$ takes time-stamps into account and reduces the influence of older predictions. We define the estimated covariance matrix $\widehat{\Sigma} \in \mathbb{R}^{M\times M}$ as

$$\widehat{\Sigma}_{mm'} = \widehat{\mathrm{cov}}(f_m, f_{m'}). \tag{25}$$

$\widehat{\Sigma}$ is symmetric and positive semi-definite. Note that the covariance (and variance) function in TDEC is not restricted to be the one in equation (23). In general, we expect that better approximations to $\mathbb{E}_{\mathbf{x}}\left[\mathrm{cov}(f_m(\mathbf{x}), f_{m'}(\mathbf{x}))\right]$ may serve as more effective regularizers.

E. Optimization

In this section, we demonstrate how to solve TDEC via a transformation into a convex quadratic programming problem.

Define

$$\mathbf{w} := [\alpha, \beta_1, \cdots, \beta_M]^T \in \mathbb{R}^{M+1},$$

$$\mathbf{y} := [y_{\hat{t}-T}, y_{\hat{t}-T+1}, \cdots, y_{\hat{t}-1}]^T \in \mathbb{R}^T,$$

$$P := \begin{pmatrix} c_{\hat{t}-T} & f_{1\hat{t}-T} & \cdots & f_{M\hat{t}-T} \\ c_{\hat{t}-T+1} & f_{1\hat{t}-T+1} & \cdots & f_{M\hat{t}-T+1} \\ \vdots & \vdots & \vdots & \vdots \\ c_{\hat{t}-1} & f_{1\hat{t}-1} & \cdots & f_{M\hat{t}-1} \end{pmatrix}$$

$$\Lambda := \mathrm{diag}\left(w(T-1;\theta), \cdots, w(0;\theta)\right) \in \mathbb{R}^{T\times T}$$

$$S := \begin{pmatrix} 0 & \mathbf{0}_M^T \\ \mathbf{0}_M & \widehat{\Sigma} \end{pmatrix} \in \mathbb{R}^{(M+1)\times(M+1)},$$

where $\mathbf{0}_M \in \mathbb{R}^M$, $\widehat{\Sigma}_{mm'} := \widehat{\mathrm{cov}}(f_m, f_{m'})$. Also, let $\mathbf{1}_M$ be a vector of ones, and $\mathbf{I}_{M\times M}$ be the identity matrix of size

M. TDEC can then be written as a quadratic programming problem

$$\begin{aligned} \min_{\mathbf{w}} \, & (\mathbf{y} - P\mathbf{w})^T \Lambda (\mathbf{y} - P\mathbf{w}) + \lambda \mathbf{w}^T S \mathbf{w} \\ & \begin{bmatrix} 0 & \mathbf{1}_M^T \end{bmatrix} \mathbf{w} = 1 \\ & \begin{bmatrix} \mathbf{0}_M & \mathbf{I}_{M \times M} \end{bmatrix} \mathbf{w} \geq \mathbf{0}_{(M+1)} \\ & \begin{bmatrix} 1 & \mathbf{0}_M^T \end{bmatrix} \mathbf{w} \geq L \\ & \begin{bmatrix} 1 & \mathbf{0}_M^T \end{bmatrix} \mathbf{w} \leq U. \end{aligned} \tag{TDEC-QP}$$

Therefore, solving TDEC-QP is not a more difficult problem than stack regression.

IV. Experiments

To assess the performance of consensus ensemble prediction, we test the base methods and ensemble methods on arterial traffic flow. We believe arterial flow forecasting is a more difficult task than freeway flow forecasting, because of its more variable road conditions.

A. Data Description

We conducted experiments on traffic flow data collected from arterial sensors in Arcadia, CA in 2015. The raw data is processed into traffic flow time series whose consecutive measurements are separated by a fifteen minute interval, measured in number of cars per hour.

B. Experimental Procedure

Traffic control centers receive flow measurements periodically. Using historical data and sensor readings from the recent past, traffic operators wish to make multi-step traffic forecasts into the near future. In our experiments, we considered the case where traffic flow measurements are sent to the control center every hour, and forecasts for the following hour is desired. Hence, each prediction consists of four time-steps separated by intervals of fifteen minutes. For instance, using historical data and today's traffic flow up to 7 AM, the flow predictions at 7:15 AM, 7:30 AM, 7:45 AM and 8:00 AM are computed. After that, the true flow at these times are "observed" by the algorithms, and predictions for the next hour are made. Using the notation from the previous sections, the verification and prediction horizon is $l = 4$ in this setting. Notice that under this rolling procedure, we never use flows after the forecasting time-step in the prediction algorithms.

In the proposed consensus prediction system, there are two levels of training required - one for the base methods and additional training for the ensemble TDEC. For each prediction step t, we first train the base models and make base predictions for this step. Next, the pruning procedure (Algorithm 1) removes anomalous base forecasts. After that we query historical flow observations and the past T base predictions to formulate problem TDEC. Optimal solutions from TDEC-QP are then used to construct the consensus forecast. The base model parameters and ensemble parameters $\alpha, \{\beta_m\}_{m=1}^M$ are updated in every time-step.

We use the following metrics for comparing the performance of different base methods and the consensus method. The absolute error of method m at time t is defined as

$$\texttt{AE}(t) = |y_t - f_{mt}|$$

which quantifies for the magnitude of the prediction error. The mean absolute error (`MAE`) is the mean of `AE` in all tested time steps. In addition to the mean, we are interested in the standard deviation

$$\texttt{StdAE} = \sqrt{\frac{\sum_t (\texttt{AE}(t) - \texttt{MAE})^2}{\text{number of steps evaluated} - 1}}.$$

`StdAE` provides a view on the robustness of a model. When the `MAE` of two models are close, the one with lower `StdAE` is preferred.

C. Automatic Hyperparameter Search

We now describe the selection of hyperparameters in TDEC. In general, hyperparameter optimization is expensive and requires repeated model evaluation. We use two hyperparameter search strategies, grid search and random search. Given the traffic flow time series, let $t_{\mathcal{H}}$ be a cut-off time such that measurements before $t_{\mathcal{H}}$ are used for constructing the hyperparameter validation set $\mathcal{V}$, and measurements after $t_{\mathcal{H}}$ are use for testing. Note that with the rolling training process described in section IV-B, there is a *cold-start* period, which is the minimum number of time-steps needed to train the base models, plus an additional T steps needed to verify the base predictions and build the consensus model.

In grid search, a set of candidate hyperparameters are specified. Denote $\mathcal{H}(\theta)$ the set of decay rate hyperparameters, $\mathcal{H}(\lambda)$ the set of regularization hyperparameters in TDEC, and $\mathcal{H}(T')$ the number of time-steps used to compute the error-correction term in equation (19). In our experiments, we let

$$\begin{aligned} \mathcal{H}_{\text{grid}} &= \mathcal{H}(\theta) \times \mathcal{H}(\lambda) \times \mathcal{H}(T'), \\ \mathcal{H}(\theta) &= \{0, 0.05, 0.1, 0.15\}, \\ \mathcal{H}(\lambda) &= \{0, 1, 3, 5\}, \\ \mathcal{H}(T') &= \{8, 40, 80\}. \end{aligned} \tag{26}$$

Grid search evaluates all configurations in $\mathcal{H}_{\text{grid}}$. The ensemble model enumerates all possible hyperparameter configurations in the grid search space $\mathcal{H}_{\text{grid}}$ and uses the exponential decay-function $w_{\exp}(t; \theta) = \exp(-t\theta)$. For each hyperparameter choice, the `MAE` on validation set $\mathcal{V}$ is recorded. The one that achieves minimum `MAE` on $\mathcal{V}$ is chosen. Hyperparameter grid search in high dimension is computationally expensive. For N hyperparameters and each with c possible values, grid search procedure results in $O(|\mathcal{V}|c^N)$ model evaluations. Random search is an alternative to grid search, which does not enumerate all possible hyperparameter settings. Rather, each pass over the validation set randomly selects a configuration from the search space. Many researchers have suggested that random search is very competitive in high dimension, due to the *curse of dimensionality* [39]. Recall that the decay-function appears in the error-correction term (equation (19)), in the weighted $l2$ loss term of TDEC, and in the estimated

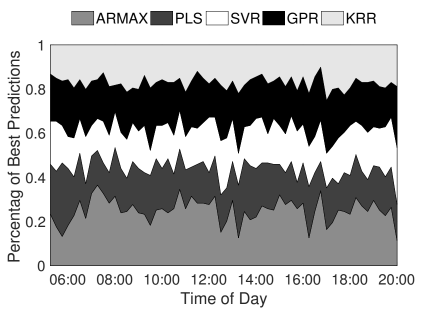

Fig. 3. Area plot of percentage of best predictions by each method. Area of the strips are proportional to the percentage.

covariance matrix (equation (23)). The dimension of hyperparameters are expanded if we let each component adapt its own parametrization of decay-function $w(\cdot;\theta)$ and decay-rate θ. In addition, the lower bound L and upper bound U for the error-correction coefficient could also be tuned. We apply random search for hyperparameters from the following search space:

$$\begin{aligned}\mathcal{H}_{\text{random}} &= \mathcal{H}(w_{\text{loss}}) \times \mathcal{H}(w_{\text{ec}}) \times \mathcal{H}(w_{\text{cov}}) \\ &\quad \times \mathcal{H}(\theta_{\text{loss}}) \times \mathcal{H}(\theta_{\text{ec}}) \times \mathcal{H}(\theta_{\text{cov}}) \\ &\quad \times \mathcal{H}(\lambda) \times \mathcal{H}(T') \times \mathcal{H}(L,U), \\ \mathcal{H}(w_{\text{loss}}), \mathcal{H}(w_{\text{ec}}), \mathcal{H}(w_{\text{cov}}) &= \{\exp(-t\theta), (1+t)^{-\theta}\} \\ \mathcal{H}(\theta_{\text{loss}}), \mathcal{H}(\theta_{\text{ec}}), \mathcal{H}(\theta_{\text{cov}}) &= \{0, 0.05, 0.1, 0.15\}, \\ \mathcal{H}(\lambda) &= \{0, 1, 3, 5\}, \\ \mathcal{H}(T') &= \{8, 40, 80\}, \\ \mathcal{H}(L,U) &= \{L, U \in [0,1], L \leq U\}. \end{aligned} \tag{27}$$

Similar to grid search, the hyperparameter configurations producing the lowest `MAE` from the random search will be used in the testing set. In our experiments, we make 50 uniform random draws from $\mathcal{H}_{\text{random}}$. Note that the embedding dimension p for $\mathbf{x}_t$, and the training set size $\hat{T}$, T may also be tuned. For comparison in later sections, we set $p = 48$ (12 hours), $\hat{T} = 120 * 24 * 4$ (120 days), $T = 20 * 4$ (80 hours).

We use the function `fitrsvm` for SVM and `fitrgp` for GPR from Matlab Machine Learning and Statistics toolbox with their heuristic default values for the hyperparameters [40], [41]. We implement PLS and KRR and apply similar default heuristics for hyperparameter configuration (see supplementary document). The tables and figures reported in the paper are obtained from base methods with their default hyperparameters. Although the hyperparameters for each base method could be fine-tuned with grid search, we do not find it impactful on the prediction quality of ensemble models substantially. The complete experiment details with tuning can be found in the supplementary file.

D. Overview of Results

One of the motivations for developing a consensus method is that it is unlikely that a single base prediction could consistently outperform others all the time. We verify this hypothesis by comparing the absolute error obtained by the base methods across different time and on different detectors. For each prediction step, a method is marked as the winner if it achieves the lowest absolute error. We compare the percentage of testing days achieving the lowest absolute error by each method at different time. The result is visualized by the area plot in figure 3, in which each method is represented by a shaded strip. The width and area of each strip is proportional to the percentage of testing days won by the respective method. From figure 3, there is no single strip whose area dominates the plot. In addition, the strips are in zig-zag shapes. It is not easy to identify a base method that consistently won over a continuous portion of the day. This motivates us to study a model combination approach for flow prediction. Figure 4 displays a showcase of the results on three consecutive days randomly selected on a detector. Despite wide ranges of base predictions around the morning rush hours, the ensemble predictions TDEC closely aligned with the actual value of flows.

E. Baseline and Experimental Goals

Model combination has been studied in many domains, for example, machine learning [17], [38], [42] and econometrics [43]–[45]. Despite a large body of literature in this area, a common empirical observation in many areas is that the simple average combination which assigns equal weights for the base methods often outperforms complicate combination schemes [44], [46]. This is known as the *"forecast combination puzzle"* in the statistical forecasting literature [44], [46]. Some authors suggested that the weights learned from historical data are unstable and unreliable, as a consequence of overfitting [45], [46]. Therefore, we use simple average combination as a baseline to compare with. A second baseline is the base method achieving the lowest `MAE` and `StdAE` for each detector. We are interested in 1. examining whether the proposed ensemble prediction improves over the simple average combination and best base method. 2. studying which components in the proposed ensemble model contribute to the performance improvement or decline. We also compare our method with two multi-model combination methods in traffic flow forecasting literature in section IV-H. Note that we do not compare the performance of TDEC with bagging and random forest [19], [20], since our study focus on multi-model combination schemes, whereas random forest uses the same "weak learner" together with data sub-sampling strategy.

F. Effect of Pruning

In the proposed ensemble system, the pruning scheme may be applied prior to solving the optimization problem TDEC in each time-step. We run the rolling experiments to compare the performance with and without the pruning step. The motivation of unsupervised pruning step is to safeguard the procedure against unrealistic base predictions. Table II displays the mean absolute error and standard deviation for the following methods: • TDEC-rs, model TDEC where the hyperparameters are automatically selected by the random search procedure outlined in section IV-C • TDEC-gs such that the hyperparameters are set by grid search • SR, Stack regression [18] applied to the rolling experiment setting •

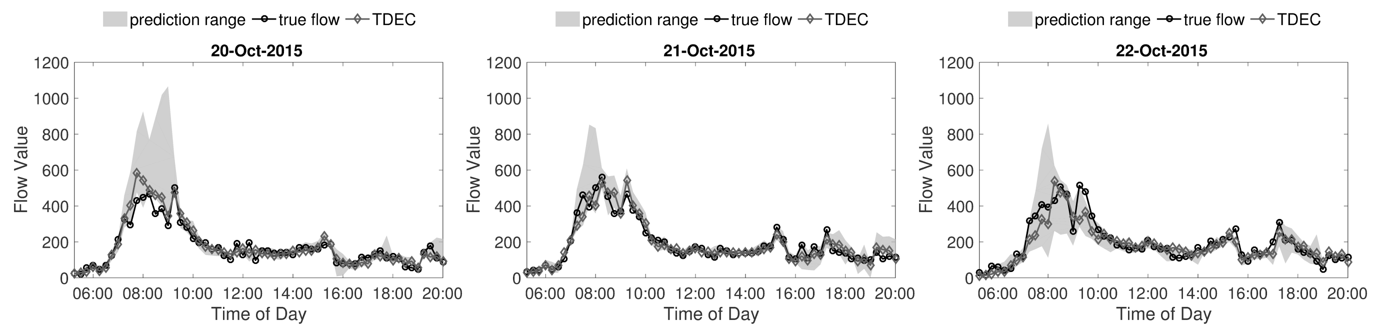

Fig. 4. Predicted flow values and the true flow in three consecutive days. Despite a wide range of base predictions, the ensemble is closer to the true flow.

AVG, simple average combination of the base methods. In addition, the sub-columns marked by $\gamma = 5$ indicate threshold of the pruning scheme, $\gamma = \infty$ indicate no pruning.

For each detector, the lowest MAE is achieved either by TDEC-gs or TDEC-rs. On 12 out of 14 detectors, TDEC (-gs and -rs combined) obtained the smallest StdAE. Table III lists the percentage reductions in MAE and StdAE with different values of γ, compared to no pruning for each method, averaged from all tested detectors. A positive percentage change denotes an improvement, a negative percentage implies decline in performance. The pruning criterion is designed to be less sensitive with larger γ. Note that there are improvements to the MAE and StdAE with all three values of γ.

In table II, TDEC-rs and TDEC-gs with $\gamma = 5$ outperforms simple average combination with $\gamma = 5$ in almost all the detectors. However, if the pruning scheme is removed, TDEC-gs with $\gamma = \infty$ produced higher standard deviation than AVG with $\gamma = \infty$ in 5 of 14 cases. Therefore, the pruning scheme is necessary to produce stable results and consistent improvements over simple average combination. In addition, TDEC (-rs and -gs combined) with the pruning step achieved lower mean and standard deviation of absolute error on all detectors compared to the best base method. This shows that the proposed ensemble model could indeed be used to integrate existing base methods.

G. Effect of Hyperparameters

The upper panel in Table IV displays the average percentage reductions in MAE and StdAE obtained by each ensemble method combined with the pruning scheme compared to the best base model on all detectors, the lower panel shows the maximum improvement of MAE and StdAE among the fourteen tested detectors. A higher value indicates greater improvements. On average, TDEC, with hyperparameters selected either by random search (-rs) or grid search (-gs), outperformed stack regression (SR) and simple average combination (AVG). The improvements by the ensemble methods are less pronounced when $\gamma = 10$. In addition, TDEC with random search resulted in greater improvements over the best base model than TDEC with grid search scheme, likely due to the enlarged hyperparameter search space we allowed. In our experiment, we use 50 random draws from the search space $\mathcal{H}_{\text{random}}$. As a result, this requires $50|\mathcal{V}|$

TABLE II
COMPARISON OF CONSENSUS AND BASE PREDICTORS. LOWER VALUES ARE BETTER FOR BOTH METRICS. BEST VALUES FOR EACH DETECTOR ARE SHOWN IN BOLD FONT.

Mean Absolute Error MAE

ID	TDEC-rs	TDEC-gs		SR		AVG		Ridge		Lasso		Best Base	
–	$\gamma = 5$	$\gamma = 5$	$\gamma = \infty$	$\gamma = 5$	$\gamma = \infty$	$\gamma = 5$	$\gamma = \infty$	$\gamma = 5$	$\gamma = \infty$	$\gamma = 5$	$\gamma = \infty$		
1	**126.6**	126.8	128.2	127.9	129.1	129.2	141.3	131.2	133.3	152	154.3	135.2	KRR
2	**55.8**	56.2	60.3	56.7	60.9	57.1	64.8	59.2	62.4	73.1	75.5	57.1	KRR
3	**42.4**	**42.4**	46.7	42.8	47.5	44.1	46.5	43.1	47.8	47.2	51.5	45.7	KRR
4	**45.2**	**45.2**	47.5	45.5	47.7	46.7	59	47.3	49.7	63.4	66.5	**45.2**	KRR
5	**45.1**	**45.1**	47.3	45.2	47.7	45.4	51.8	46.3	48.7	62.7	64.8	45.1	SVR
6	35.2	35.3	**34.6**	35.9	34.8	39.9	39.8	36.4	35.9	60.2	60.2	42	KRR
7	35	35.1	**34.8**	35.2	34.9	36.1	35.9	35.8	35.7	42.3	42.1	37.4	KRR
8	**34.3**	34.5	34.5	**34.3**	**34.3**	35.2	35.4	35.5	35.5	37.1	37.2	36.6	KRR
9	**29.4**	29.5	31.5	30.1	31.9	30.3	35.2	30.7	32.7	33.5	34.9	30.7	KRR
10	**29.2**	**29.2**	31.2	29.3	30.9	29.5	31.1	30.2	32.1	31.1	32.6	30.3	SVR
11	**17.3**	17.4	18.2	17.4	18.1	**17.3**	17.9	17.7	18.3	23.6	24.2	17.6	SVR
12	**10.4**	**10.4**	10.5	**10.4**	10.5	**10.4**	10.5	10.6	10.7	12.7	12.8	10.8	SVR
13	**9.7**	**9.7**	9.8	**9.7**	9.8	9.8	9.8	9.9	10	12.2	12.3	9.9	GPR
14	**8.4**	8.5	**8.4**	8.5	**8.4**	8.6	8.6	8.6	8.5	9.9	9.8	9	SVR

Standard Deviation of Absolute Error StdAE

ID	TDEC-rs	TDEC-gs		SR		AVG		Ridge		Lasso		Best Base	
–	$\gamma = 5$	$\gamma = 5$	$\gamma = \infty$	$\gamma = 5$	$\gamma = \infty$	$\gamma = 5$	$\gamma = \infty$	$\gamma = 5$	$\gamma = \infty$	$\gamma = 5$	$\gamma = \infty$		
1	**131.2**	131.6	146.6	133.4	147.5	138.2	562.4	135.2	155	165.3	177.5	146.8	KRR
2	**54.3**	55	153.8	55.6	160.1	56.9	157.3	56.7	133.6	66.1	111.3	55.3	KRR
3	38.7	**38.4**	264.2	39.2	298.9	41.5	156.5	39.5	297.7	41.8	257.3	44.3	KRR
4	**42.1**	43.5	67.4	44	62.9	45.6	228.3	45.9	68.6	56.5	66.7	44	KRR
5	39.2	**39.1**	86.6	39.3	110.8	39.6	355.9	40	112.5	51.6	109.1	39.3	SVR
6	45.6	46.3	45.3	47.4	**45.2**	52.6	49.1	47	46.2	56.6	56.7	54.7	KRR
7	32	32.2	**31.8**	32.6	**31.8**	33.8	32.6	33.5	33	38.6	37.9	35.7	KRR
8	32.6	**32.3**	32.4	32.6	32.6	34.7	35.3	33.4	35.5	33.4	33.4	39	KRR
9	**32.5**	32.8	63.2	33.5	62.2	33.3	155.2	34.2	63.5	36.6	53.5	33.6	KRR
10	**32.9**	33.2	142.1	33.3	134.2	**32.9**	120.6	34.3	131.3	34.4	112.6	33.6	SVR
11	15.9	16.2	52	15.9	45.5	**15.6**	36.1	16.3	38.7	20.2	39.6	16.2	SVR
12	**8.6**	8.7	10.3	**8.6**	10.4	8.7	9	8.8	10.5	10.6	11.7	9.2	SVR
13	**7.8**	7.9	8.7	7.9	8.5	7.9	8.5	7.9	8.5	9.7	10.1	8	GPR
14	**6.9**	7	**6.9**	7.1	**6.9**	7.1	**6.9**	7.1	7	8.1	8	7.3	SVR

TABLE III
AVERAGE PERCENTAGE REDUCTIONS IN MAE AND StdAE OF ENSEMBLE METHODS WITH PRUNING, COMPARED TO NO PRUNING ($\gamma = \infty$).

	MAE				StdAE			
γ	TDEC-rs	TDEC-gs	SR	AVG	TDEC-rs	TDEC-gs	SR	AVG
3	3.3	3	2.9	6.1	33.2	33.2	32.7	42.1
5	3.4	3	2.8	5.8	33.3	33.1	32.6	42.1
10	2.9	2.5	2.5	5.1	31.8	31.8	31.5	41.5

model evaluations of TDEC, where $|\mathcal{V}|$ is the number of time-steps in the hyperparameter validation set $\mathcal{V}$. Note that grid search from $\mathcal{H}_{\text{grid}}$ requires $|\mathcal{V}| \times 3^2 \times 4^2$ model evaluations. Hence, random search is more efficient and effective than grid search for selecting hyperparameters in our experiments. This observation is consistent with others in the literature on hyperparameter optimization. Some theoretical analysis

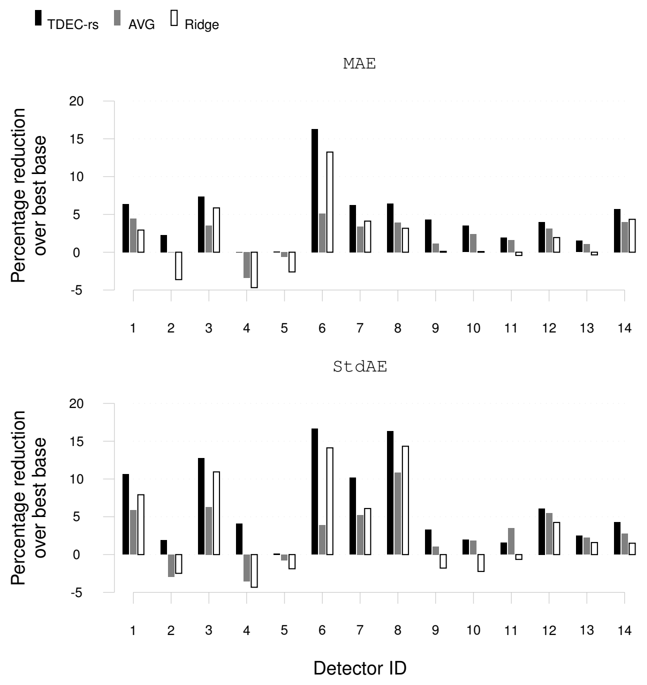

Fig. 5. Percentage Reduction in MAE and StdAE of ensemble methods with pruning ($\gamma = 5$), compared to the best base method. Higher values are better for both metrics.

TABLE IV
AVERAGE AND MAXIMUM PERCENTAGE REDUCTIONS IN MAE AND STDAE OF ENSEMBLE METHODS WITH PRUNING, COMPARED TO THE BEST BASE METHOD. HIGHER VALUES ARE BETTER FOR BOTH METRICS. BEST VALUES FOR EACH DETECTOR ARE SHOWN IN BOLD FONT.

Average Percentage Reduction								
	MAE				StdAE			
γ	TDEC-rs	TDEC-gs	SR	AVG	TDEC-rs	TDEC-gs	SR	AVG
3	**4.6**	4.5	4	2.5	**6.5**	6	5.3	3.4
5	**4.7**	4.4	3.9	2.1	**6.6**	5.9	5.1	3
10	**4.3**	4	3.6	1.3	**4.2**	3.3	3.2	0.5
Maximum Percentage Reduction								
	MAE				StdAE			
γ	TDEC-rs	TDEC-gs	SR	AVG	TDEC-rs	TDEC-gs	SR	AVG
3	15.6	**15.9**	14.5	6.5	16.7	**16.9**	16.2	10.6
5	**16.3**	15.9	14.6	5.1	16.3	**17**	16.3	10.8
10	**16.3**	16.2	15.3	8.7	**15.9**	17	16.4	9.4

suggests that because models typically have non-homogeneous sensitivity with respect to different hyperparameters and data distributions, grid search spends too much time exploiting less sensitive hyperparameters [39].

H. Compare with Other Multi-Model Methods

We compare our proposed method with other multi-model combination strategies for traffic forecasting to further evaluate its performance. The Ridge Regression Ensemble and Lasso Ensemble were proposed by Li et al. [21] for freeway traffic estimation. This work shares the same motivation with ours, that "any models existing are imperfect and have their own strengths and weakness". Using the same notations from section III-E, the Ridge Regression Ensemble solves

$$\min_{\mathbf{w}} \|\mathbf{y} - P\mathbf{w}\|_2^2 + \lambda_{\text{ridge}} \|\mathbf{w}\|_2^2, \tag{28}$$

and Lasso Ensemble solves

$$\min_{\mathbf{w}} \|\mathbf{y} - P\mathbf{w}\|_2^2 + \lambda_{\text{lasso}} \|\mathbf{w}\|_1, \tag{29}$$

to obtain the ensemble weights. In Ridge Regression Ensemble, the penalty term $\lambda_{\text{ridge}} \|\mathbf{w}\|_2^2$ forces shrinkage of the solution to avoid overfitting. The l_1-norm penalty in Lasso Ensemble produces a sparse solution, hence fewer base methods will be selected in the ensemble than the one obtained from a least square fitting. Comparing our method TDEC-QP and equation (28) in Ridge Regression Ensemble, the covariance penalty term in TDEC-QP could be viewed as a generalization to the euclidean norm penalty. Also, it is noteworthy to point out that neither Ridge Regression Ensemble nor Lasso Ensemble requires the weights to be summed-to-one and non-negative. We run the same base methods and compute the ensemble prediction with equation (28) and equation (29), with the regularization parameter λ_{ridge} and λ_{lasso} selected via grid search from $\{0.1, 1, 3, 5\}$ on the validation data. Both ensemble methods are tested with and without applying the pruning procedure (Algorithm 1). The MAE and StdAE for each detector under the Ridge Regression Ensemble and Lasso Ensemble are listed in Table II. The Lasso Ensemble predictions, somewhat surprisingly, underperformed the best base method in all detectors; however, this result is consistent with the one reported in [21], in which the authors found Lasso Ensemble improves freeway traffic density (in vehicles per kilometer per lane) estimate but worsen flow rate prediction in many cases. The relative improvements of TDEC-QP, simple averaged combination, and Ridge Regression Ensemble over the best base method for each detector are displayed in figure 5. For each bar, positive value denotes improvement and higher is better, vice versa. Our method outperforms Ridge Regression Ensemble and simple averaging in almost all the detectors in both MAE and StdAE. Moreover, TDEC-QP offers improvement over the best base method even when the other two multi-model methods fail (detector 2, 4, 9, 10, 11). In addition, simple averaging performs better than Ridge Regression Ensemble in more than half of the cases. This observation confirms simple average combination is indeed a very strong baseline [44], [46].

I. Discussion on Computational Time

There are two major computational stages when running the system proposed in this paper. The first stage is to train the base models and generate predictions from each of them. The second stage is to obtain the ensemble parameters via solving a convex quadratic programming problem TDEC-QP. The first stage is common for most multi-model based ensemble methods, for example, Ridge Regression Ensemble [21] discussed in the previous section. Table V shows the running time in seconds spent by different components of our system in a one-hour-ahead traffic forecast scenario. The numbers reported are the average from ten runs. In our problem setting (section IV-B), the base models and ensemble model TDEC-QP are refitted every hour. Four predictions spanning one hour are produced after the model fitting. Solving convex quadratic programming based problem is much more efficient than parameter optimization in neural network, which makes our ensemble method computationally more feasible than neural network-based ensemble model for online traffic flow forecast [19], [22]. In our Matlab implementation, the total time needed to finish an ensemble four-step-prediction is in the order of seconds (table V). The running time for solving TDEC-QP is 0.02 seconds on average. Therefore in real operation, the computational time attributed to model fitting and predictions is only a tiny fraction of the one hour time budget. Obtaining a solution for Ridge Regression Ensemble takes only 0.003 seconds on average, since there is a closed-form formula available. A non-smooth convex optimization problem needs to be solved for Lasso Ensemble. On average, the running time with Matlab built-in `lasso` function takes 0.02 seconds.

TABLE V
TEN-RUN-AVERAGED RUNNING TIME (SECONDS) SPENT BY DIFFERENT COMPONENTS OF THE SYSTEM FOR ONE-HOUR-AHEAD FORECAST, MODEL FITTING AND PREDICTION COMBINED. RESULTS MEASURED ON A INTEL I5 2.40 GHZ DUAL CORE PROCESSOR.

ARMAX	PLS	SVM	KRR	GPR	TDEC-QP	Total
0.09	0.32	0.42	0.38	1.19	0.02	2.42

V. Conclusion

We addressed an important practical problem in traffic flow prediction: how to combine the advantage of multiple flow forecasting models to yield a result that is at least as accurate and stable as the best one. An ensemble learning model was proposed to this end. Our method was based on three core ideas: 1. learning from mistakes in the recent past, 2. balancing model diversity and accuracy, and 3. applying a pruning scheme to remove extreme forecasts. In addition, we explained how to tailor some widely used machine learning and statistical models for traffic flow prediction. On the tested arterial traffic sensors, our proposed ensemble model achieved as much as 16.3% and 17% improvements, and on average 4.7% and 6.6% improvements, respectively in mean and standard deviation of absolute error over the best base model. We consistently outperformed two recently published ensemble prediction schemes based on Ridge Regression and Lasso, and produced more accurate and robust predictions even in scenarios which the other ensemble methods backfire. In addition, our framework does not have restrictions on the type of sub-models used. Our future work will extend our methodology to network-scale traffic flow prediction.

Acknowledgment

This work is supported in part by the Office of Science of the U.S. Department of Energy under contract No. DE-AC02-05CH11231. Many thanks to the team at the U.C. Berkeley Connected Corridors Program for providing the data used in this study. This work is also supported by the US National Science Foundation grant ACI-1253881 and a Penn State College of Engineering seed grant.

References

[1] U. S. Department of Transportation, "Integrated Corridor Management (ICM)," https://www.its.dot.gov/research_archives/icms/index.htm, last accessed Sep 2017.

[2] C. F. Daganzo, "The cell transmission model, part II: Network traffic," *Transp. Res. Part B: Methodological*, vol. 29, no. 2, pp. 79–93, 1995.

[3] 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, 2015.

[4] H. Nicholson and C. D. Swann, "The prediction of traffic flow volumes based on spectral analysis," *Transp. Res.*, vol. 8, no. 6, pp. 533–538, 1974.

[5] I. Okutani and Y. J. Stephanedes, "Dynamic prediction of traffic volume through Kalman filtering theory," *Transp. Res. Part B: Methodological*, vol. 18, no. 1, pp. 1–11, 1984.

[6] M. M. Hamed, H. R. Al-Masaeid, and Z. M. Bani Said, "Short-term prediction of traffic volume in urban arterials," *Journal of Transportation Engineering*, vol. 121, no. 3, pp. 249–254, 1995.

[7] M. Van Der Voort, M. Dougherty, and S. Watson, "Combining Kohonen maps with ARIMA time series models to forecast traffic flow," *Transp. Res. Part C: Emerging Technologies*, vol. 4, no. 5, pp. 307–318, 1996.

[8] S. Lee and D. Fambro, "Application of subset autoregressive integrated moving average model for short-term freeway traffic volume forecasting," *Transp. Res. Rec.*, vol. 1678, pp. 179–188, 1999.

[9] B. Ghosh, B. Basu, and M. O'Mahony, "Multivariate short-term traffic flow forecasting using time-series analysis," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 2, pp. 246–254, 2009.

[10] Y. J. Stephanedes, P. G. Michalopoulos, and R. A. Plum, "Improved estimation of traffic flow for real-time control," *Transp. Res. Rec.*, vol. 795, pp. 28–39, 1981.

[11] B. Williams, "Multivariate vehicular traffic flow prediction: Evaluation of ARIMAX modeling," *Transp. Res. Rec.*, vol. 1776, pp. 194–200, 2001.

[12] C.-J. Wu, T. Schreiter, R. Horowitz, and G. Gomes, "Traffic flow prediction using optimal autoregressive moving average with exogenous input-based predictors," *Transp. Res. Rec.*, vol. 2421, pp. 125–132, 2014.

[13] A. Pascale and M. Nicoli, "Adaptive Bayesian network for traffic flow prediction," in *Proc. IEEE Statistical Signal Processing (SSP) Workshop*, 2011.

[14] S. Coogan, C. Flores, and P. Varaiya, "Traffic predictive control from low-rank structure," *Transp. Res. Part B: Methodological*, vol. 97, pp. 1–22, 2017.

[15] *Highway Capacity Manual*. Transportation Research Board, 2010.

[16] M. Cremer and H. Keller, "A new class of dynamic methods for the identification of origin-destination flows," *Transp. Res. Part B: Methodological*, vol. 21, no. 2, pp. 117–132, 1987.

[17] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.

[18] L. Breiman, "Stacked regressions," *Machine Learning*, vol. 24, no. 1, pp. 49–64, 1996.

[19] S. Sun, "Traffic flow forecasting based on multitask ensemble learning," in *Proc. ACM/SIGEVO Summit on Genetic and Evolutionary Computation (GEC)*, 2009.

[20] Y. Hou, P. Edara, and C. Sun, "Traffic flow forecasting for urban work zones," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 4, pp. 1761–1770, 2015.

[21] L. Li, X. Chen, and L. Zhang, "Multimodel ensemble for freeway traffic state estimations," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 3, pp. 1323–1336, 2014.

[22] 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, 2009.

[23] L. Dimitriou, T. Tsekeris, and A. Stathopoulos, "Adaptive hybrid fuzzy rule-based system approach for modeling and predicting urban traffic flow," *Transp. Res. Part C: Emerging Technologies*, vol. 16, no. 5, pp. 554–573, 2008.

[24] T. L. Lai, "Recursive estimation in ARMAX models," in *New Directions in Time Series Analysis: Part II*, D. Brillinger, P. Caines, J. Geweke, E. Parzen, M. Rosenblatt, and M. S. Taqqu, Eds., 1993, pp. 263–288.

[25] S. Wold, A. Ruhe, H. Wold, and W. J. Dunn, III, "The collinearity problem in linear regression. the partial least squares (PLS) approach to generalized inverses," *Journal on Scientific and Statistical Computing*, vol. 5, no. 3, pp. 735–743, 1984.

[26] H. Drucker, C. C. Burges, L. Kauffman, A. Smola, and V. Vapnik, "Support vector regression machines," in *Proc. Advances in Neural Information Processing Systems (NIPS)*, 1996.

[27] V. Vapnik, S. E. Golowich, and A. Smola, “Support vector method for function approximation, regression estimation, and signal processing,” in *Proc. Advances in Neural Information Processing Systems (NIPS)*, 1996.
[28] Y. Zhang and Y. Xie, “Forecasting of short-term freeway volume with v-support vector machines,” *Transp. Res. Rec.*, vol. 2024, pp. 92–99, 2008.
[29] D. Wei and H. Liu, “An adaptive-margin support vector regression for short-term traffic flow forecast,” *Journal of Intelligent Transportation Systems*, vol. 17, no. 4, pp. 317–327, 2013.
[30] V. N. Vapnik, *Statistical learning theory*. Wiley New York, 1998.
[31] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2006.
[32] C.-C. Chang and C.-J. Lin, “LIBSVM: a library for support vector machines,” *Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 27:1–27:27, 2011.
[33] R.-E. Fan, P.-H. Chen, and C.-J. Lin, “Working set selection using second order information for training support vector machines,” *Journal of Machine Learning Research*, vol. 6, pp. 1889–1918, 2005.
[34] B. Schölkopf, R. Herbrich, and A. J. Smola, “A generalized representer theorem,” in *Proc. International Conference on Computational Learning Theory (COLT)*, 2001.
[35] Y. Xie, K. Zhao, Y. Sun, and D. Chen, “Gaussian processes for short-term traffic volume forecasting,” *Transp. Res. Rec.*, vol. 2165, pp. 69–78, 2010.
[36] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.
[37] G. Cormode, V. Shkapenyuk, D. Srivastava, and B. Xu, “Forward decay: A practical time decay model for streaming systems,” in *Proc. Int’l. Conf. on Data Engineering (ICDE)*, 2009.
[38] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer-Verlag New York, 2009.
[39] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.
[40] “Mathworks® documentation, fitrgp,” https://www.mathworks.com/help/stats/fitrgp.html, accessed: 2017-09-14.
[41] “Mathworks® documentation, fitrsvm,” https://www.mathworks.com/help/stats/fitrsvm.html, accessed: 2017-09-14.
[42] Z.-H. Zhou, *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC, 2012.
[43] C. W. J. Granger and R. Ramanathan, “Improved methods of combining forecasts,” *Journal of Forecasting*, vol. 3, no. 2, pp. 197–204, 1984.
[44] J. Smith and K. F. Wallis, “A simple explanation of the forecast combination puzzle,” *Oxford Bulletin of Economics and Statistics*, vol. 71, no. 3, pp. 331–355, 2009.
[45] V. Genre, G. Kenny, A. Meyler, and A. Timmermann, “Combining expert forecasts: Can anything beat the simple average?” *International Journal of Forecasting*, vol. 29, no. 1, pp. 108–121, 2013.
[46] R. T. Clemen, “Combining forecasts: A review and annotated bibliography,” *International Journal of Forecasting*, vol. 5, no. 4, pp. 559–583, 1989.

Hongyuan Zhan is a PhD candidate in the Computer Science and Engineering department at The Pennsylvania State University. He received the bachelor degree in Mathematics with honors from Penn State in 2014. His research interests are at the intersection of machine learning for streaming data, optimization, and high performance computing, with applications to transportation research, finance, and network analysis.

Gabriel Gomes is an Assistant Research Engineer with the Institute of Transportation Studies at U.C. Berkeley. Dr. Gomes’ research at ITS focuses on modeling, simulation, and control of traffic networks. He collaborates with the Connected Corridors project, which aims to develop and deploy decision support systems for urban corridors. He is the principal developer of BeATS - the Berkeley Advanced Traffic Simulation software.

Xiaoye S. Li earned Ph.D. in Computer Science from UC Berkeley, and is now a Senior Scientist at Lawrence Berkeley National Laboratory. She has worked on diverse problems in high performance scientific computations, including parallel computing, sparse matrix computations, high precision arithmetic, and combinatorial scientific computing. She has (co)authored over 95 publications, and has led and contributed to the development of several open source mathematical libraries. She is a SIAM Fellow and an ACM Senior Member.

Kamesh Madduri received the bachelor degree from the Indian Institute of Technology Madras and the PhD degree from Georgia Institute of Technology. He is an assistant professor in the Computer Science and Engineering department at The Pennsylvania State University. His research interests include high-performance computing and network science.

Alex Sim is a Senior Computing Engineer at the Lawrence Berkeley National Laboratory. His current R&D activities focus on data mining and modelling, data analysis methods, distributed resource management, and high performance data systems. He authored and co-authored more than 130 technical publications, and released a few software packages under open source license.

Kesheng Wu is a Senior Scientist at Lawrence Berkeley National Laboratory. He works extensively on data management, data analysis, and scientific computing topics. He is the developer of a number of widely used algorithms including FastBit bitmap indexes for querying large scientific datasets, Thick-Restart Lanczos (TRLan) algorithm for solving eigenvalue problems, and IDEALEM for statistical data reduction and feature extraction.