

## ORIGINAL RESEARCH

# Predicting network flows from speeds using open data and transfer learning

 Vishal Mahajan<sup>1</sup>  | Guido Cantelmo<sup>2</sup> | Raoul Rothfeld<sup>1</sup> | Constantinos Antoniou<sup>1</sup> 

<sup>1</sup>Chair of Transportation Systems Engineering, TUM School of Engineering and Design, Technical University of Munich, Munich, Germany

<sup>2</sup>Department of Technology, Management and Economics, Transport Division, Technical University of Denmark, Copenhagen, Denmark

**Correspondence**

Vishal Mahajan, Chair of Transportation Systems Engineering, Technical University of Munich, Arcisstraße 21, Munich, Germany.  
Email: vishal.mahajan@tum.de

**Funding information**

Deutsche Forschungsgemeinschaft, Grant/Award Number: 415208373

**Abstract**

Traffic flow/volume data are commonly used to calibrate and validate traffic simulation models. However, these data are generally obtained from stationary sensors (e.g. loop detectors), which are expensive to install and maintain and cover a small number of locations in the transport network. On the other hand, Floating Car Data (FCD) are readily available at the network level, usually from a sample of vehicles. We present an indirect traffic flow estimation approach using transfer learning to address the traffic flow data scarcity and model generalization across cities. Using two cities (Paris and Madrid) as study areas, we demonstrate the indirect estimation using only exogenous features for flow prediction, mirroring limited predictive features without past link flows. Subsequently, we use the model pre-trained on data from Paris city and test on data from Madrid city, and investigate the scenarios for successful transfer learning. Overall, the training set must adequately capture the flow-speed relationship for successful indirect flow estimation. Transfer learning is beneficial when the data for the target task is minimal, in which case transferred models outperform newly trained models from scratch. Using real-world and publicly available data, our approach and models can help scale a smaller traffic flow dataset to a larger sample across cities.

## 1 | INTRODUCTION

Traffic forecasting is a prevalent task in traffic research with applications in traffic management. Traffic state is characterized by the three main variables, that is, flow (or volume), speed, and density. Researchers have developed a wide range of models for traffic forecasting ranging from so-called white-box models (statistical models such as simple moving average or autoregressive regression) to black-box ones (deep learning models such as feedforward or recurrent graph neural networks) [1]. The current state of the art shows that deep learning models have been successful in traffic prediction tasks [2]. These advanced models can use spatial, temporal, contextual, and network/topological information for forecasting. Developing algorithms, model testing, and evaluation has been one of the core challenges in traffic forecasting [1].

Data is the second pillar supporting traffic forecasting research, apart from models. Traffic data come in many forms, such as point data, point-point data, or area-wide data [3]. The

form depends on the method of data collection and its source. Fixed ground-based sensors, such as induction loop detectors or street-side cameras, detect vehicles on or across a specific location. They can only provide localized observations such as flow, density, or spot speed. Traffic data from onboard devices, such as smartphones or navigation systems, primarily use Global Navigation Satellite System (GNSS) receivers to record the location of vehicles during their trip and, thus, provide mobility metrics (location, speed, travel time) for trip legs. This data, when gathered from a larger fleet of vehicles, is also known as Automatic Vehicle Location (AVL) data or Floating Car Data (FCD), or Probe Vehicle Data (PVD). New drone-based data collection methods can provide observations over an area or part of the network since they have a wide field of view [4]. However, these methods are relatively new and not yet adopted on a large scale for long-term data collection.

One of the main features of traffic flow forecasting methods is that time-lagged flow or speed data are used as an input in autoregressive formulations [2, 5]. For instance,

This is an open access article under the terms of the [Creative Commons Attribution-NonCommercial License](https://creativecommons.org/licenses/by-nc/4.0/), which permits use, distribution and reproduction in any medium, provided the original work is properly cited and is not used for commercial purposes.

© 2022 The Authors. *IET Intelligent Transport Systems* published by John Wiley & Sons Ltd on behalf of The Institution of Engineering and Technology.

past/time-lagged values of a signal (flow or speed) or variable are used to predict a future variable (speed or flow, respectively). However, in some cases, such flow data is wholly or partially unavailable or contains lag—thus, posing challenges for deploying traffic prediction models and their applications in natural settings. Moreover, using benchmark datasets such as the California Department of Transportation (2020) [6] does not portray the data availability challenges, varying from region to region. Hence, it could inhibit the practical deployment of traffic forecasting models.

Another issue worth highlighting is the ease of collection and data availability. Speed and flow (volume) data tell us different aspects of the traffic state. Flow tells us about the load on the link or the number of vehicles passing through a specific road and has applications in highway and pavement design, highway-side advertising, and commercial or real-estate investments. In this work, we use the term **link** to imply road segments. Speed depicts the link's congestion, travel time, or time delay. Speed does not directly represent the number of vehicles on the road. Instead, speed data is used to provide travel time estimates or the level of service on the road. Traffic flow data are more challenging to collect than link speed data since the latter can be approximated from a sample of vehicles [7, 8]. Speed data are derived by aggregating the traces from a fleet of cars—also known as PVD or FCD—or mobile phones via the GNSS receivers. This is one of the reasons network-wide traffic speed data are more prevalent for more cities than traffic flow data. Many companies [9, 10] collect data from vehicle fleets or smartphones, and some make it publicly available to a certain extent [11]. For instance, Uber, a global ride-hailing company, shared such data during 2016–2020 for many cities worldwide under non-commercial use licenses [9]. The data is available for download in bulk. In addition, navigation companies such as TomTom provide link speed data with limited free API calls followed by paid usage.

On the other hand, collecting traffic flow or count data requires dedicated hardware or collection methods which incur time and cost burdens. Specifically, traffic flow data is primarily collected via magnetic loop detectors installed on the streets. Such data collection infrastructure comes with high installation costs, is not scalable, and is common in many cities. As a result, traffic flow data are not available for many cities worldwide and are scarce even for cities in developed countries.

Due to the collection and coverage asymmetry between traffic flow data and traffic speed data, there arises an opportunity to use the speed data to infer the traffic flows. However, the possible solutions exclude using autoregressive models since time-lagged flow is assumed to be unknown and needs to be indirectly estimated. Therefore, this is a problem of indirect flow estimation. Here, we see the potential to use publicly available data to derive flow data from samples to larger parts of networks. Our first research question is motivated to address the data scarcity within a study area or city: “how accurately can we indirectly estimate the dynamic link flows at the network level given the link's speed data from open data sources?” The second research question is aimed to address the data

scarcity across cities: “What are the conditions when the pre-trained model can successfully be applied to a new city?” We use real-world study areas where this could contribute to tackling data scarcity. For instance, traffic counting stations only cover a limited part of the network and are expensive to install. Thus, the proposed model could help scale the flow of information from a few links to the whole network, provided the characteristics of the test data (link characteristics, spatial-temporal conditions) have a similar distribution as the training data.

Another downstream application of such an exogenous flow prediction model will tackle the underdetermination encountered in the calibration of large-scale traffic simulation models. Specifically, link counts are common data used to calibrate traffic simulation models, but these data are often only available for a few links. Therefore, an increased number of flow observations can help improve the dynamic origin-destination demand estimation solution. The proposed model can help overcome the challenges researchers and practitioners face in traffic management and transportation modeling due to data scarcity.

In this pursuit, we first identify prominent data that capture the relevant features and are determinants of the traffic flow. We download and curate traffic data for Paris and Madrid from heterogeneous and publicly available sources. We fuse the collected data for training models to predict traffic flows. Secondly, we use machine learning (including deep learning) and explore the effect of link types on prediction error, feature combinations, the temporal distribution of errors, and models' uncertainties. The use of open data in our work makes it easier for researchers and practitioners to replicate and benchmark our models. Our research using open data demonstrates exogenous flow forecasting using deep learning. The results show that, when combined with relevant geometric, temporal, and contextual features, prevalent speed data from floating cars can help reasonably forecast flows. In case of data insufficiency, transfer learning helps obtain accurate flow predictions.

The rest of the paper is structured as follows: the next section reviews the literature on traffic flow forecasting, estimation, and transfer learning, the following section describes the methodology of the study describing the data processing and prediction methodology, and the following section presents the data collection, followed by data analysis, the next section presents the results of the study, followed by discussion and conclusion.

## 2 | LITERATURE REVIEW

This section is organized into two parts. First, we provide a state-of-the-art overview of traffic forecasting research. For a more thorough discussion of traffic forecasting, we refer the reader to review by [1, 12]. Secondly, we discuss traffic state estimation research. This structure is followed because indirect/exogenous flow prediction is at the junction of traffic forecasting and traffic state estimation. Lastly, we discuss transfer learning.

## 2.1 | Traffic forecasting

Time series forecasting is the task of predicting the values of a target given input covariates. Forecasting models, where the input and target variable is the same yet time-lagged, are called autoregressive models since the future values of the variables are predicted using their past values. Time series forecasting can be formulated or extended as non-linear time series forecasting, multi-time step forecasting (predicting a sequence of future values instead of a single value), and multivariate forecasting (input features consist of multiple time-series or scalar variables). A generalized formulation for autoregressive multivariate and multi-step forecasting is given below:

$$\begin{aligned} [W, (X^{t-kf}, \dots, X^{t-f}, X^t), (Y^{t-kf}, \dots, Y^{t-f}, Y^t)] \rightarrow \\ [Y^{t+f}, Y^{t+2f}, \dots, Y^{t+(p-1)f}, Y^{t+pf}], \end{aligned} \quad (1)$$

where  $W$  and  $X$  are fixed and dynamic features, respectively,  $X$  refers to time-varying exogenous features.  $Y$  is the target variable.  $f$  is the fixed time interval or the granularity of the data;  $k$  is the lookback length, that is, how much past data the model has access to make a prediction;  $p$  is the prediction or future horizon length;  $t$  denotes the current time instant.

In traffic forecasting, common features are traffic variables such as traffic flow, link speed, trip travel-time, traffic density, occupancy, and congestion [12]. A dataset collected by the California Transportation Agencies (CalTrans) Performance Measurement System (PeMS), known as CalTrans PeMS is one of the most popular datasets for traffic prediction. This data is point-type data (from inductive loops, side-fire radar, and magnetometers [6]) containing traffic volume, occupancy, and speed. Other popular datasets are the Beijing point and trajectory datasets [13]. Commonly, studies on short-term traffic forecasting deal with forecasting horizons in the range of a few minutes to a few hours [1].

Regarding targets, the main task in traffic prediction or forecasting is to predict traffic flow or speed. For our work, we use traffic “flow” for the volume or number of vehicles passing a road section over time, whereas “speed” is the link speed which could be space-mean speed or time-mean speed. While researchers have used different predictors or features in their models, one common feature is using a time-lagged target variable as one predictor since most time-series forecasting models use an autoregressive model formulation. Other features are representations/metrics derived from trajectory data, covariates (weather or time of the day), Spatio-temporal maps, or videos [14].

Machine learning models have been widely used for traffic prediction and forecasting. Specifically, deep learning has recently outpaced the traditional time-series forecasting models such as Autoregressive Integrated Moving Average (ARIMA), as evidenced by recent studies [2, 5]. This development has resulted in many innovative traffic forecasting model architectures using cross-domain concepts such as convolutional neural networks and recurrent networks from computer vision and Natural Language Processing (NLP). Recurrent Neural

Networks (RNN) are special neural networks with a chain-like structure capable of learning time dependencies. Long-Short Term Memory (LSTM) [15] and Gated Recurrent Units (GRU) are specialized to learn long-term dependencies using a similar chain-like structure with modified units. LSTMs and GRUs have been successfully applied in various tasks such as language translation and image captioning. In traffic forecasting, too, LSTMs have been used for extreme event forecasting [16] or network-wide traffic speed prediction [17]. Lara-Benítez et al. (2021) [18] conducted an experimental review of multiple deep learning models for time-series forecasting, including traffic datasets. They found that LSTM and CNN are the best models, the LSTM models obtaining the most accurate results. It is relevant to point out that their analysis did not cover relatively new models such as Graph Neural Networks (GNN), transformers, or models with attention mechanisms. RNN-based architectures struggle to learn long-term dependence, and thus, attention mechanisms were applied by Wu et al. (2018) [19] to address this shortcoming. The attention mechanisms can identify and select information in the input relevant to a specific task, even if it is a long-term dependency. The attention layer assigns weights to specific input sequence regions relevant to the prediction task.

GNNs have recently gained popularity due to their ability to handle non-euclidean data. GNN models can also handle topological correlations between entities in the data using node and edge features in graphs. Further, GNN has been applied on Spatio-temporal datasets, for example, by stacking time-dependent graph snapshots leading to architectures such as Graph RNN (GRNN) [20], diffusion convolutional RNN (DC-RNN) [21], Temporal Graph Convolutional Networks (T-GCN) [22], consisting of Graph Convolutional Networks (GCN) to handle spatial features and either of the RNN, GRU or LSTM units to handle the temporal features. Buroni et al. (2021) [23] applied a multi-task learning strategy with GCNs to predict flow and speed and tested their method on different types of roads. Despite the benefits of GNNs, Zhao et al. (2020) [22] found that graph networks struggle to predict peaks because of averaging effects.

## 2.2 | Traffic state estimation

The process of inferring traffic state variables using partially-observed information is known as traffic state estimation [24]. In the comprehensive review by Seo et al. (2017) [24], authors classified traffic state estimation methods into three categories: model-driven, data-driven, and streaming-data-driven, based on preliminary information and input data. When the target or the predicted variable is traffic flow, it is called traffic flow estimation. Since practitioners and researchers use different traffic data for flow estimation, it can also be classified into direct and indirect methods based on the applied data collection methodology. Direct methods refer to counting vehicles on the road using manual or automatic techniques (magnetic loop detectors, gantry cameras, or drone videography). Direct methods use physical, visual (street cameras, drone videography, or satellite imagery), acoustic [25], or other signals (Bluetooth or cellular) to

detect the presence of a vehicle. Indirect methods try to estimate the flow using exogenously-correlated data. As shown by Aslam et al. (2012) [8], indirect methods use analytical and data-driven models for mapping predictor variables to the flow variables. A generalized multivariate and multi-step formulation for indirect state estimation is given in equation 2. A clear distinction from equation 1 is that the time-lagged target variable is not available as a predictor in the indirect traffic state estimation. Thus, for a given domain and data, the indirect state estimation is more challenging than traffic forecasting due to lesser information in its predictors.

$$\begin{aligned} [W, (X^{t-kf}, \dots, X^{t-f}, X^t)] \rightarrow \\ [Y^{t+f}, Y^{t+2f}, \dots, Y^{t+(p-1)f}, Y^{t+pf}]. \end{aligned} \quad (2)$$

In this paragraph, we review some representative studies on dynamic and indirect flow estimation. The traffic fundamental diagram is one of the most popular and well-established models relating traffic state variables (flow, speed, and density or occupancy). Different fundamental diagrams, such as Greenshield's fundamental diagram [26], correlate traffic flow with speed [27]. Therefore, readily available speed data (together with other covariates) can be used to predict traffic flows if the fundamental relationship is manifested. One drawback of the fundamental diagram is that it lacks a time-dependent representation of traffic state variables and, thus, requires suitable model extensions for a dynamic representation. To handle this, Neumann et al. (2013) [7] used Bayesian networks to model the time dependencies and predict traffic flows (from six hundred detectors) from speed data for the city of Berlin. Kumarage et al. (2018) [28] used K-nearest neighbor regression with Spatio-temporal attributes to predict flow at fewer locations. Pun et al. (2019) [29] used topological and geometric features for traffic flow estimation. Gkoutouna et al. (2020) [30] used data from thirty-six sensor locations for developing bi-level flow estimation models. The novelty in their method was the use of principal component analysis and clustering to identify road segment archetypes in the first level. This information is used in the second-level regression model. Rinaldi and Viti (2020) [31] used a Kalman filtering framework for flow estimation. Zhang et al. (2020) [32] used a geometric matrix completion model for network-wide traffic flow estimation, using real-world (twenty-four road segments) and synthetic datasets. Recently, Abdelraouf et al. (2022) [33] used speed and volume features from PVD as an input to recurrent GCN to predict the traffic state parameters. In contrast to other indirect estimation works, using the flow from PVD as an additional feature provides direct information for accurate prediction but also makes the model dependent on such data. Among the above, only few studies [7, 33] consider data from more than a hundred detectors, whereas the other studies use relatively minor datasets. Further, Neumann et al. (2013) [7] note that their model performance was not accurate enough for freeways or roads with higher speed limits. Traffic flow estimation is complicated by static or dynamic changes in link characteristics such as speed limit, the number of lanes, data collection, data quality (presence of noise and

anomalies), and data processing (smoothing, aggregation). Further, the methodological steps play a pivotal role when processing raw FCD to obtain link speed data [34]. These factors can distort and induce scatter in the fundamental diagram, thus rendering indirect traffic flow estimation quite challenging.

### 2.3 | Transfer learning

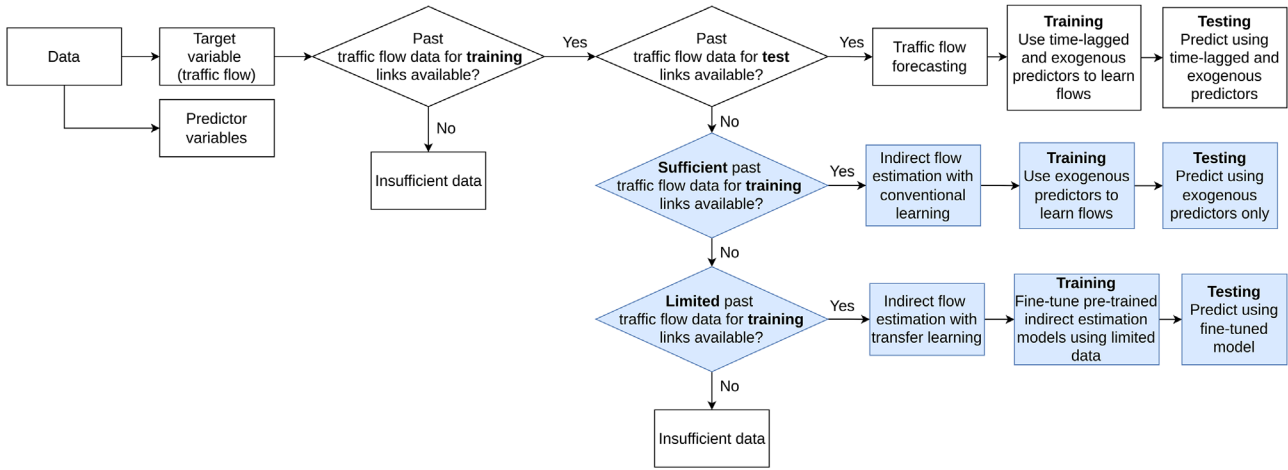
Machine learning models are developed under the assumption that the training data and test data “must be in the same feature space and have the same distribution” [35]. However, this does not generally hold true when applying models to the new study area. Even if the same set of features is developed for the data from two locations, the data distributions from two study areas can differ. This will have an impact on the model performance. This challenge is tackled using Transfer learning, which is improving the learning of a new task (target task) using the knowledge from an already learned related task (source task) [36].

Transfer learning is defined formally in terms of domain and task in the survey paper by Pan and Yang (2010) [35]. A domain  $\mathcal{D}$  consists of a feature space  $\mathcal{X}$  ( $X = \{x_1, \dots, x_n\} \in \mathcal{X}$ ) and a marginal probability distribution  $P(X)$  [35]. A task consists of a label space  $\mathcal{Y}$  and an objective predictive function  $f(\cdot)$  (denoted by  $\mathcal{T} = \{y, f(\cdot)\}$ ) [35]. The objective predictive function is not observed but can be learned from the training data.  $f(x)$  can be written as  $P(y | x)$ . After the model is learned on the source domain ( $\mathcal{D}_S$ ) for a source task ( $\mathcal{T}_S$ ), the aim is to transfer the learned knowledge for learning the target predictive function  $f_T(\cdot)$  for solving the target task ( $\mathcal{T}_T$ ) in target domain ( $\mathcal{D}_T$ ) [35], where  $\mathcal{D}_S \neq \mathcal{D}_T$ , or  $\mathcal{T}_S \neq \mathcal{T}_T$ . For instance, two cities, even with the same road links, could display different levels and patterns of traffic flow, depending on local traffic conditions.

Pan and Yang (2010) [35] noted four transfer learning algorithms based on how the knowledge is transferred from the source task to the target task. These four types are instance-based, feature-based, model-based, and relation-based algorithms. In deep learning, using pre-trained models for secondary tasks is essentially the same as model-based transfer learning [35]. In model-based transfer learning, it is assumed that the model's parameters learned from the source domain will be helpful for the target task in the target domain. These parameters and hyperparameters are fine-tuned using the limited training data from the target domain.

For short-term travel-time prediction, Luan et al. (2018) [37] showed that link-to-link transfer of their model is possible but emphasized further research into factors that affect transferability. Li et al., (2021) [38] and Mallick et al. (2021) [39] used transfer learning techniques in short-term traffic prediction using source and target links consisting of links from different locations. Li et al. (2021) [38] found that transfer learning can provide more accurate predictions when the source and target links have consistent data patterns. When sufficient labeled data is available, it makes sense to train the model using that without any pre-trained model. Mallick et al. (2021) [39] proposed the Transfer Learning-DCRNN model





**FIGURE 1** Flow chart showing scenarios for traffic flow forecasting, indirect flow estimation and transfer learning, depending on data availability

using a graph-partitioning-based transfer learning approach for short-term traffic forecasting, which outperformed other models. Using source knowledge through transfer learning can help to reduce dependence on large datasets and improve the existing models. We are also interested in investigating when or at what levels of data transfer learning outperforms the new models trained from scratch. Transfer learning for indirect traffic flow estimation is still not explored.

## 2.4 | Summary

With the help of the flow chart (Figure 1), we want to position our study when there is no historical/ real-time flow data for the links we want to make predictions. We divide the links into two sets: training links and test links. If traffic flow data is available for both sets, then traffic forecasting can be used, where time-lagged traffic flow is used as one of the predictors. If past data for test links are unavailable, the indirect traffic flow estimation approach using conventional learning is applicable, where only exogenous predictors are used to learn the traffic flow mapping so that during the model testing, we do not need time-lagged flow values as predictors. In the previous case, if sufficient flow data for training links are unavailable, flow estimation is done using transfer learning. Here, the pre-trained model is fine-tuned using limited data. Finally, if no flow data for training and test links are applicable, which means we only have predictors but no targets, then it is a case of insufficient data.

From the above review, some research gaps are evident. Firstly, deep learning models are not prevalent in dynamic and indirect traffic state estimation, unlike in traffic forecasting. This is true even though both approaches have inherent similarities among input and target features. Apart from prediction, quantifying the uncertainty in flow estimation is also vital. A model which outputs a range of predictions can help us to judge its preciseness. Lastly, the transferability of the indirect state estimation models using real data is still unexplored.

Because of the above and the public availability of large longitudinal traffic data, we use LSTM, a sequential deep learning

model for dynamic indirect flow estimation with uncertainty. Here, we also quantify the accuracy differences between different sets of features that correspond to traffic forecasting or indirect state estimation problem formulations. We conduct a systematic sensitivity analysis for different lengths of input feature sequences and output flow sequences to identify the best input-output length configuration. Finally, we transfer the model to an entirely new location and identify the best transfer learning configuration. We also show at what proportions of the data transfer learning outperform training a machine learning model from scratch.

## 3 | METHODOLOGY

The overall methodology of the study is shown in Figure 2. As mentioned above, We rely on the publicly available data in this study. Our research needs traffic flow counts and speed data to train and validate our models for the exact links. We identify prospective study areas based on availability and retrieve the data to achieve this. Since it is expected that the traffic count and speed data are in different formats, we fuse these datasets. Samples of links' speed and traffic counts are matched according to the link/ location in data fusion.

This paper aims to develop a model which predicts traffic flow in transport networks exclusively from given link speeds and other relevant time and contextual covariates. Thus, we adopt a formulation for indirect traffic estimation and assume the unavailability of time-lagged flow data as a predictor. Inspired by [39], we borrow and adapt their problem formulation for our case. Given, a set  $\mathcal{S}$  of  $d$  links, the traffic flow at time step  $t$  for a  $d^{\text{th}}$  link is  $Y_t^d \in \mathbb{R}^1$ . For all the links in  $\mathcal{S}$  such as the  $d^{\text{th}}$  link, given static predictors  $W^d \in \mathbb{R}^E$ , where  $E$  is the number of such predictors (length, number of lanes, road width, type, speed limit) and  $H$  historical observations of dynamic exogenous predictors (speed, time, speed deviation)  $X^d = (X_{t_1}^d, X_{t_2}^d, \dots, X_{t_H}^d) \in \mathbb{R}^{H \times E}$ ,  $H$  historical observations of the traffic flow  $Y^d = (Y_{t_1}^d, Y_{t_2}^d, \dots, Y_{t_H}^d) \in \mathbb{R}^H$ ,  $P$  current observations of the same predictors  $X^d = (X_{t_1}^d, X_{t_2}^d, \dots, X_{t_P}^d) \in$

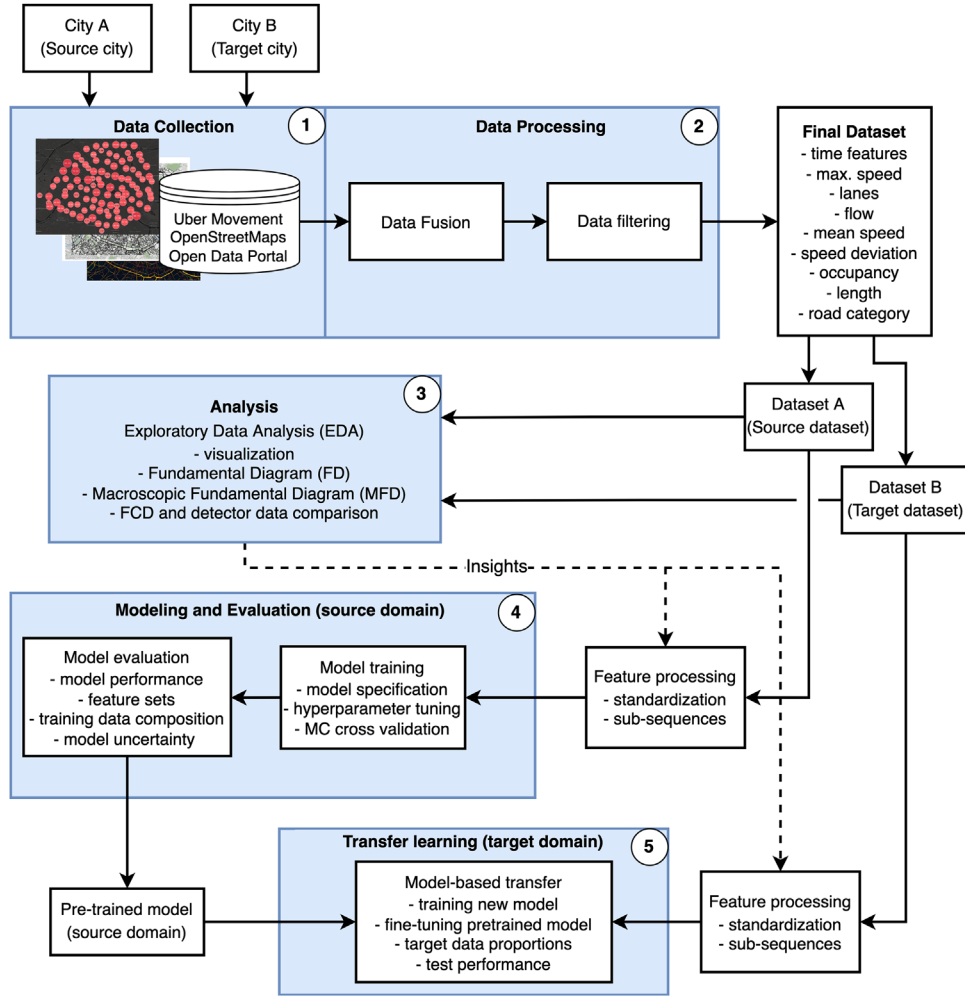


FIGURE 2 Methodological flow showing data collection, analysis and modeling

$\mathbb{R}^{P \times F}$ , where  $F$  is the number of such predictors and  $H \gg P$ , we want to forecast the traffic flow of all the links in  $S$  for the next  $Q$  time steps,  $\hat{Y}^d = (\hat{Y}^d_{t_{p+1}}, \hat{Y}^d_{t_{p+2}}, \dots, \hat{Y}^d_{t_{p+Q}}) \in \mathbb{R}^Q$ . Thereafter, Let  $S'$  be the set of  $j$  links for which we do not have the historical time series data. Given static predictors  $W^{j'} \in \mathbb{R}^E$ , and  $P$  observations of the current exogenous predictors for  $j^{th}$  link  $X^{j'} = (X^{j'}_{t_1}, X^{j'}_{t_2}, \dots, X^{j'}_{t_p}) \in \mathbb{R}^{P \times F}$ , the goal is to develop a model that can forecast the traffic flow of the next  $Q$  time steps for all the links in  $S'$ ,  $\hat{Y}^{j'} = (\hat{Y}^{j'}_{t_{p+1}}, \hat{Y}^{j'}_{t_{p+2}}, \dots, \hat{Y}^{j'}_{t_{p+Q}}) \in \mathbb{R}^Q$ .

### 3.1 | LSTM model

Our task can be formulated as supervised machine learning because the model learns the mapping from features to the given targets. We select LSTM networks as our primary model. LSTM is appropriate for modeling time-series data such as traffic flow or speed, where correlations between time intervals have a long lag. Recent studies have also used LSTMs [38, 39] for comparing model performance. In studies [33, 39]

using GNN for short-term traffic forecasting, we observed that even though graph-based models (DCRNN) give the best performance, LSTM's performance is still competitive.

LSTM is considered a more advanced version of the standard vanilla RNN. Traditional RNNs model a sequence of events by using the output from the previous time interval as an input to predict the system's state ( $b_t$ ) at the current time interval. The key aspect of an RNN is, therefore, that the hidden state works similarly to the lagged variable in an autoregressive model, meaning that the predictions at the current time interval  $b_t$  depend on the hidden state at the previous time interval  $b_{t-1}$ .

Although a successful architecture, RNNs suffer from vanishing/ exploding gradients and short-term memory problems. To alleviate the issues above, LSTM introduces the cell state  $c_t$  in addition to the existing hidden state of RNNs. LSTM consists of a memory cell that controls the flow of information by using input, forget, and output gate layers that discard non-essential information and memorize only essential information. This architecture is used to update the cell state  $c_t$  and select which information should be preserved and which one should be lost. The operations in the LSTM model [15, 40] can be

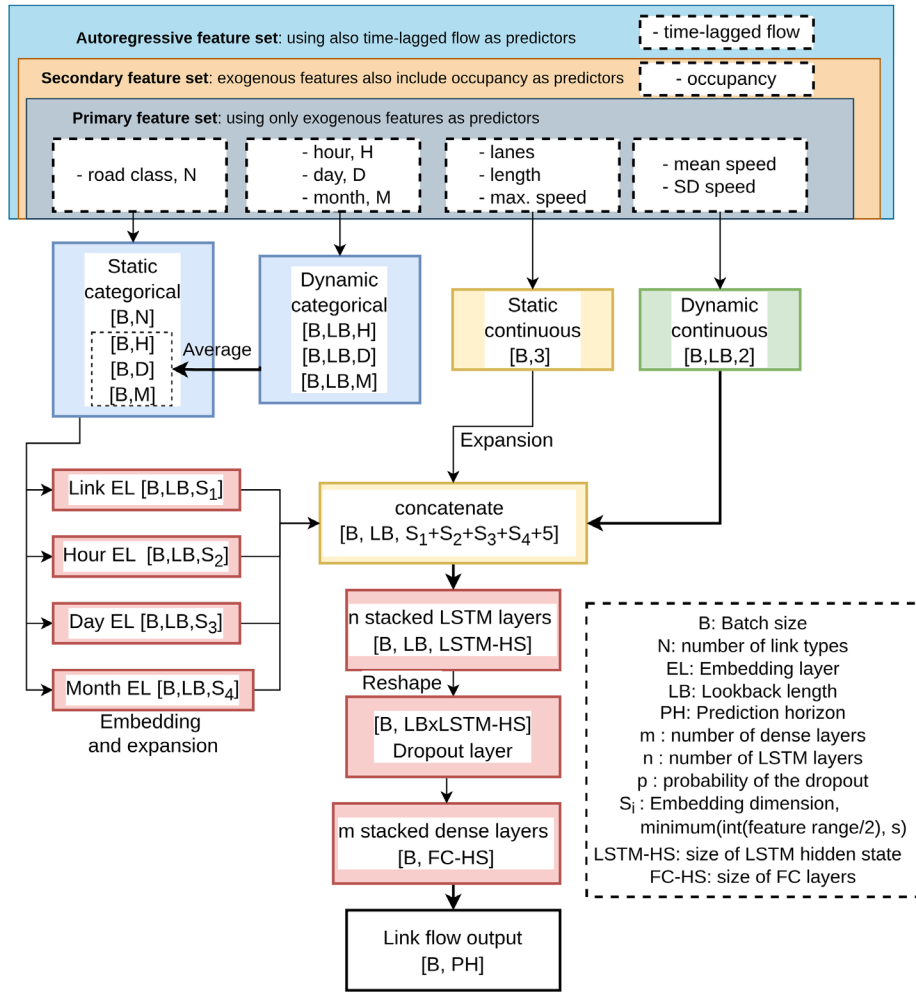


FIGURE 3 Architecture of the deep learning model using embedding and LSTM layers

represented by the following set of equations:

$$\begin{aligned}
 i_t &= \sigma(W_{ij}x_t + b_{ij} + W_{hi}b_{t-1} + b_{hi}) \\
 f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}b_{t-1} + b_{hf}) \\
 g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}b_{t-1} + b_{hg}) \\
 o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}b_{t-1} + b_{ho}) \\
 c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
 h_t &= o_t \odot \tanh(c_t)
 \end{aligned} \quad (3)$$

where  $b_t$ : hidden state of layer,  $c_t$ : cell state,  $x_t$ : input, all at time  $t$ ;  $b_{t-1}$ : hidden state of layer at time  $t-1$ ; and  $i_t, f_t, g_t, o_t$  are the input, forget, cell, and output gates, respectively.  $\sigma$  is the sigmoid function, and  $\odot$  is the Hadamard product. Equations are sourced from [41].

### 3.2 | Model architecture

The input data for the LSTM model can be static or dynamic and continuous or categorical, leading to four different input types (Figure 3): static categorical inputs (links type with  $N$

classes), dynamic categorical inputs (hour with  $H$  classes, day with  $D$  classes, month with  $M$  classes), continuous static inputs (link length, number of lanes, maximum speed), continuous dynamic inputs (hourly speed). In our model (Figure 3), we use *embedding* layers to process the categorical features. Embedding layers use a fixed-length continuous vector to represent a categorical feature. The embeddings are learned during model training, and similar feature categories will have closer representations in the embedding space. To reduce the need to learn the multiple embeddings for dynamic categorical features such as *hour*, *weekday*, and *month*, we convert the dynamic categorical features to static features by taking their average value over the lookback length. This gives us single values of these features for each training sample, irrespective of the lookback length. The single value provides the time context for the day, weekday, and month. The size of the feature's embedding layer ( $S$ ) is a minimum of  $s$  and half the number of unique values of a feature, where  $s$  is a hyperparameter. This step reduces the number of feature embeddings to be learned and, thus, reduces the complexity of the model. The dynamic features are provided for a certain past/lookback length, so the static embeddings and features are expanded along the time dimension. The *concatenate* layer combines then continuous

and embedding outputs to form a single time-dependent input for the next LSTM layers. This input is passed through a sequence of stacked  $n$  LSTM layer(s). LSTM layer(s) compute the function in Equation 3 at each time step or hidden state to produce the output equal to the latent dimension of hidden state.

Dropout is applied to the output of the final LSTM layer to randomly switch-off neurons with a probability  $p$ . Dropout is a regularization technique to reduce overfitting [42] and helps to improve the model's robustness. Dropout can also be used to understand the model's uncertainty [43]. Later during the model testing, we used dropout to obtain the uncertainty estimates by running the model several times (e.g. ten times) and obtaining the confidence intervals for the model predictions. The output of the dropout layer is fed to the stacked  $m$  fully connected or dense layer(s) to give a single output or a sequential output (in the case of multi-step forecasting), as shown in Figure 3.

### 3.3 | Feature sets

We construct three sets of features to observe the model performance differences, as shown in Figure 3. The first set of features for the primary model is only the *exogenous* features, which can be either static or dynamic. This set of features is the main focus of this paper. These features exclude flow and occupancy data from the detectors and thus only use easily available data. In the second set, we add time-lagged occupancy ( $o$ ) data from detectors to the exogenous set (*exogenous covariates, o*). In the third set, we further add time-lagged flow ( $q$ ) to the second set resulting in (*exogenous covariates, o, q*). When using the third feature, our model follows autoregressive formulation. We hypothesize that model performance will improve with the inclusion of  $o$  and  $q$  into the feature set since the model has direct and endogenous information to predict the flow. However, this information cannot be used when the objective is to predict traffic volumes on links that are not equipped with sensors. Before training, features in source and target data are standardized (removing the mean and scaling to unit variance) independently.

### 3.4 | Model evaluation

We use the XGBoost regression model as the benchmark since it is not designed to handle sequential data. However, XGBoost has shown superior performance on tabular datasets in research and practice [44]. Therefore, we use XGBoost with non-sequential inputs. This means we only provide the input data for the current time step but not the past intervals while predicting one step. The XGBoost model [45] is based on the Gradient Boosting Machines (GBM) concept. In boosting, observations with high residuals generally receive ever-increasing influence with each iteration [46]. Boosting models are generally considered "off-the-shelf classifiers" [46] and need less feature preprocessing and parameter tuning than other deep learning models such as neural networks.

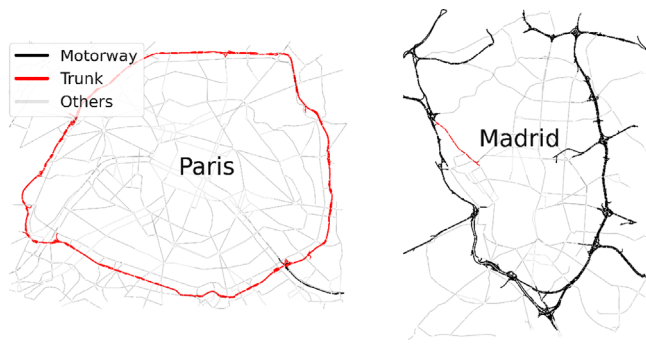
The dataset corresponding to all links is split into a train set (85%) and a test set (15%). We use group-based splitting (using detector ID) of the data, which means that data from each detector can be into either a train set or test set to avoid overestimating the model performance due to temporal correlations within the data from one detector. This also mirrors the scenario of data availability for partial links in the network. The hyperparameters of the LSTM model used in this paper are the size of the embedding layers, learning rate, batch size, maximum epochs, number of LSTM layers ( $n$ ), size of the LSTM hidden state, dropout rate ( $p$ ), and weight decay or L-2 type regularization of weights ( $w$ ), and number ( $m$ ) and size of the dense layers. The primary hyperparameters of the boosting model are the number of iterations and the size of each of the constituent trees (number of leaves in the tree) [46]. The model training is stopped when the validation error does not improve for twenty iterations. This is also known as early stopping. We use Bayesian optimization [47] to tune the hyperparameters of the XGBoost and LSTM models. During tuning, the Monte Carlo cross-validation (MCCV) error is used. In MCCV, the model is trained by randomly splitting the training data into training (85%) and validation data (15%) for ten runs. The average error on the validation data is used to select the best hyperparameters.

Choice of forecasting metric is crucial and varies from task to task. In our case, we have a time series corresponding to each detector. The target variable (flow) scale can vary between links belonging to different categories. Thus, we use percentage error metric [48]. Mean Absolute Percentage Error (MAPE) is one of the popular percentage error metrics. However, we do not use MAPE because it has no upper bound and can be problematic when the actual values are close to zero. Instead, we select Symmetric Mean Absolute Percentage Error (SMAPE), shown in Equation 4, as the model training and evaluation criterion due to the time-series nature of the input data.

$$SMAPE_d = \frac{1}{n} \sum_{i=1}^N \frac{|x_i - \hat{x}_i|}{(|x_i| + |\hat{x}_i|)/2}, \quad (4)$$

where  $SMAPE_d$  is the SMAPE for the  $d^{th}$  link,  $\hat{x}_i$  is the predicted value, and  $x_i$  is the observed value. In contrast to MAPE, SMAPE has both lower and upper bound. It is noted that the data within each detector is correlated. Thus we use mean error over detectors instead of the sample means to prevent the dominance of detectors with large samples. In other words, we first estimate the error for each detector and then estimate the mean error over all the detectors for reporting the model performance. One of the drawbacks of SMAPE is that it does not treat large positive and negative errors equally and thus is not "symmetric" as its name suggests [49]. No single metric is sufficient for forecasting, so we also use RMSE for model evaluation. However, RMSE is only used when the target scale is similar, so the evaluation is fair, for example, when the data for a single link type is used. We use the Python frameworks XGBoost [45] and Pytorch [40] for developing the XGBoost and the LSTM model, respectively.





**FIGURE 4** Full Road network within ring road of Paris (left), that is, source domain and Madrid (right), that is, target domain. Maps created using Python library OSMnx [51]

### 3.5 | Model transferability

We select the best-trained model on the source data and check its generalization ability if the model can be applied to study areas without or insufficient traffic flow data. This is done using transfer learning. We collect and prepare the source data ( $\mathcal{D}_S, \mathcal{T}_S$ ) and target data ( $\mathcal{D}_T, \mathcal{T}_T$ ) with same set of features ( $\mathcal{X}_S = \mathcal{X}_T$ ) and same type of labels ( $\mathcal{Y}_S = \mathcal{Y}_T$ ). Still, there is no guarantee that the feature's marginal distributions ( $P_S(X) \neq P_T(X)$ ) and the label conditional probability distributions ( $P(Y_S | X_S) \neq P(Y_T | X_T)$ ) are similar among the source data and target data. For instance, link attributes and traffic flow patterns can vary between locations. Thus, this is a case of transductive transfer learning [35], or domain adaptation [50]. We use model-based transfer where a pre-trained model is used. This means that the weights of parameters in the pre-trained model are used as priors or initial values for the target task. Further, the model can be used for the target task without any changes or further retraining or fine-tuning on the sample of target data. We devise a systematic method to transfer the trained model as listed below and check model performances to find the best transfer learning scenario.

1. Baseline model without transfer learning. Here, the model with the same architecture as the source task but randomly initialized parameters are used. This model is trained and tested on the target data only; thus, it has no knowledge transfer from the source task.
2. Transferring the model pre-trained for source task without retraining on the target data. Model architecture is also not changed.
3. Transferring the model trained for the source task by fine-tuning one or more of the fully connected (FC) layers, LSTM layers, and embedding layers, but the rest of the model is frozen. For instance, LSTM layers with parameters from the source task are fine-tuned on target data. In contrast, the parameters of the rest of the layers remain fixed. Model architecture remains the same as in the source task.

For the target domain, we test our model under different proportions of training data, simulating the scenarios of limited

training data availability. For evaluating the model transferability, we use twenty runs of MCCV. First, target data is divided into training and test set. In each MCCV run, target training set detectors are further divided into the training and validation set according to the training proportion, for example, if the training proportion is 0.65, then 65% of detectors (excluding detectors corresponding to test links) are assigned for re-training the pre-trained model, and rest are used for the model validation (e.g. early stopping). After model training, a test set is used to evaluate all the models from all the runs. Thus, our approach can capture the sampling variability during model transfer.

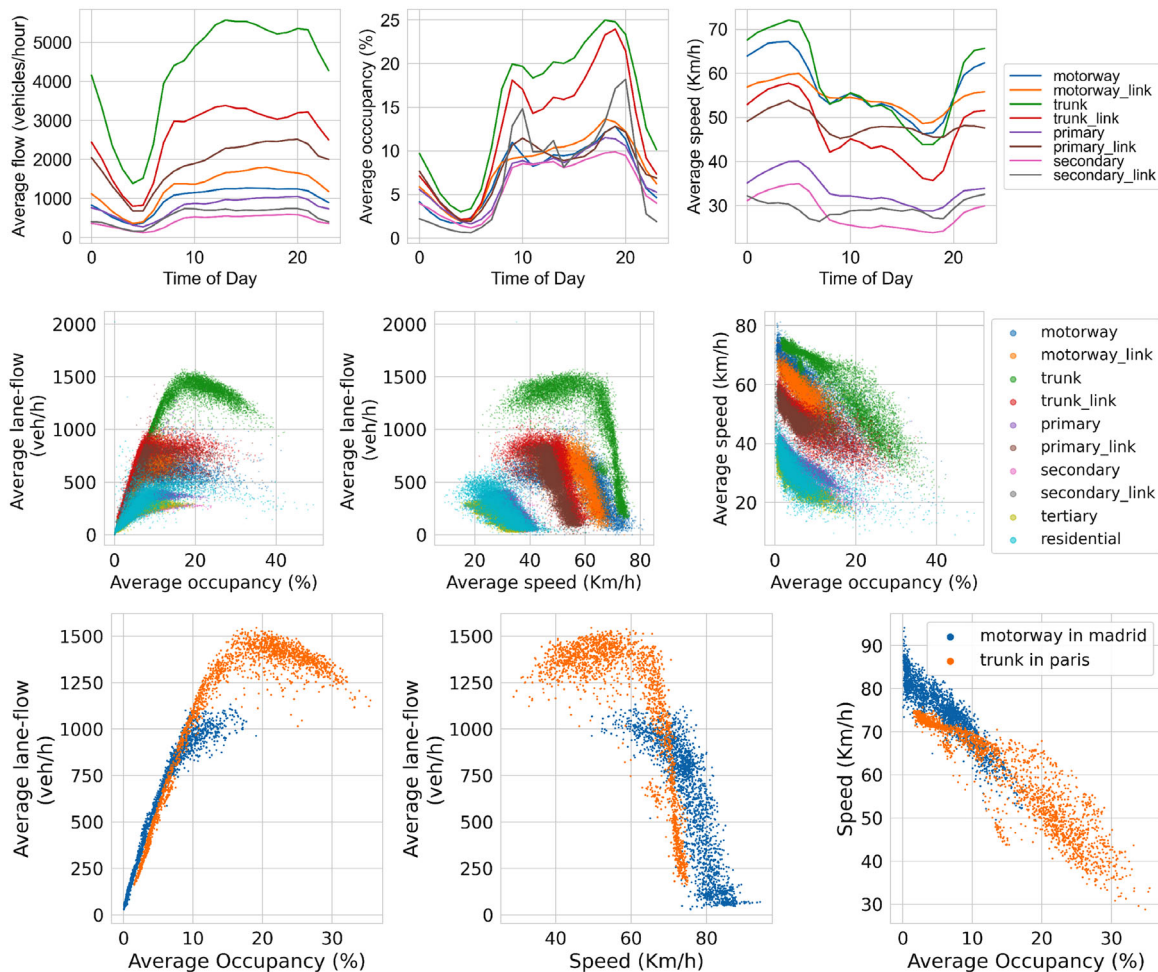
## 4 | DATA COLLECTION

We use Paris (region within Paris' ring road or Boulevard Périphérique) as our primary study area or source domain/ city (Figure 4), as the Paris open data portal [52] provides historical traffic flow/volume and occupancy data. For this study, we assume we have a sufficient source dataset for training and testing our model. We train our original model using these data.

Traffic flow data (dependent variable) are collected from the traffic sensors (loop detectors) installed on the road. The data for the full year 2019 was retrieved. We use these data for training our machine and deep learning models. The raw dataset from the portal is at the aggregation interval of one hour, and it defines the predictive resolution of our models. Our models cannot predict for a horizon of less than one hour.

We use link speed data from the Uber Movement portal for the same study area and period. Uber, a Transportation Network Company (TNC), provides aggregated speeds by road segments at hourly granularity [53]. The speed values are derived from average speed readings from on-trip ride-hailing vehicles associated with the Uber [53]. The raw data in GPS (Global Positioning System) pings are ingested in real-time every four seconds. Uber performs map matching based on a Hidden Markov chain Model (HMM) to assign the GPS pings to a road segment. This map-matched data are used to calculate traversal speed per segment [53]. Speed is given by dividing the length of the road by the time a vehicle takes to traverse it. Uber does not publish speed if the number of traversals is below a minimum threshold to safeguard privacy. Finally, the speed traversals are aggregated into time windows during a time interval. We retrieved the data from [9]. In the retrieved data, each road segment has a mean speed and a speed deviation at hourly granularity in 2019.

We use Madrid as the secondary study area or target domain/ city for investigating transfer learning performance. Open Data Madrid provides historical data from flow, occupancy, and speed (only for inter-city roads) at an aggregation interval of 15 min [54]. The data are aggregated at the interval of one hour so that the attributes are consistent with the model trained using Paris data. Link speeds for Madrid are also available from Uber Movement (one-hour interval). For Madrid, therefore, we have link speed data from two sources, and we can compare these two sources to check the plausibility of the FCD data (from Uber Movement).



**FIGURE 5** Average trends (top) of the speed, flow and occupancy for the links from source city, MFD shows average flow, occupancy, and speed for (middle) all links from source (Paris) city. MFD (bottom) for links from source and target (Madrid) cities for trunk and motorway links, respectively

To match the flow and speed data, we use Shared Streets, a standard for streets, that is, roads are assigned a unique identifier for referencing. Shared Streets [55] also provides a tool to match the geographic objects (in the form of points and edges) using a probabilistic HMM map matching [56]. Since the flow and speed datasets are geo-referenced, we utilize this tool to map traffic flow and speed data to a common Shared Streets standard, and then merge them. We also retrieve the road’s static features from the OpenStreetMap (OSM). These features include length, type, number of lanes, and speed limit. During data fusion, we dropped the roads/ links if static features (number of lanes or speed limit) were missing.

In the OSM data, highway segments are classified into *motorway*, *trunk*, *primary*, *secondary* and *tertiary*, *unclassified* and *residential*, according to their importance in the road network. Further, the segment or link types such as *motorway\_link*, *trunk\_link*, *primary\_link* and *secondary\_link* refer to the slip roads/ramps and physically separated at-grade turning lanes in the OSM data. For definitions of these links, we refer the reader to [57] and [58]. Finally, we do not consider the effects of dynamic traffic management on features (such as dynamic speed limits) derived from OSM data because such data are unavailable. The flow-

speed-OSM matched data consist of a time series for each of the road segments with static (geometric and contextual) and dynamic (speed and flow) features (Figure 2).

## 5 | DATA ANALYSIS

We show the trends of the mean flow, speed, and occupancy for different link types during the day in Figure 5 (top row). Trunk type links show the highest average flows. We also plot the Macroscopic Fundamental Diagram (MFD) [59] or Network Fundamental Diagram (NFD) [60] and traffic fundamental diagrams to understand the traffic state dynamics. We use the weighted average formulation by [61] to represent the MFD mentioned below:

$$q_i^w = \frac{\sum_i q_{it} l_i}{\sum_i l_i} \tag{5}$$

$$o_i^w = \frac{\sum_i o_{it} l_i}{\sum_i l_i} \tag{6}$$

$$s_i^w = \frac{\sum_i s_{it} l_i}{\sum_i l_i}, \tag{7}$$

where  $q_i^w$ ,  $d_i^w$ , and  $s_i^w$  are average lane-flow, occupancy and speed, respectively, at time  $t$ ,  $l_i$  is link length,  $q_{it}$ ,  $o_{it}$ , and  $s_{it}$  are flow, occupancy and speed for the  $i^{th}$  link at time  $t$ . Separate MFD is estimated for each link-type category. Factors such as spatial distribution of the congestion and location of detectors can affect the shape, scatter, and the existence of a well-defined MFD [62].

Figure 5 (middle and bottom row) also shows the average traffic states or MFD (flow, occupancy, speed) for all links within the source city. Each point in the plot corresponds to a time interval of one hour. In this figure, we have not adjusted or filtered the data to account for the homogeneity of the congestion since it is not the focus of this study. Still, the existence of MFD is evident, albeit some link types show more scatter than others; for instance, the trunk type links show well-defined MFD over a wide range of speed values and occupancy (Figure 5). This is a crucial element for our experiment, as we focus only on temporal aspects and do not explicitly consider network characteristics (for example, through a GNN). MFD for other link types (such as residential type links) shows more scatter and is mostly confined over a narrow range of speed values. We suppose this is typical for urban roads with speed limits on the lower side (50 km/h or 30 km/h). Further, the scatter is prominent in all link types during high average occupancy or low average speeds. This could be due to heterogeneity in the congestion patterns leading to the loss of well-defined MFD.

Data for time instances when any of the link's dynamic (flow or speed data) features were missing were also dropped. Low importance roads have more missing speed data values. The remaining Paris data has median completeness of 68%, 63%, and 96% over a full year (8670 hours) for primary, secondary, and trunk links (Table 1), respectively. For Madrid, the median data completeness is 81% (motorway-type links). Due to the generally high rate of data completeness for these link types, we have enough samples (Table 1) for training and testing our models. Otherwise, as a pre-processing step, data imputation can be needed if data completeness is low [63].

We provide the descriptive statistics for the selected links in Table 1. For the source city, flow data from primary, secondary, and trunk link types constitute 90% of the dataset, with about 8.2 million samples for 1290 unique links. Source data for Paris consist of data of 809, 363, and 122 links for primary, secondary, and trunk links, respectively.

Primary and secondary links show similar speed characteristics with a mean speed of around 27–30 Km/h, whereas trunk links have a mean speed of 58 Km/h. Trunk links belong to the Boulevard Périphérique, an uninterrupted road system, but primary and secondary links are interrupted by traffic signals. Both primary and secondary links are shorter than the trunk links and have, on average fewer lanes. The mean hourly flow on primary and secondary links is about 650 and 450 vehicles/h, respectively, whereas trunk links have a significantly higher mean flow of 4150 vehicles/hour. In Table 1, it can be seen that scale, as seen from the mean and standard deviation (SD), of the flow values, are different for the primary, secondary, and trunk link

types. Also, flow variance in primary (503 vehicles/h) and secondary (342 vehicles/h) links is lower than that of trunk links (1865 vehicles/h). This is why we do not report RMSE when all links are considered in the training data because RMSE is a scale-dependent metric.

In the traffic fundamental diagram in Figure 6, it is evident that primary and secondary type links have more scatter than others. For instance, the trunk links display the evolving relationship between the flow and speed over a wide range of values at different times.

From midnight to the early morning, traffic remains majorly in a free-flow regime. We see the typical fundamental diagram from the morning to the evening hours, wherein links are either in free-flow, transitions, or congestion regime. This finding is essential for exogenous flow modeling because speed is clearly correlated with the flow for the trunk type links in the dataset. On the other hand, the same is not valid for the primary and secondary type links as their speed-flow plot is only confined within a limited range. Possible explanations are the existence of speed limits on the lower side (50 km/h or 30 km/h) and traffic signals, which results in non-uniform traffic states and restricted range of traffic state variables. Some scattering is also because plots are not controlled for variables such as speed limit, and number of lanes.

For the target data, we have data of 129 motorway links belonging mostly to M30 orbital motorway (also an uninterrupted system). The fundamental diagram for these links is shown in Figure 7. The mean speed on these links is higher (75 Km/h), and the mean flow is lower (2300 vehicles/h) than those on the trunk links in source data. This shows that features in source and target data have different distributions. However, their standard deviation of flow (1561 vs. 1865 vehicles/h) and speed (15.5 vs. 16.8 km/h) are similar. Although trunk and motorway link features have different distributions, their ranges have significant overlap (Figure 8). This makes us confident that a model trained using source trunk links is a better candidate for transfer than using a model trained with all link types.

## 5.1 | Comparison of detector and FCD data

While comparing the speed data from detectors and FCD sources for Madrid, we find that the data are not uniformly consistent across the traffic states, as shown in Figure 9. The mean error is high in regions of low data density (low speeds, very low flows, and high occupancy). This shows that FCD data is not reliable in these ranges. For flow values greater than 400 vehicles per hour, the mean percentage error stays within –10% to 5%. Speed data from UBER movement is more trustworthy in the flow higher than 400 vehicles per hour. The high percentage error occurs for low values of the speed, that is, less than 50 kmph, and for higher values of link occupancy, that is, greater than 30%. We conclude that link speed data from the UBER movement dataset is consistent with the detector measured speeds for high flows and low occupancy or higher speeds.

**TABLE 1** Descriptive statistics of the features in the pre-filtered Paris and Madrid data

	Max. speed (km/h)	Length (m)	Lanes (no.)	Hourly speed (km/h)	Speed SD (km/h)	Hourly flow (vehicles/h)
<b>Pre-filtered Paris data</b>						
Link type: <b>Primary</b> ; detectors: 809; samples: 4.28x10 <sup>6</sup>						
Min	30.00	10.05	1.00	0.39	0.07	0.00
Mean	48.85	187.94	2.77	29.89	10.27	654.19
Median	50.00	133.97	3.00	29.62	9.81	540.00
Max	60.00	933.01	5.00	126.36	70.99	11152.00
SD	4.33	153.59	1.05	10.24	3.84	502.94
Link type: <b>Secondary</b> ; detectors: 363; samples: 1.88x10 <sup>6</sup>						
Min	30.00	13.37	1.00	0.47	0.06	0.00
Mean	46.42	143.42	2.50	27.57	9.71	450.58
Median	50.00	108.87	2.00	27.71	9.32	364.00
Max	50.00	607.46	5.00	107.36	63.44	6152.00
SD	7.28	108.49	0.89	8.25	3.64	342.72
Link type: <b>Trunk</b> ; detectors: 122; samples: 9.60x10 <sup>5</sup>						
Min	50.00	97.20	2.00	0.83	0.11	0.00
Mean	69.80	609.13	3.77	58.13	10.28	4149.65
Median	70.00	581.63	4.00	65.17	9.38	4378.00
Max	70.00	1362.56	5.00	92.38	62.97	9021.00
SD	1.90	243.62	0.58	16.86	3.56	1865.58
<b>Pre-filtered Madrid data</b>						
<b>Motorway</b> ; detectors: 129; samples: 7.27x10 <sup>5</sup>						
Min	50.00	27.06	3.00	3.06	0.27	0.00
Mean	81.84	843.82	3.41	75.02	11.04	2299.09
Median	90.00	558.58	3.00	78.89	9.67	2117.75
Max	100.00	4658.27	5.00	141.77	67.55	8632.00
SD	12.42	925.01	0.62	15.56	5.77	1561.87

## 5.2 | Summary

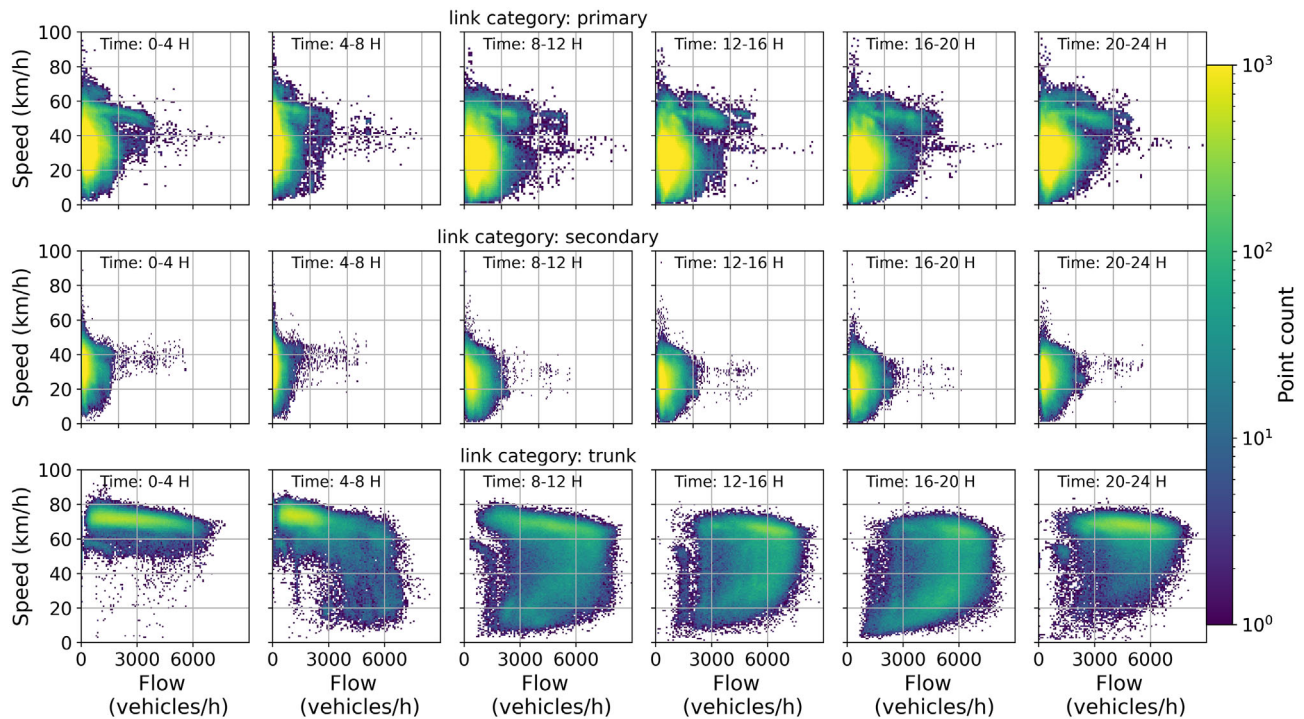
Based on the above analysis, we conclude that it makes sense to develop two types of models from source data based on the link types in the input data. The first type of model considers datasets from all three source link types for training. The second model only uses data from source trunk-type links. Contrasting between two models helps us confirm our belief regarding the adverse effects of the scattering in the fundamental diagram, distinct feature statistics, and speed data errors on the flow prediction model's performance.

## 6 | RESULTS

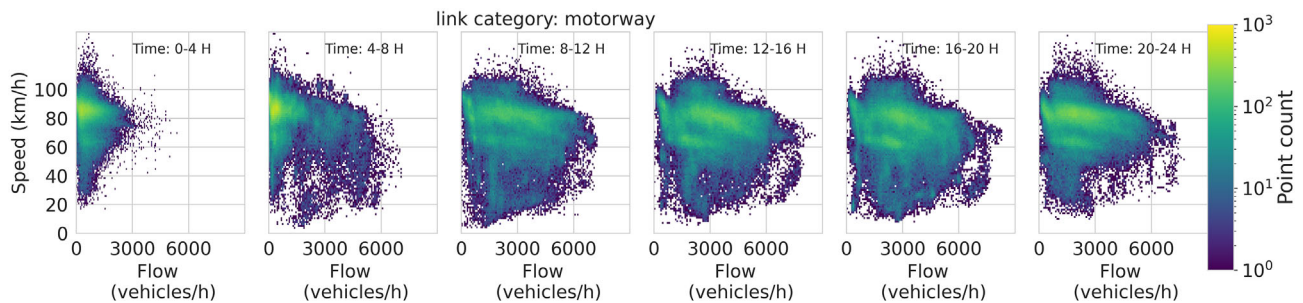
The list of hyperparameters and their range of possible values for search is shown in Table 2. The best parameters for the

XGboost and LSTM models are also shown in Table 2 and are based on the best SMAPE on the validation dataset. When the input data contains all links (primary, secondary, and trunk), both models fail to achieve good SMAPE on the test data and are under-fitting (Table 3). SMAPE of XGBoots and LSTM models are 51.76% and 40.17%, respectively. This finding was expected due to the lack of structure in the fundamental diagram for primary and secondary links and thus a weak correlation between speed and flow. Still, the LSTM model fits better than the XGBoost. For the input data with only trunk type links, the LSTM model again performs better than the XGBoost model in terms of both SMAPE and RMSE on test data (Table 3). LSTM model outperforms the XGBoost in test SMAPE and RMSE by approximately 21% and 13%, respectively. SMAPE and RMSE of the LSTM model on the test data are comparable to those on the training data, showing that the LSTM model can better generalize on the unseen

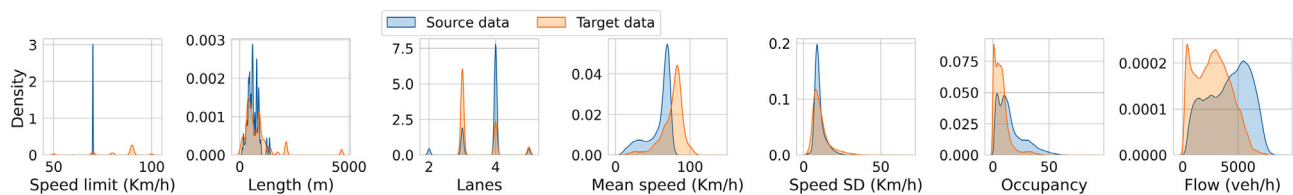




**FIGURE 6** Traffic fundamental diagram for primary (top), secondary (middle), and trunk (bottom) links from source city (Paris) during the year 2019. The fundamental diagram is more prominent for the trunk category links than the other links



**FIGURE 7** Traffic fundamental diagram for motorway links from target city (Madrid)



**FIGURE 8** Feature distributions for source and target data

data. Test SMAPE of the XGBoost model is much higher than on the training set, indicating an overfitting problem. Lastly, the confidence intervals of the LSTM model (Table 3) are narrower than the confidence intervals of the XGBoost model's error, indicating that the LSTM model has low variance.

We also show the effect of the lookback length and prediction horizon on the performance of the LSTM model in Figure 10. In Figure 10 and Table 4, model performance degrades with the increase in the prediction horizon. This degradation is expected since it becomes challenging to predict accurately with increasing prediction horizons. The model shows good performance

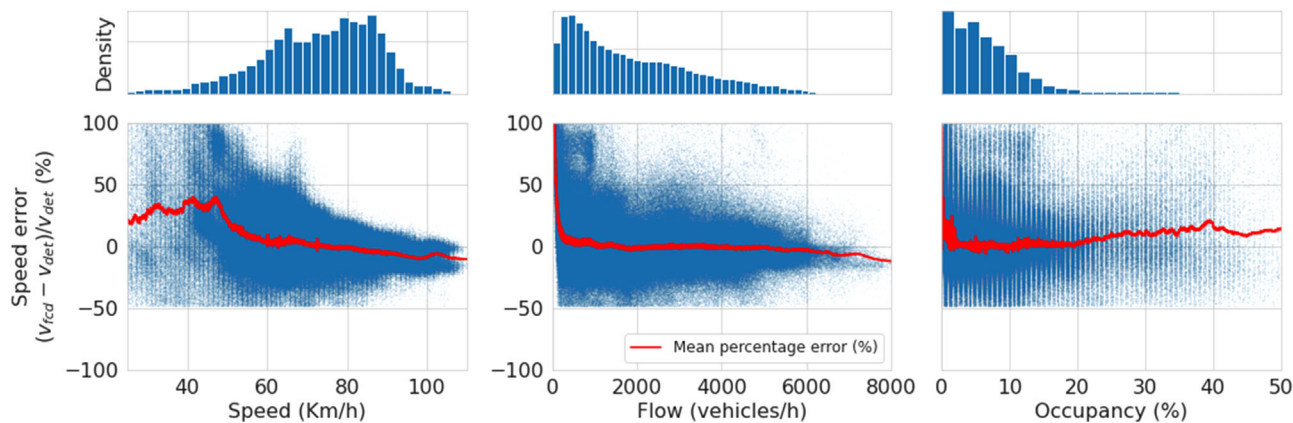


FIGURE 9 Speed error between stationary detector data and FCD data

TABLE 2 Parameter ranges for tuning hyperparameters of machine learning models using Monte Carlo cross-validation (MCCV)

Model	Parameter	Search range	Best value
XGBoost	Learning rate	(1e-4, 0.9)	0.4
	Maximum depth of tree	{2,3,4,5,6}	5
	Column sub-sampling	[0, 1]	1
	Sub-sampling	[0, 1]	0.6
	Iterations	up to 4000	Varies <sup>1</sup>
LSTM	Learning rate	(1e-6, 1e-1)	1.8e-4
	Batch size	{1024,2048,4096,8192}	2048
	Weight decay	(1e-7, 1e-4)	1e-5
	maximum size of embedding	5,10,15	10
	Dropout rate	(0, 1)	0.5
	Number of LSTM layers	{1,2,3,4}	3
	Size of LSTM hidden state	{40,50,100,200}	50
	Number of dense layers	{1,2,3}	2
	Size of penultimate hidden layer	{30,50,100,200}	50
	Size of last hidden layer	{5,10,20,40}	{5,10} <sup>2</sup>
Epochs	up to 4000	Varies <sup>1</sup>	

<sup>1</sup>early stopping based.

<sup>2</sup>based on output size.

with a prediction horizon shorter than three hours. However, the error increases rapidly when the prediction horizon is more than six hours, as shown in Table 4. On the other hand, an increase in lookback length does not show a major change in the model performance, but the lookback length of six steps shows the best performance in our experimental setting (Table 4).

In Table 4, we compare the model’s performance (with lookback length and prediction horizon of six steps and one step, respectively) with the different feature compositions. We show the effect of step-wise addition of features from the loop detectors, namely, the occupancy ( $\rho$ ) and flow ( $q$ ), to the exogenous set of features (link attributes, speed attributes). We

TABLE 3 Model performance on different metrics

Model	Link types	Loss criteria	Performance Metric	Training data	Test data
XGBoost	All	SMAPE (%)	SMAPE (%)	45.15 ± 2.02	51.76 ± 5.28
	Trunk		SMAPE (%)	14.04 ± 1.39	21.65 ± 2.93
			RMSE	725 ± 88	862 ± 157
LSTM	All	SMAPE (%)	SMAPE (%)	40.75 ± 0.51	<b>40.17 ± 0.90</b>
	Trunk		SMAPE (%)	14.05 ± 0.47	<b>16.89 ± 0.31</b>
			RMSE	634 ± 19	<b>743 ± 14</b>

Note: RMSE is not reported for link types “all”, since the scale of target variable largely varies across the primary, secondary, and trunk link types.

find that the model SMAPE reduces by approx. 41% when we add  $\rho$  to the input features. When we use the past value(s) of  $q$  as an input feature, the model formulation resembles autoregressive forecasting with exogenous inputs. In this setting, SMAPE reduces by more than 61% over the baseline indirect estimation model. This improvement is expected past target variables provide direct information about the scale or range for future predictions due to the autocorrelation among the target variables. Thus, the autoregressive forecasting setting provides more accurate predictions than the purely exogenous or indirect estimation setting, indicating that the latter is more challenging.

In Figure 11, we show specimens of the model predictions for different detectors. There are a few noticeable things. First, the model can capture the flow periodicity, that is, the ascent and descent of flow trend during the day. This signifies good predictions when the flow transitions from off-peak to peak flow and vice-versa. The model shows good performance for detectors 5380 and 5299, as the predicted peak flow is closer to the true (actual or measured) value. The model performs reasonably well with a SMAPE of less than 17% on the test data, considering exclusive exogenous input features. In a few instances, the model struggles to capture peak and off-peak flow (e.g. crests for detector 5169 and troughs for detector 5273), where the model either over-predicts or under-predicts the flow compared to the actual value. The dropout during the model testing

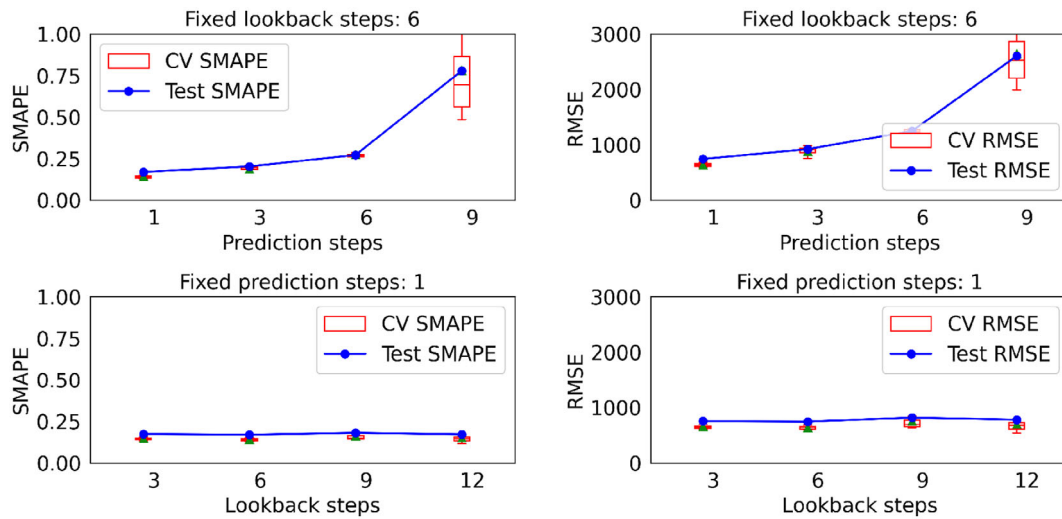


FIGURE 10 Cross-validation error and test error for different performance metrics, lookback length and prediction step

TABLE 4 Effect of lookback length and prediction horizon on test data. Best test performance is shown in bold

Model - Input features	Lookback length (hour)	Prediction horizon (hour)	RMSE
			(vehicles/hour)
LSTM - (exogenous)	6	1	<b>16.89 ± 0.31</b> <b>743 ± 14</b>
	6	3	20.20 ± 1.06    919 ± 42
	6	6	27.18 ± 0.60    1247 ± 20
	6	9	77.80 ± 16.45    2602 ± 317
	3	1	17.30 ± 0.47    751 ± 24
	6	1	<b>16.89 ± 0.31</b> <b>743 ± 14</b>
	9	1	18.00 ± 1.38    815 ± 73
	12	1	17.11 ± 1.08    777 ± 56
LSTM - (exogenous, o)	6	1	9.95 ± 2.01    466 ± 92
LSTM - (exogenous, o, q)	6	1	6.47 ± 0.97    321 ± 37

Note: o, occupancy; q, time-lagged flow.

provides insights into the prediction uncertainty. For this, we show the 95% confidence interval of the model predictions. The uncertainty is higher near the peak flows, as seen from the wider prediction intervals at the peaks. During off-peak hours, the predictions have low variance. The uncertainty estimates are as important as the predictions since it helps understand how much the model predictions can be trusted.

We also show the distribution of the errors, namely, SMAPE and RMSE, across time of the day, weekday, and month to identify any error correlations. In Figure 12, it is seen that the mean SMAPE for night and morning hours (during 2100–0900 h) are higher than during the rest of the day. On the other hand, the RMSE during the corresponding hours is lower than the rest of the day. One of the plausible explanations is the small magnitude of the flows during the off-peak hours, which pushes the

SMAPE to higher values. Finally, the SMAPE and RMSE are almost constant across different weekdays. For August, errors are slightly higher than in the rest of the months, possibly due to distinct traffic patterns during the vacation period in Paris.

## 6.1 | Model transferability

In Table 5, we show the model performances on test data using high (0.65) and low (0.10) proportions of training data for the target domain. When using 65% of the target data for training, the baseline model with randomly initialized parameters achieves a SMAPE of 22.24%. Pre-trained model without fine-tuning the target data does not lead to accurate predictions, as evident from its higher SMAPE. However, selective fine-tuning of the pre-trained model on the target data helps achieve even better results than the new model. Out of the different combinations of unfrozen layers in the pre-trained model, the model with all the unfrozen layers achieves the best SMAPE of 20.5%, which is a marginal improvement of 8% over the baseline model. A model with only LSTM layers as unfrozen layers also performs well with a SMAPE of 21.49%. Thus, fine-tuning the LSTM layer is essential when transferring the knowledge from the source to the target domain. This is due to the difference in temporal patterns between the target and the source city. Thus, the model relearns the new patterns from the target dataset.

The benefits of transfer learning are prominent in low data availability scenarios. When using 10% of the target data for training, the new model has a high bias and variance, as seen from its higher SMAPE and 95% confidence interval. Pre-trained model (with fine-tuned LSTM layer) can achieve SMAPE of 27.4% even in the case of data insufficiency. In Figure 13, we show the performance differences between the baseline or new model and fine-tuned pre-trained model with the different proportions (from 0.04 to 0.96 at a spacing of 0.04) of the target data used for fine-tuning. We fine-tune only the



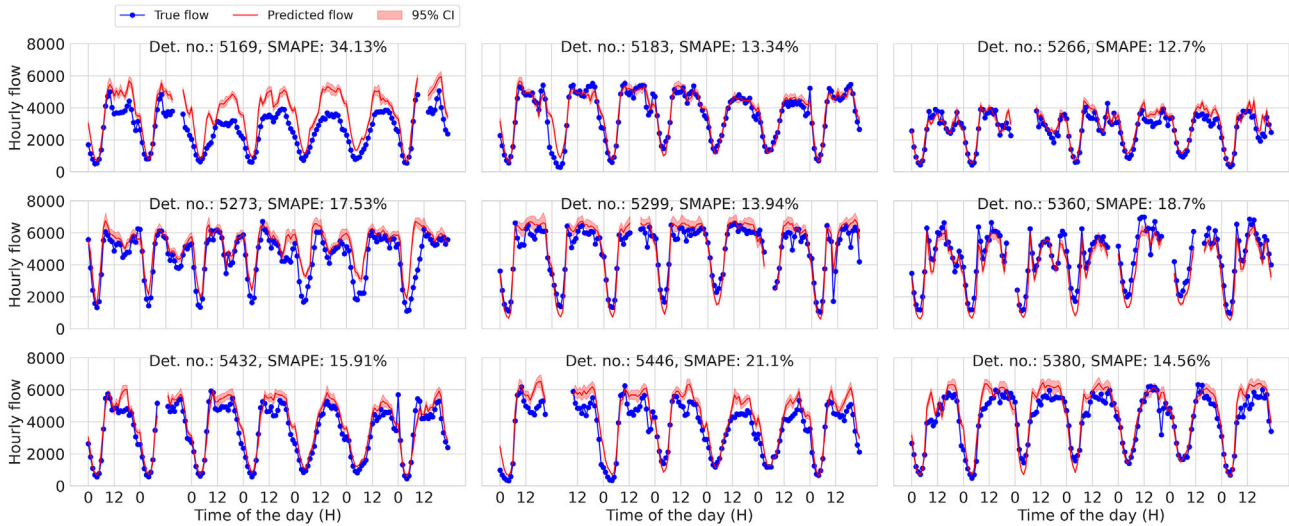


FIGURE 11 Examples showing flow predictions for detectors in test data from Paris

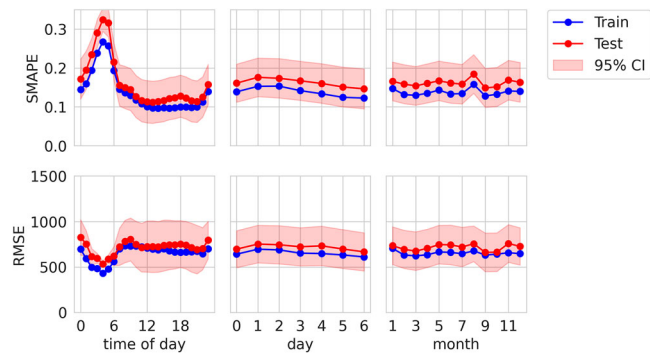


FIGURE 12 Trend of SMAPE and RMSE with time of day, weekday, and month

LSTM layer since it is crucial for successful transfer learning. We find that with sufficient training data, that is, when more than 40% of the target domain data is used for re-training, both models perform equally well with the test SMAPE of around 20%. When the proportion of training data falls below 40%, we notice two trends. First, the variance of the performance of the baseline model increases considerably. This is due to the high variance in sampling training datasets at low proportions since smaller training datasets do not capture complete distribution over the target domain. In contrast, the variance of the fine-tuned model is low and about consistent, which points to the advantage of transfer learning over the baseline model.

Second, when the proportion of training data is very low such as less than 20% (Figure 13), the baseline model's bias also increases, and it performs poorly compared to the pre-trained model. This is due to the model's over-fitting of the small training data distribution, which is very different from the test data distribution. In contrast, the performance of the fine-tuned pre-trained model is stable. Thus, we conclude that transfer

learning performs equally well when the data is sufficient. More importantly, transfer learning outperforms the baseline model when the data for the target domain is insufficient since source knowledge helps overcome the lack of data in the target domain.

Figure 14 shows example predictions on test target data from the new and pre-trained models at different training data proportions. When using less than 10% of the training data, the pre-trained model's predictions are more accurate and consistent than the new model's predictions. In these examples, when using insufficient training data, high bias and high variance in the predictions by the new model are evident. An increase in training data helps the model to make accurate predictions. But the pre-trained model can make accurate predictions using even a small amount of data.

## 7 | DISCUSSION

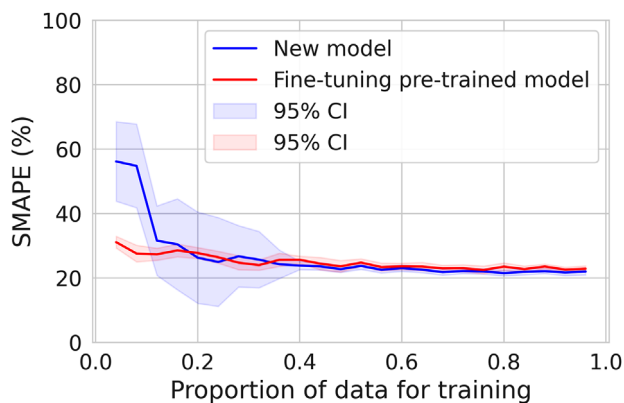
The objective of the proposed framework is to predict traffic counts on links that are not equipped with a traffic sensor by training and testing the model on the source links and then transferring the model to target links. The transferred model can help when the data is insufficient for developing models from scratch. LSTM model outperforms the XGBoost model in exogenous flow prediction with a test SMAPE of about 17% for the source trunk links. The trunk type links belong to high-speed category links, and thus our results somewhat address the limitation of previous work by Neumann et al. (2013) [7], where the model predictions for high-speed links were found to be less accurate. It is fundamental not only to predict these values but, even more notably, to estimate how precise these predictions are. The proposed deep learning architecture also answers this question using a dropout mechanism. Even though all link types exhibit typical MFD curves to varying extents, the LSTM model only fits well for trunk-type links. Models did not



**TABLE 5** Performance comparison between training new model and fine-tuning pre-trained model with the change in the proportion of training data.

LSTM Model-type	Weight initialization	Proportion of target data for training	Unfrozen/fine-tuned layers	Test SMAPE (%)	Improvement over baseline (%)
Baseline	Random	0.65	All	22.24 ± 1.96	–
			None	65.82 ± 2.74	–195
			FC3	46.54 ± 0.39	–109
			FC2-3	46.55 ± 0.34	–109
			FC1-3	45.87 ± 0.29	–106
			LSTM, FC	21.24 ± 0.99	4
			LSTM	21.49 ± 0.90	3
			E	22.75 ± 0.73	–2
			<b>E, LSTM, FC</b>	<b>20.50 ± 1.10</b>	<b>8</b>
Transfer	Pre-trained	0.10	All	55.30 ± 16.06	–
			None	65.82 ± 2.74	–19
			FC3	47.15 ± 1.30	15
			FC2-3	47.29 ± 1.00	15
			FC1-3	47.60 ± 1.00	14
			LSTM, FC	29.07 ± 1.70	47
			<b>LSTM</b>	<b>27.41 ± 1.68</b>	<b>50</b>
			E	29.14 ± 1.61	47
			E, LSTM, FC	30.92 ± 1.89	44

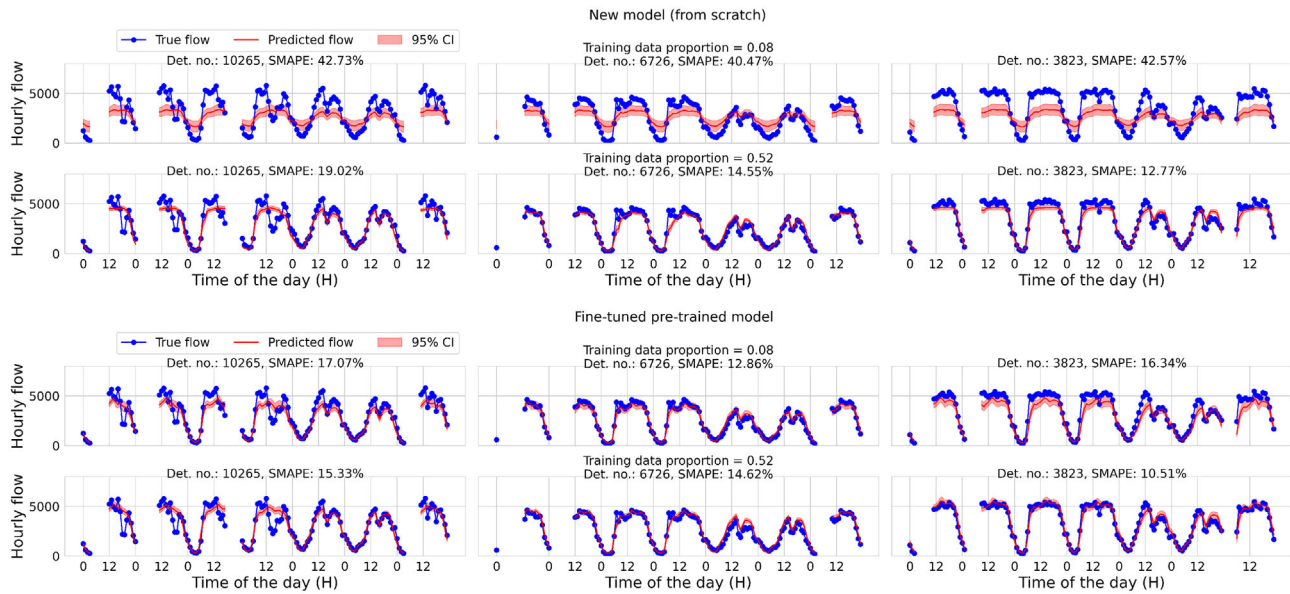
Note: FC, fully connected layer; E, embedding layer.

**FIGURE 13** Comparison between training new model and fine-tuning pretrained model with the change in the proportion of training data

perform as well in the case of primary and secondary type links, possibly due to several reasons. If a fundamental diagram is not well-formed, models will struggle to learn link speed and flow mapping. Further, unreliable FCD speed data in specific ranges of traffic variables will degrade the model performance. We conclude that the manifestation of a traffic fundamental diagram and reliable FCD data over a wide range of speed and flow is essential for indirect dynamic flow estimation from speed.

Our experiments conclude that the indirect traffic flow estimation task has two components (a) transferable patterns and (b) nontransferable patterns. Training a new model on minimal target data will lead to high bias and high variance in predictions. Thus, transfer learning can help to bridge this gap. Distributions of source and target data are prone to be distinct across different cities. Thus, applying a pre-trained model without fine-tuning on target data does not give accurate results. The pre-trained model contains the insights from the source domain, thus eliminating the need for a model to re-learn the transferable patterns. For learning the nontransferable patterns, some data is still required. Nevertheless, the overall data requirements are lesser than the scenario without transfer learning, and the pre-trained model outperforms the newly trained model when data is scarce.

The study's limitations can be categorized into relating to either data or model. Data issues such as data inaccuracy due to noise and anomalies, non-perfect data matching, information loss due to aggregation, and heterogeneity of data sources can lead to distortion of the same information coming from different sources. This directly affects the quality of the training data and decreases the signal-to-noise ratio, thus making it challenging for the model to learn the underlying correlations. Further, the static features such as maximum speed and the number of lanes ignore the effect of dynamic traffic management (such as dynamic speed limits or lane closures) in the recorded flow values.



**FIGURE 14** Flow predictions on test data in target domain using new model and fine-tuned pre-trained model. Both models are trained using different training proportions of training data

On the model side, even though we tackled temporal correlations using LSTM-based architecture, which is still competitive, there is still scope for further research using state-of-the-art deep learning models. For instance, GNNs [64, 65] or Temporal Fusion Transformer [66, 67] are more specialized to learn network or topological data and temporal sequences, respectively. Therefore, they can further help to reduce the forecasting error. Lastly, our model does not establish causation between link flows and link speeds but only uses their correlation for prediction.

The practical implications of our research are that the exogenous flow prediction can help fill the flow data unavailability for traffic management or transport model validation. Further, open data and transfer learning can help address this challenge by reducing data acquisition costs. Flow predictions with uncertainty estimates can help reduce the practitioners' hesitancy to apply these models.

## 8 | CONCLUSION

In this work, we developed the indirect traffic state estimation model (for predicting flow). Our model uses only exogenous features as input for the prediction. This is motivated by the fact that traffic flow data are sparse and the need to infer it from other readily available data. We collected the publicly available traffic data from heterogeneous sources for Paris and Madrid to form a year-long longitudinal traffic dataset. Using data from Paris, we trained an LSTM model, which performs well when the fundamental diagram is well-formed, as in the case of trunk-type links. We show that pre-trained models outperform the new model using transfer learning when the data for the target task is insufficient. The pre-trained model needs minimal data to make accurate predictions. Thus transfer learning and indirect

flow estimation can help to tackle the traffic flow data scarcity in transport modeling and traffic management applications.

We aim to integrate the model and predicted forecasts for improving transport demand model calibration. Specifically, the link flows provided by the model increase the transport network observability; thus, we aim to analyze their impacts on the calibrated demand estimates in future research. Here only reliable forecasts can be used, whereas the rest can be discarded. Special events, planned or unplanned, lead to changes in the traffic patterns [2] and thus can be added as an additional feature to improve the model performance. Future works should explore other data sources such as real-time traffic updates and extract the relevant features to augment the training data. Using the enriched data, it can be possible to apply transfer learning for long-term flow estimation like daily traffic flow estimation [68] or Annual Average Daily Traffic (AADT) flows. Another challenging work is investigating additional features, especially for the primary and secondary type links, to address their scatter in their fundamental diagram. Additional features that help address the scatter can help obtain accurate predictions, especially for lower category link types (primary and secondary). Additional features could help explain variance in the fundamental diagram and thus provide an improved signal to train the model.

## CREDIT AUTHORSHIP CONTRIBUTION STATEMENT

Vishal Mahajan: Conceptualization, Data curation, Formal analysis, Methodology, Software, Validation, Visualization, Writing - original draft. Guido Cantelmo: Conceptualization, Methodology, Supervision, Writing - review & editing. Raoul Rothfeld: Methodology, Supervision, Writing - review & editing. Constantinos Antoniou: Conceptualization, Funding acquisition, Supervision, Writing - review & editing.

## ACKNOWLEDGEMENTS

This work was supported by the Deutsche Forschungsgemeinschaft (DFG) under Grant 415208373 for Project TraMPA-Transport Modelling using Publicly Available Data. The authors want to sincerely thank the three anonymous reviewers for taking the time to provide their valuable comments and helping to enrich the manuscript.

Open Access funding enabled and organized by Projekt DEAL.



## CONFLICT OF INTEREST

No potential competing interest was reported by the authors. On behalf of all authors, the corresponding author states that there is no conflict of interest.

## DATA AVAILABILITY STATEMENT

The data that support the findings of this study were derived from the following resources available in the public domain: at [52], [9] and [54]. Further, the derived data are available from the corresponding author upon reasonable request.

## ORCID

Vishal Mahajan  <https://orcid.org/0000-0003-2131-2831>  
Constantinos Antoniou  <https://orcid.org/0000-0003-0203-9542>

## REFERENCES

- Vlahogianni, E.I., Karlaftis, M.G., Golias, J.C.: Short-term traffic forecasting: Where we are and where we're going. *Transport. Res. Part C: Emerg. Technol.* 43, 3–19 (2014). ISSN 0968-090X. <https://doi.org/10.1016/j.trc.2014.01.005>. <https://www.sciencedirect.com/science/article/pii/S0968090X14000096>. Special Issue on Shortterm Traffic Flow Forecasting
- Polson, N.G., Sokolov, V.O.: Sokolov. Deep learning for short-term traffic flow prediction. *Transport. Res. Part C: Emerg. Technol.* 79, 1–17 (2017). ISSN 0968-090X. <https://doi.org/10.1016/j.trc.2017.02.024>. <https://www.sciencedirect.com/science/article/pii/S0968090X17300633>
- Lopes, J., Bento, J., Huang, E., Antoniou, C., Ben-Akiva, M.: Traffic and mobility data collection for real-time applications. In: 13th International IEEE Conference on Intelligent Transportation Systems, pp. 216–223. IEEE, Piscataway (2010)
- Barmounakis, E., Geroliminis, N.: On the new era of urban traffic monitoring with massive drone data: The pneuma large-scale field experiment. *Transport. Res. Part C: Emerg. Technol.* 111, 50–71 (2020). ISSN 0968-090X. <https://doi.org/10.1016/j.trc.2019.11.023>. <https://www.sciencedirect.com/science/article/pii/S0968090X19310320>
- Ma, T., Antoniou, C., Toledo, T.: Hybrid machine learning algorithm and statistical time series model for networkwide traffic forecast. *Transport. Res. Part C: Emerg. Technol.* 111, 352–372 (2020). ISSN 0968-090X. <https://doi.org/10.1016/j.trc.2019.12.022>. <https://www.sciencedirect.com/science/article/pii/S0968090X19303821>
- California Department of Transportation: Pems user guide. Technical Report Version 6, 02 (2020). [https://pems.dot.ca.gov/Papers/PeMS\\_Intro\\_User\\_Guide\\_v6.pdf](https://pems.dot.ca.gov/Papers/PeMS_Intro_User_Guide_v6.pdf)
- Neumann, T., Bühnke, P.L., Touko-Tcheumadjeu, L.C.: Dynamic representation of the fundamental diagram via bayesian networks for estimating traffic flows from probe vehicle data. In 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013), pp. 1870–1875. IEEE, Piscataway (2013). <https://doi.org/10.1109/ITSC.2013.6728501>
- Aslam, J., Lim, S., Pan, X., Rus, D.: City-scale traffic estimation from a roving sensor network. In Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems, SenSys '12, pp. 141–154. Association for Computing Machinery, New York (2012). <https://doi.org/10.1145/2426656.2426671>
- Uber Movement: Movement cities. <https://movement.uber.com/> and <https://www.npmjs.com/package/movement-data-toolkit>. Accessed: 2021-09-02
- TomTom: Traffic stats. <https://www.tomtom.com/products/traffic-stats/>, 2021. Accessed: 2021-09-02.
- Mahajan, V., Kuehnel, N., Intzevidou, A., Cantelmo, G., Moeckel, R., Antoniou, C.: Data to the people: a review of public and proprietary data for transport models. *Transport Rev.* 1–26 (2021). <https://doi.org/10.1080/01441647.2021.1977414>
- Vlahogianni, E.I., Golias, J.C., Karlaftis, M.G.: Short-term traffic forecasting: Overview of objectives and methods. *Transport Rev.* 24(5), 533–557 (2004). <https://doi.org/10.1080/0144164042000195072>
- Tedjopurnomo, D.A., Bao, Z., Zheng, B., Choudhury, F., Qin, A.K.: A survey on modern deep neural network for traffic prediction: Trends, methods and challenges. *IEEE Trans. Knowl. Data Eng.* 1–1 (2020). <https://doi.org/10.1109/TKDE.2020.3001195>
- Wang, S., Cao, J., Yu, P.: Deep learning for spatio-temporal data mining: A survey. *IEEE Trans. Knowl. Data Eng.* (01), 1–1 (2019). ISSN 1558-2191. <https://doi.org/10.1109/TKDE.2020.3025580>
- Hochreiter, S.: Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München, (1991)
- Laptev, N.P., Yosinski, J., Li, L.E., Smyl, S.: Time-series extreme event forecasting with neural networks at Uber. (2017)
- Cui, Z., Ke, R., Pu, Z., Wang, Y.: Stacked bidirectional and unidirectional lstm recurrent neural network for forecasting network-wide traffic state with missing values. *Transport. Res. Part C: Emerg. Technol.* 118, 102674–09 (2020). <https://doi.org/10.1016/j.trc.2020.102674>
- Lara-Benítez, P., Carranza-García, M., Riquelme, J.C.: An experimental review on deep learning architectures for time series forecasting. *Int. J. Neural Syst.* 31(03), 2130001 (2021). <https://doi.org/10.1142/S0129065721300011>. PMID: 33588711.
- Wu, T., Chen, F., Wan, Y.: Graph attention lstm network: A new model for traffic flow forecasting. In 2018 5th International Conference on Information Science and Control Engineering (ICISCE), pp. 241–245. (2018). <https://doi.org/10.1109/ICISCE.2018.00058>
- Wang, X., Chen, C., Min, Y., He, J., Yang, B., Zhang, Y.: Efficient metropolitan traffic prediction based on graph recurrent neural network (2018)
- Li, Y., Yu, R., Shahabi, C., Liu, Y.: Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. arXiv preprint arXiv:1707.01926 (2017)
- Zhao, L., Song, Y., Zhang, C., Liu, Y., Wang, P., Lin, T., et al.: T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE Trans. Intell. Transp. Syst.* 21(9), 3848–3858 (2020). <https://doi.org/10.1109/TITS.2019.2935152>
- Buroni, G., Lebichot, B., Bontempi, G.: Ast-ml: An attention-based multi-task learning strategy for traffic forecasting. *IEEE Access* 9, 77359–77370 (2021). <https://doi.org/10.1109/ACCESS.2021.3083412>
- Seo, T., Bayen, A.M., Kusakabe, T., Asakura, Y.: Traffic state estimation on highway: A comprehensive survey. *Ann. Rev. Control* 43, 128–151 (2017). ISSN 1367-5788. <https://doi.org/10.1016/j.arcontrol.2017.03.005>. <https://www.sciencedirect.com/science/article/pii/S1367578817300226>
- Lefebvre, N., Chen, X., Beuseroy, P., Zhu, M.: Traffic flow estimation using acoustic signal. *Eng. Appl. Artif. Intell.* 64, 164–171 (2017). ISSN 0952-1976. <https://doi.org/10.1016/j.engappai.2017.05.019>. <https://www.sciencedirect.com/science/article/pii/S0952197617301197>
- Greenshields, B.D., Bibbins, J.R., Channing, W.S., Miller, H.H.: A study of traffic capacity. In: Highway Research Board Proceedings 1935. Transportation Research Board, Washington DC (1935)
- Kühne, R.: Foundation of traffic flow theory i: Greenshields legacy highway traffic. In TRB Traffic Flow Theory and Characteristics Committee Summer Meeting and Symposium on the Fundamental Diagram: 75 years ("Greenshields75" Symposium). Transportation Research Board, Washington DC (2008). <https://elib.dlr.de/54950/>



28. Kumarage, S.P., Rajapaksha, R.P.G.K.S., De Silva, D., Bandara, J.M.S.J.: Traffic flow estimation for urban roads based on crowdsourced data and machine learning principles. In: Kováčiková, T., Buzna, Ľ., Pourhashem, G., Lugano, G., Cornet, Y., Lugano, N. (eds.) *Intelligent Transport Systems – From Research and Development to the Market Uptake*, pp. 263–273. Springer International Publishing, Cham (2018)
29. Pun, L., Zhao, P., Liu, X.: A multiple regression approach for traffic flow estimation. *IEEE Access* 7, 35998–36009 (2019). <https://doi.org/10.1109/ACCESS.2019.2904645>
30. Gkountouna, O., Pfoser, D., Züfle, A.: Traffic flow estimation using probe vehicle data. In: 2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA), pp. 579–588. IEEE, Piscataway (2020). <https://doi.org/10.1109/DSAA49011.2020.00073>
31. Rinaldi, M., Viti, F.: A cascading kalman filtering framework for real-time urban network flow estimation. In: 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), pp. 1–6. IEEE, Piscataway (2020). <https://doi.org/10.1109/ITSC45102.2020.9294175>
32. Zhang, Z., Li, M., Lin, X., Wang, Y.: Network-wide traffic flow estimation with insufficient volume detection and crowdsourcing data. *Transport. Res. Part C: Emerg. Technol.* 121, 102870 (2020). <https://doi.org/10.1016/j.trc.2020.102870>. <https://www.sciencedirect.com/science/article/pii/S0968090X20307701>
33. Abdelraouf, A., Abdel Aty, M., Mahmoud, N.: Sequence-to-sequence recurrent graph convolutional networks for traffic estimation and prediction using connected probe vehicle data. *IEEE Trans. Intell. Transp. Syst.* 1–11 (2022). <https://doi.org/10.1109/TITS.2022.3168865>
34. Zhu, W., Chang, A., Jiang, G., Zhang, W.: Link average speed of traffic flow estimation method based on floating car. In: Ninth International Conference of Chinese Transportation Professionals, pp. 1–6. American Society of Civil Engineers, Reston, VA (2009). [https://doi.org/10.1061/41064\(358\)229](https://doi.org/10.1061/41064(358)229).
35. Pan, S.J., Yang, Q.: A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* 22(10), 1345–1359 (2010). <https://doi.org/10.1109/TKDE.2009.191>
36. Torrey, L., Shavlik, J.: Transfer learning. In: *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, pp. 242–264. IGI Global, Hershey, PA (2010). <https://doi.org/10.4018/978-1-60566-766-9.ch011>.
37. Luan, J., Guo, F., Polak, J., Hoose, N., Krishnan, R.: Investigating the transferability of machine learning methods in short-term travel time prediction. In: Transportation Research Board 97th Annual Meeting. Transportation Research Board, Washington DC (2018)
38. Li, J., Guo, F., Sivakumar, A., Dong, Y., Krishnan, R.: Transferability improvement in short-term traffic prediction using stacked lstm network. *Transport. Res. Part C: Emerg. Technol.* 124, 102977 (2021). ISSN 0968-090X. <https://doi.org/10.1016/j.trc.2021.102977>. <https://www.sciencedirect.com/science/article/pii/S0968090X21000140>
39. Mallick, T., Balaprakash, P., Rask, E., Macfarlane, J.: Transfer learning with graph neural networks for short-term highway traffic forecasting. In: 2020 25th International Conference on Pattern Recognition (ICPR), pp. 10367–10374. IEEE, Piscataway (2021)
40. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., et al.: Pytorch: An imperative style, high-performance deep learning library. In: *NIPS'19: Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pp. 8024–8035. Curran Associates, Inc., Red Hook (2019). <http://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>
41. Torch Contributors: Lstm. <https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html>. Accessed: 2021-09-02.
42. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* 15(56), 1929–1958 (2014). <http://jmlr.org/papers/v15/srivastava14a.html>
43. Gal, Y., Ghahramani, Z.: Dropout as a bayesian approximation: Representing model uncertainty in deep learning (2016)
44. Shwartz-Ziv, R., Armon, A.: Tabular data: Deep learning is not all you need. *CoRR abs/2106.03253* (2021). <https://arxiv.org/abs/2106.03253>
45. Chen, T., Guestrin, C.: XGBoost: A scalable tree boosting system. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16*, pp. 785–794. ACM, New York (2016). <https://doi.org/10.1145/2939672.2939785>
46. Hastie, T., Tibshirani, R., Friedman, J.: *The elements of statistical learning*. In: *Springer Series in Statistics*. Springer, New York (2001)
47. Nogueira, F.: *Bayesian Optimization: Open source constrained global optimization tool for Python* (2014). <https://github.com/fmfn/BayesianOptimization>
48. Redko, K.: Time series forecast error metrics you should know. <https://towardsdatascience.com/time-series-forecast-error-metrics-you-should-know-cc88b8c67f27>. Accessed: 2021-12-02
49. Goodwin, P., Lawton, R.: On the asymmetry of the symmetric mape. *Int. J. Forecast.* 15(4), 405–408 (1999). [https://doi.org/10.1016/S0169-2070\(99\)00007-2](https://doi.org/10.1016/S0169-2070(99)00007-2). <https://www.sciencedirect.com/science/article/pii/S0169207099000072>
50. Redko, I., Habrard, A., Morvant, E., Sebban, M., Bennani, Y.: 2 - Domain adaptation problem. In: Redko, I., Habrard, A., Morvant, E., Sebban, M., Bennani, Y. (eds.) *Advances in Domain Adaptation Theory*, pp. 21–36. Elsevier, London (2019). <https://doi.org/10.1016/B978-1-78548-236-6.50002-7>. <https://www.sciencedirect.com/science/article/pii/B9781785482366500027>
51. Boeing, G.: Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Comput. Environ. Urban Syst.* 65, 126–139 (2017). <https://doi.org/10.1016/j.compenvurbusys.2017.05.004>. <https://www.sciencedirect.com/science/article/pii/S0198971516303970>
52. Open Data Paris: Comptage routier - données trafic issues des capteurs permanents. <https://opendata.paris.fr/explore/dataset/comptages-routiers-permanents>. Accessed: 2021-09-02
53. Uber Movement: Uber movement: Speeds calculation methodology. <https://movement.uber.com/faqs?lang=en-US>
54. Open Data Madrid: Tráfico. histórico de datos del tráfico desde 2013. <https://datos.madrid.es/portal>. Accessed: 2022-04-05
55. Sharedstreets: sharedstreets-js (2018). <https://github.com/sharedstreets/sharedstreets-js>. Accessed: 2021-09-02.
56. Sharedstreets: sharedstreets-matcher (2017). <https://github.com/sharedstreets/sharedstreets-matcher>. Accessed: 2021-09-02
57. OpenStreetMap: Highway link. <https://wiki.openstreetmap.org/wiki/Key:highway/>. Accessed: 2022-09-02
58. OpenStreetMap: Highway link. [https://wiki.openstreetmap.org/wiki/Highway\\_link/](https://wiki.openstreetmap.org/wiki/Highway_link/). Accessed: 2021-09-02.
59. Daganzo, C.F., Geroliminis, N.: An analytical approximation for the macroscopic fundamental diagram of urban traffic. *Transport. Res. Part B: Methodolog.* 42(9), 771–781 (2008). <https://doi.org/10.1016/j.trb.2008.06.008>. <https://www.sciencedirect.com/science/article/pii/S0191261508000799>
60. Mahmassani, H.S., Saberi, M., Zockaie, A.: Urban network gridlock: Theory, characteristics, and dynamics. *Transport. Res. Part C: Emerg. Technol.* 36, 480–497 (2013). <https://doi.org/10.1016/j.trc.2013.07.002>. <https://www.sciencedirect.com/science/article/pii/S0968090X13001551>
61. Geroliminis, N., Daganzo, C.F.: Existence of urban-scale macroscopic fundamental diagrams: Some experimental findings. *Transport. Res. Part B: Methodolog.* 42(9), 759–770 (2008). <https://doi.org/10.1016/j.trb.2008.02.002>. <https://www.sciencedirect.com/science/article/pii/S0191261508000180>
62. Geroliminis, N., Sun, J.: Properties of a well-defined macroscopic fundamental diagram for urban traffic. *Transport. Res. Part B: Methodolog.* 45(3), 605–617 (2011). <https://doi.org/10.1016/j.trb.2010.11.004>. <https://www.sciencedirect.com/science/article/pii/S0191261510001372>
63. Chen, X., Zhang, C., Zhao, X.L., Saunier, N., Sun, L.: Nonstationary temporal matrix factorization for multivariate time series forecasting (2022). <https://arxiv.org/abs/2203.10651>
64. Jiang, W., Luo, J.: Graph neural network for traffic forecasting: A survey. *CoRR abs/2101.11174* (2021). <https://arxiv.org/abs/2101.11174>
65. Lin, L., He, Z., Peeta, S.: Predicting station-level hourly demand in a large-scale bike-sharing network: A graph convolutional



- neural network approach. *Transport. Res. Part C: Emerg. Technolog.* 97, 258–276 (2018). <https://doi.org/10.1016/j.trc.2018.10.011>. <https://www.sciencedirect.com/science/article/pii/S0968090X18300974>
66. Lim, B., ArÅšık, S.O., Loeff, N., Pfister, T.: Temporal fusion transformers for interpretable multi-horizon time series forecasting. *Int. J. Forecast.* 37(4), 1748–1764 (2021). <https://doi.org/10.1016/j.ijforecast.2021.03.012>. <https://www.sciencedirect.com/science/article/pii/S0169207021000637>
67. Beitner, J.: PyTorch Forecasting: Time series forecasting with PyTorch (2020). <https://pytorch-forecasting.readthedocs.io/>
68. Ma, D., Song, X., Li, P.: Daily traffic flow forecasting through a contextual convolutional recurrent neural network modeling inter- and intra-day

traffic patterns. *IEEE Trans. Intell. Transp. Syst.* 22(5), 2627–2636 (2021). <https://doi.org/10.1109/TITS.2020.2973279>

**How to cite this article:** Mahajan, V., Cantelmo, G., Rothfeld, R., Antoniou, C.: Predicting network flows from speeds using open data and transfer learning. *IET Intell. Transp. Syst.* 17, 804–824 (2023). <https://doi.org/10.1049/itr2.12305>