TraDWin: An interactive Digital Twin for City Traffic

Shreyansh Yadav, Vignesh Sivaraman, *Member, IEEE* and Vignesh Sridharan, *Member, IEEE*

***Abstract*—In the contemporary world characterized by rapid urbanization and burgeoning populations, cities worldwide confront pressing challenges, notably the effective management of traffic congestion and the optimization of transportation infrastructure. The advent of smart cities facilitated by 5G and the Internet of Things (IoT) has unlocked unprecedented opportunities to harness real-time data. This technological advancement has spurred the adoption of sophisticated cyberphysical systems such as Digital Twins, which are increasingly recognized for their potential in modeling urban traffic dynamics with precision.**

This study introduces TraDWin, an end-to-end framework for a Traffic Digital Twin designed to seamlessly aggregate real-time traffic data sourced from diverse sensors, cameras, GPS devices, and analogous instruments. By leveraging physics-informed deep machine learning models, our framework represents a departure from traditional empirical approaches reliant on laborious and resource-intensive micro-simulations. TraDWin integrates comprehensive datasets encompassing road network configurations, traffic volumes at distinct junctures, and contextual parameters including road characteristics and meteorological conditions. Trained separately for each, it is able to address three pivotal aspects of urban traffic state estimation: (i) prediction of future traffic conditions, (ii) imputation of traffic state data for segments with missing information, and (iii) traffic assignments in response to evolving map topologies, crucial for scenarios such as road closures or urban development initiatives. The framework ensures compatibility with the different possible data sources you may have as well as re-learning based on new emerging patterns. Finally, we evaluate our proposed model using traffic volume data obtained from Dublin city's SCATS traffic management system, alongside simulation-based assessments using the SUMO platform to replicate pertinent scenarios.

***Index Terms*—Digital Twins, Traffic, Traffic state estimation, Traffic imputation, Traffic prediction, Urban Planning, Physics Informed Model**

I. Introduction

The contemporary landscape of urban development is characterized by rapid urbanization and increasing population densities, presenting formidable challenges in optimizing traffic flow and urban infrastructure planning. Extensive research has documented the multifaceted impacts of traffic on public health [1], economic viability [2], and social dynamics [3], highlighting the urgent need for advanced technological interventions.

The Digital Twin, as an emerging concept within cyberphysical systems (CPS), has garnered significant attention over the past decade, particularly with the advancements in big data, IoT connectivity, and affordable computing, which have rendered such systems increasingly practical [4] [5]. A Digital Twin is a virtual representation of a real-world object, system, or process, meticulously designed to replicate its physical counterpart in the digital realm [6]. This enables the Digital Twin to capture and simulate the intricate details of a physical entity, facilitating real-time monitoring, analysis, and prediction of its behavior. It transcends traditional 3D modeling by incorporating live data, sensor inputs, and advanced analytics, thereby providing a dynamic and interactive digital mirror of the physical world [6]. This capability has found applications across a multitude of fields, including manufacturing, engineering, healthcare, urban planning, and beyond.

Our traffic digital twin, TraDWin, serves as an end-to-end framework for an interactive simulation of city traffic, continuously updating its internal state with real-time data from various sources and re-learning to better tackle emerging traffic patterns. The proliferation of smart cities, equipped with IoT sensors and live camera feeds, has led to the development of numerous methodologies to enhance this capability. For example, the Sydney Coordinated Adaptive Traffic System (SCATS) [7] employs inductive loop-based sensors at traffic signals to monitor traffic volumes and is operational in over 180 cities across 28 countries, including New Zealand, Dublin, Shanghai, and Hong Kong [8]. Cities such as New York and Los Angeles have implemented systems similar to SCATS. Additional techniques involve traffic probes [9], GPS-enabled mobile phones [10], exemplified by Google's provision of congestion and travel time data, and deep learning computer vision models that utilize cameras to identify and count vehicles. These methods, however, primarily provide absolute volume counts at various nodes, rather than more granular data such as source-destination pairs, which present logistical and privacy challenges. In our study, we rely exclusively on absolute traffic volume data, without incorporating any information related to individual vehicles.

In the contemporary literature review, as discussed subsequently, mathematical and deep learning time series analysis methods prevail for tasks (i) and (ii). However, for task (iii), microscopic traffic simulation software such as SUMO [11] and Vissim [12] are commonly utilized. There are also approaches, for example by Kušić et al. (2023) [13], that use a macroscopic simulation model to create a digital twin of the city transport, continuously syncing with existing traffic data and then running a simulation with those parameters. These simulation engines are computationally intensive and require

S. Yadav and V. Sivaraman are associated with the Department of Computer Science and Engineering, Indian Institute of Technology, (BHU) Varanasi, India.
V. Sridharan is associated with Institute for Infocomm Research (I^2R), A*STAR, Singapore.
(E-mail: shreyansh.yadav.cse19@itbhu.ac.in, vignesh.sivaraman@u.nus.edu and shreyansh.yadav.cse19@itbhu.ac.in)

detailed information on vehicle types, origins, and destinations to produce accurate results, which may pose challenges in data acquisition. Another notable limitation of previous approaches, which predominantly focus on time series analysis, is their exclusive consideration of traffic flow data without accounting for potential exogenous variables such as weather [14], holidays [15], and other contextual factors.

We argue that traffic dynamics are shaped by both intrinsic data relationships and external factors such as weather [14] and holidays [15]. Moreover, there may be latent relationships among these variables that are not immediately evident. Thus, our proposed TraDWin framework is designed to be highly adaptable, allowing for the inclusion of new features to enhance its predictive capacity.

In this paper, we aim to provide an end-to-end framework digital twin framework that first collects and aggregates real-time traffic data, updates its internal state parameters to reflect changes in the real world, along with use cases for consumption of that data like integration with city's traffic control systems and vehicle ADAS (Advanced Driver Assisted Systems). Our proposed framework utilizes the aggregated data to address three key tasks related to traffic modeling:

(i) **Traffic Prediction**: Predicting how traffic volume will change in the future based on a given traffic volume time series and historical data.
(ii) **Imputation**: For traffic volume time series with missing data (potentially due to sensor failure or other issues), our framework aims to accurately fill in the missing values.
(iii) **Re-assignment on Edge Addition or Removal**: Our objective is to predict how traffic flow will be affected by changes in the road network, such as the addition or removal of road segments.

For each of the tasks, the architecture of our model remains the same, though it is to be trained independently for each of them. Attempts at unifying it across all tasks using multi-task learning could be a future direction of research.

Our proposed framework (i) takes input a graph representation of map where different regions are the vertices of the graph and different roads form the edges of the graph. (ii) Next, it encodes the structural features of graph nodes using Node2Vec [16] and transforms them into feature embeddings. (iii) These embeddings are augmented with relevant traffic data and other semantic information, such as weather, encoded using Word2Vec [17]. (iv) This encoded representation of the graph is used as input to an informed Wasserstein distance-based Generative Adversarial Imputation Net (GAIN), a modified version of the GAIN [18], which is a type of conditional GAN. This adaptation incorporates elements from the Wasserstein GAN (WGAN) [19], enhancing GAN training effectiveness. Additionally, we integrate a modified loss function from physics-informed deep learning techniques [20], ensuring the model adheres to physical conservation laws. (v) This proposed model is trained using real-world data and the trained model is used compute the tasks of prediction, imputation, and re-assignment of traffic.

To summarize, we outline the our contributions as follows:

(i) We introduce an end-to-end workflow for an interactive Traffic Digital Twin system that utilizes traffic volume data as its primary input.
(ii) We highlight how this can be integrated with existing traffic based systems as a Digital Twin of the physical traffic state.
(iii) Our proposed framework considers contextual data like geographical, meteorological, temporal data that are aggregated according to physical conservation laws.
(iv) We mathematically capture the dynamics involved in traffic modeling using physical laws.
(v) Using this physics-informed traffic model, we develop a Wasserstein distance-based Generative Adversarial Imputation Net (GAIN) framework to predict and compute various tasks using real-world traffic data.

II. Related Work

First we discuss existing model twin concepts related to traffic modeling and smart cities in general. There have been several reviews [21] [22] and studies on application of digital twins for smart cities. Edge computing where each vehicle interacts with others is a common point but that since that requires elaborate and secure infrastructure to exist as precursor to deployment of such a twin, we forego use of any such technology in our study at the time. Another significant challenge is the integration of heterogeneous data sources. Smart cities rely on data collected from various systems, including sensors, IoT devices, and legacy systems, which often use incompatible formats and protocols, this is usually handled by a unifying centralized service. Security and transparency is another concern and use of blockchain [23] for it has been explored for it as well, but it doesn't apply to our case as we mostly use centralized services like traffic sensors and street cameras within the city's control. Application of such a twin in city planning [24] highlight the economic benefit of such a digital twin through simulation results of planned scenarios.

Next, we review the existing literature on traffic modeling, focusing on current spatio-temporal modeling methods. We compare these approaches with our framework, addressing recent work relevant to each of the specific tasks our model aims to perform: prediction, imputation, and prediction in response to changing topology, the latter being a distinctive feature of our work.

Autoregressive Integrated Moving Average (ARIMA) [25] and its variants, such as time-oriented ARIMA [26] and Seasonal ARIMA [27], [28], are widely used traditional algorithms for prediction and forecasting tasks. ARIMA operates on time series data and is frequently combined with other algorithms to incorporate spatial features. For instance, C. Xu et al. (2016) [29] integrated ARIMA with a genetic algorithm to estimate future traffic flow on roads. There are also hybrid approaches that combine this with machine learning models like LSTMs [30] making it more effective While ARIMA models are proficient in capturing linear time series data, the addition of genetic algorithms enhances their ability to extract features from nonlinear historical data. This integration demonstrates the versatility and adaptability of ARIMA-based

approaches in addressing complex prediction challenges. Consequently, we also compare our model's performance with that of the ARIMA model. But one major problem this method suffers from is the lack of spatial information about the road network, which is crucial for traffic prediction tasks.

Other methods, such as Temporal Regularized Matrix Factorization (TRMF) [31], Bayesian Temporal Regularized Matrix Factorization (BTRMF) [32], and Bayesian Temporal Matrix Factorization (BTMF) [32], have also been explored in the context of traffic prediction [33]. These methods extend the principles of matrix and tensor factorization to capture temporal dependencies and spatial correlations in high-dimensional traffic data.

TRMF, introduced by Lai et al. (2017), extends matrix factorization to tensor data structures, enabling the modeling of multi-dimensional traffic flow data with enhanced flexibility. Similarly, BTRMF, proposed by Liu et al. (2019), integrates Bayesian inference into tensor factorization to provide probabilistic predictions and uncertainty quantification in traffic forecasting tasks. BTMF and Bayesian Temporal Tensor Factorization (BTTF) further extend the Bayesian framework to matrix and tensor factorization, respectively, offering robust approaches for modeling temporal dynamics and spatial interactions in traffic networks while accounting for uncertainties in the prediction process.

These methods contribute to the diverse landscape of predictive models for traffic forecasting, each offering unique advantages and insights for accurately addressing the challenges of traffic flow prediction. But again, these methods are limited in their ability by the lack of spatial information about the road network, as was the case with ARIMA variants.

Using machine learning for the task provide for yet another approach at tackling the problem. In this domains, LSTMs are of particular interest in tasks related to time-series modeling. Specialized LSTM models like ConvLSTM [34] which uses a linear feature enhanced convolutional LSTM have been shown to be effective at network traffic prediction tasks in a Digital Twin but does not consider exogenous enviromental factors. DS-TFSN [35] based Digital Twin uses an LSTM model dividing the input into macro, micro and environmental factors similar to our. Though, LSTMs fall short is their ability to adapt to imputation and redistribution tasks.

Next, we explore GNNs which is a new class of models that try to address this limitation.

Graph Neural Networks (GNNs) [36] have emerged as a prominent tool for spatiotemporal modeling, demonstrating effectiveness in various applications such as city traffic forecasting and imputation [37] [38]. Originally starting with studies by Li et al. (2018) [39] and Zhang et al. (2019) [40] have originally highlighted their utility in accurately predicting traffic patterns and filling in missing data in traffic volume time series. They have also been effectively applied in the digital twin domain for predictive tasks [41], anomaly detection [42] etc. Latest research involves using specialized approaches for traffic like separate the diffusion and traffic state [43] information giving an improved accuracy. GNNs iteratively update node representations based on local neighborhood information, allowing them to capture complex spatial and temporal dependencies within graph-structured data.

However, a limitation of traditional GNNs is their reliance on a static network topology during training, which poses challenges when adapting them to dynamic graphs. This limitation is particularly relevant for tasks involving changes in the graph structure, such as predicting traffic flow when adding or removing edges, a crucial aspect of our research focus.

Physics-informed deep learning (PIDL) [44] is an emerging methodology wherein a neural network is trained to perform learning tasks while adhering to the principles of physics, as defined by physics-based constraint equations and general nonlinear partial differential equations. By embedding the fundamental laws of physics as biases during training, the model is not required to independently discover these dependencies. This approach enhances the data efficiency of the resultant neural network.

For traffic modeling, physics-informed deep learning (PIDL) provides an effective middle ground between purely data-driven models and purely model-driven methods and there are several promising approaches already like using LWR [45] and CTM [46] as physical model for TSE [20] [47]. Purely model-driven approaches use mathematical models and prior knowledge of traffic dynamics to estimate future states, assuming the model accurately represents real-world traffic. However, this assumption often fails to capture the intricate dynamics of real-world traffic, and multiple equally viable models can exist for the same task, making generalization challenging. Examples of model-driven traffic approaches include the Lighthill-Whitham-Richards (LWR) [45] model and the Cell Transmission Model (CTM) [46].

In contrast, pure deep learning approaches require large amounts of data to learn generalized relationships, as they lack pre-existing information on physical relations and constraints. PIDL combines both approaches by incorporating physics-based biases into the deep learning model through a parameter λ, while still allowing the model enough freedom to learn more granular relationships in traffic dynamics. This integration enables the model to leverage the strengths of both methodologies, enhancing its effectiveness in traffic modeling.

There's also macroscopic simulation approaches to the problem which involve using a simulation engine like Vissim [12] or SUMO [11] in a digital twin whose state is updated with real world data in real-time. Kušić et al. (2023) [13] explores such a digital twin with a framework for real-time syncing. Work by Saroj et al. (2021) [48] would be another. However, both suffer from not utilizing correlations between enviromental factors.

In summary, while existing methods have made significant strides in traffic prediction and imputation tasks, they often fall short in integrating spatial information about the road network, which is vital for capturing complex traffic dynamics. Traditional approaches like ARIMA and its variants, though effective in linear time series prediction, lack spatial awareness. Advanced matrix factorization techniques such as TRMF, BTRMF, and BTMF offer improved temporal modeling but still struggle with spatial information integration. GNNs have shown promise by leveraging graph structures to

model spatiotemporal dependencies; however, their inability to adapt to dynamic network topologies limits their applicability in scenarios involving changing road networks. Physics-informed deep learning (PIDL) provides a balanced approach by combining data-driven and model-driven methods, yet existing models have not fully explored their application to traffic dynamics in the context of changing network structures. Our research addresses this gap by focusing on predicting traffic flow in dynamic road networks, a novel problem area previously tackled primarily through simulation-based models like Vissim [12] and SUMO [11].

III. Problem Formulation

A road network can be considered an undirected graph $G = (V, E)$ where V represents nodes, which is a collection of N detector nodes, i.e., $V = \{v_1, v_2, ..., v_n\}$ where $N = |V|$ and v_i is the ith detector node. E represents the $N_E = |E|$ road links connecting the detectors, and let $A \in \mathbb{R}^{N \times N}$ denote the adjacency matrix of G, i.e., $A = \{e_{ij}\}, i, j = 1$ to N where $e_{ij} = 1$ if nodes v_i and v_j are adjacent and 0 otherwise.

Let each node of graph G at time t be represented by a D-dimensional feature vector represented as $\mathbf{x}_i^t \in \mathbb{R}^D$ that represents the node embeddings, which consists of graph embeddings of that node in the road network, other related exogenous variables related to that node like weather, and the traffic volume at that node. Also, let the volume of traffic at time t at node i be c_i^t.

Define feature vectors for all nodes at a particular time t as

$$\mathbf{X}_t = (\mathbf{x}_1^t, \mathbf{x}_2^t, \ldots, \mathbf{x}_N^t) \in \mathbb{R}^{N \times D} \tag{1}$$

Let T denoate the number of consecutive time steps under consideration. Then we denote the values of all feature vectors over all nodes over T consecutive time intervals starting at some time t_0 as

$$\boldsymbol{\chi} = (\mathbf{X}_{t_0}, \mathbf{X}_{t_0+1}, \ldots, \mathbf{X}_{t_0+T-1}) \in \mathbb{R}^{T \times N \times D} \tag{2}$$

Given the aforementioned notation, this paper proposes TraDWin, a Traffic Digital Twin, that is capable of addressing the following problems:

(i) **Traffic prediction:** Given a graph $G(V, E)$ and a sequence $\boldsymbol{\chi}$ of feature vectors representing observed historical traffic flow over T consecutive intervals, i.e., $\boldsymbol{\chi} = (\mathbf{X}_{t_0}, \mathbf{X}_{t_0+1}, \ldots, \mathbf{X}_{t_0+T-1})$, the objective is to predict $\mathbf{Y}$, which denotes the traffic volume counts c_i for $i \in N$ at the subsequent timestep $t_0 + T$, i.e., $\mathbf{Y} = c_1^{t_0+T}, c_2^{t_0+T}, \ldots, c_N^{t_0+T}$. This task will henceforth be referred to as task (i).

(ii) **Imputation:** Given a graph $G(V, E)$ and a sequence $\boldsymbol{\chi}$ of feature vectors representing observed historical traffic flow over T consecutive intervals, i.e., $\boldsymbol{\chi} = (\mathbf{X}_{t_0}, \mathbf{X}_{t_0+1}, \ldots, \mathbf{X}_{t_0+T-1})$, along with a binary mask vector $\mathbf{M} \in \mathbb{R}^{N \times T}$ such that $\mathbf{M} = mij$ where $m_{ij} = 1$ if the traffic volume count at detector i at time t (i.e., c_{ij}) is known and 0 otherwise to indicate it is missing. The objective here is to impute the missing c_{ij} values. This task will henceforth be referred to as task (ii).

(iii) **Traffic assignment on edge addition/removal:** Given the original graph $G(V, E)$ and the newly modified graph $G'(V', E')$ with an edge e added or removed, let ϕ be a hyperparameter denoting the number of closest neighbor nodes to the changed edge to mask out, i.e., $\mathbf{M} = m_i$ where $m_i = 1$ if dist$(v_i, e) > \phi$ and 0 otherwise. The objective is to predict $\mathbf{Y}' = c_1', c_2', \ldots, c_N'$ where c_i' represents the traffic volume of the detectors with the modified topology G' at the same timestep. This task will henceforth be referred to as task (iii).

IV. System Model

A. *Data streaming framework*

Reliable real-time data collection is a crucial aspect and prerequisite of a digital twin to effectively update internal parameters and synchronize with the real world. The Traffic Digital Twin, TraDWin, presented in this paper, requires real-time traffic volume data as input. We aim to use absolute traffic volume counts available at a manageable low frequency of 15 minutes to one hour. While systems with more sophisticated data at a higher frequency can be proposed, our goal is to maintain the collection infrastructure as affordable, straightforward, and compatible with already existing systems globally.

Our proposed system aims to ensure practicality and adaptability as follows. First, the hardware infrastructure should be inexpensive and easily replaceable. Second, the system should be easily extendable to incorporate various data sources. Third, it should function effectively using only absolute traffic counts, without requiring additional specific information such as direction and vehicle types. Finally, the system should operate with a medium-frequency update.

Various methods can be employed to collect the necessary data. The most reliable method involves installing inductive loop detectors embedded in road surfaces at city intersections. These systems are already operational in several major cities. For instance, the Sydney Coordinated Adaptive Traffic System (SCATS) [7] is utilized in over 180 cities across 28 countries, including New Zealand, Dublin, Shanghai, and Hong Kong [8]. Other examples include New York City's Adaptive Traffic Control System (ATCS), Adaptive Signal Control Technology (ASCT), SCOOT, and ACS, which are designed to control traffic signal timings and mitigate congestion. These systems can be seamlessly integrated with our proposed model, as they are capable of collecting the absolute traffic volume counts required for our analysis.

Another potential data source involves the use of surveillance cameras combined with computer vision models [49], which are already implemented at certain intersections in Shenzhen, China. Several existing studies [50] employing state-of-the-art object detection models, such as YOLO [51], have demonstrated the effectiveness of this method. This model-based approach enables cost-effective and efficient traffic monitoring utilizing the existing surveillance camera infrastructure.

Other methods, such as using GPS-enabled mobile phones [10] to track urban traffic flow, as implemented by Google in its Maps product, and the use of probe vehicles [9], which are

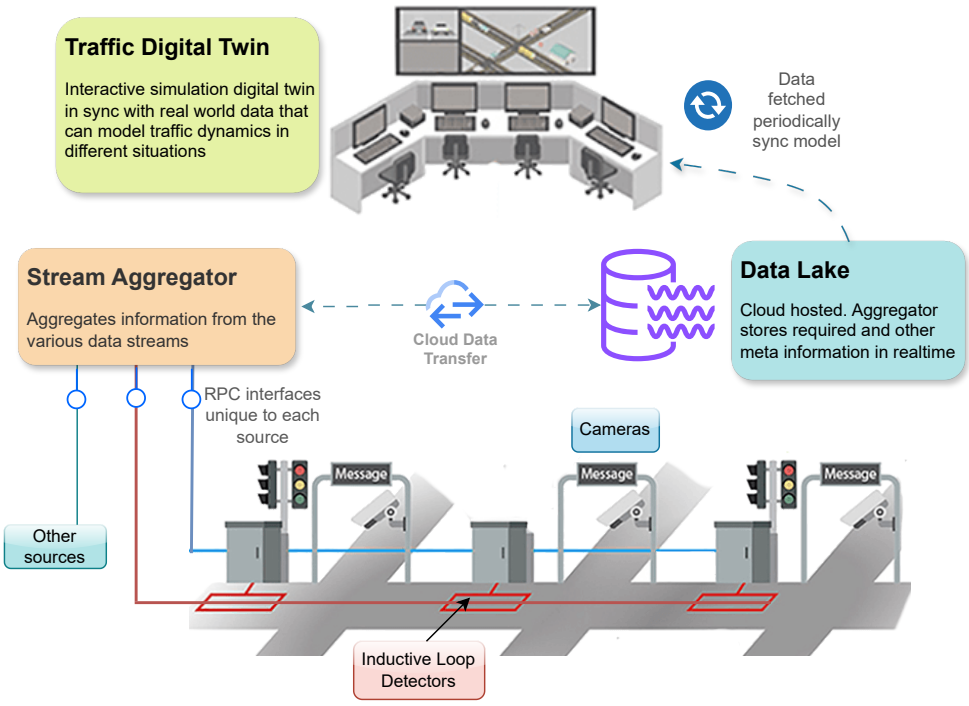


Fig. 1: TraDWin data processing framework: Multiple input data streams are aggregated and used to synchronize the model's internal state with real-world traffic state.

vehicles equipped with detectors that may be taxis or public transport, can also be utilized to approximate traffic volume based on their data.

We aim for our model to work in conjunction with multiple data streams, as different parts of the road network may be served with different data sources. We want our TraDWinto integrate with these sources seamlessly, so we need an aggregator to reconcile the data from the different streams and store them in a data lake. This data can then be used by TraDWinto keep its internal state in sync with real-world data.

One way to deploy the aggregator system is to equip it with both RPC and API interfaces for interaction with remote sensors and various data sources. Each service can integrate a separate interface specifically designed to interface with the aggregator. Subsequently, the aggregator consolidates all incoming data, along with relevant meta information, into a datalake—a centralized repository capable of handling both structured and unstructured data. This approach leverages the datalake's capacity to store additional information beyond traffic volume counts from diverse sources, paving the way for potential future enhancements and expansions of TraDWin's capabilities.

A simplified representation of the aforementioned process is depicted in Fig. 1.

B. *Syncing with realtime data and re-learning*

As a Digital Twin, there's two problems we must address, which are as following:

1) Syncing Internal State with Realtime Data

The Digital Twin synchronization mechanism incorporates a robust state update strategy that dynamically integrates real-time measurements while maintaining system state continuity.

Let $\mathbf{S}_t = [s_{1,t}, s_{2,t}, \ldots, s_{n,t}]$ represent the system state vector at time t, where each $s_{i,t}$ corresponds to a specific traffic parameter such as volume, speed, or occupancy for different road segments.

The state update is governed by the equation:

$$\mathbf{S}_{t+\Delta t} = \mathbf{D}_{\text{real}} \odot \mathbf{M} + \mathbf{S}_t \odot (1 - \mathbf{M}) \odot \mathbf{\Theta} \tag{3}$$

Mask Definitions:

The mask $\mathbf{M}$ is defined as a vector of binary indicators for data presence:

$$\mathbf{M} = \begin{cases} 1 & \text{if } \mathbf{D}_{\text{real}} \text{ is available} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The threshold matrix $\mathbf{\Theta}$ is defined based on the temporal difference between current and previous measurements:

$$\mathbf{\Theta} = \begin{cases} 1 & \text{if } |t_{\text{current}} - t_{\text{previous}}| \leq T_{\text{threshold}} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Where:

- t_{current} is the timestep of the current measurement
- t_{previous} is the timestep of the previous measurement in state
- $T_{\text{threshold}}$ is a predefined timestep threshold

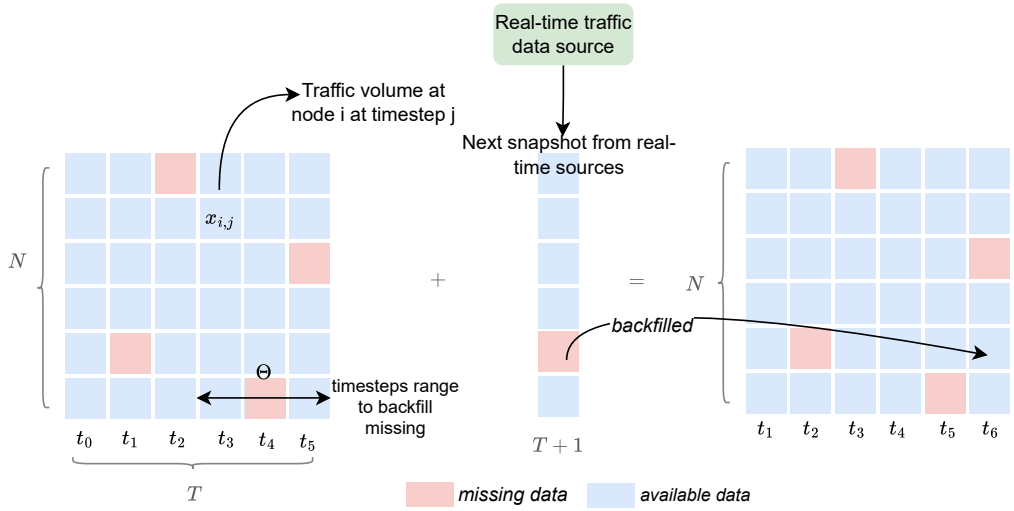


Fig. 2: TraDWin Merging real-time traffic data from sensors with existing data to get a window of T timesteps

The key idea being selecting latest data and falling back to older data if it's not available, while still maintaining that we do not use data that is too stale. This approach enables adaptive synchronization, handling sparse and heterogeneous data streams while preserving the Digital Twin's state integrity by selectively incorporating measurements based on both data availability and temporal consistency. An illustrative example can be seen in Fig. 2

2) Re-learning based on updated traffic dynamics

To maintain the accuracy and reliability of the Digital Twin, continuous learning from updated traffic dynamics is essential. Traffic patterns evolve due to various factors such as infrastructure changes, policy updates, and seasonal variations. Therefore, it is necessary to adapt the system periodically through a re-learning process.

Given the inherently noisy nature of real-time traffic data, performing continuous re-training at short intervals may introduce significant volatility and reduce predictive stability. To mitigate this, we propose a periodic re-learning strategy, executed at fixed intervals, such as on a weekly basis. This approach balances responsiveness to new traffic conditions while minimizing noise-induced instability.

A sliding window mechanism can be used for re-learning, where the training dataset continuously evolves by discarding the oldest data and appending the most recent observations. This method ensures that the model remains updated with recent traffic patterns while retaining historical context for better generalization.

Importantly, the re-learning process does not involve training the model from scratch. Instead, the model undergoes fine-tuning using newly available data, leveraging pre-existing knowledge to accelerate adaptation while preserving stability.

Let $\mathcal{D}t$ represent the dataset at time t, defined as:

$$\mathcal{D}_t = d_{t-w+1}, d_{t-w+2}, \ldots, d_t \tag{6}$$

where w denotes the window size. The window size w can be tuned based on system memory and expected traffic variability.

To enhance the model's sensitivity to recent traffic dynamics, techniques such as exponential moving average (EMA) weighting can be considered, giving higher importance to more recent data points in the loss function, while still leveraging historical data for stability.

C. Model input

For each node, a composite feature vector is constructed to encapsulate all pertinent node-related data. This vector is formed by concatenating three distinct feature vectors, each defined as follows:

1) Graph embedding generation using Node2Vec:

To facilitate the model's understanding of spatial relationships among detector nodes within the graph structure, an effective representation of the nodes' neighborhood structure is essential. Node2Vec [16] addresses this need through a feature learning approach that employs second-order biased random walks within a Skip-gram architecture. The process of generating feature vectors involves two pivotal steps:

(i) *Sampling using second-order biased random walks*: Two hyper-parameters, p and q, are introduced in the context of Node2Vec, where p controls the likelihood of revisiting nodes in the random walk (termed as the "return" parameter), and q adjusts the likelihood of exploring nodes further away from the current node (termed as the "in-out" parameter). Let l denote the fixed length of the simulated random walk starting at node u, with c_i representing the ith node in the walk ($c_0 = u$). Nodes c_i are generated according to the following distribution:

$$P(c_i = x \mid c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & \text{if } (v,x) \in E \\ 0 & \text{otherwise} \end{cases}$$

where π_{vx} is the unnormalized transition probability between nodes v and x, and Z is the normalizing constant.

For a random walk that has just traversed edge (t, v) and is currently at node v, the next step involves evaluating transition probabilities π_{vx} for edges (v, x) leading from v. Here, the transition probability is determined by $\pi_{vx} = \alpha_{pq}(t, x) \cdot w_{vx}$, where $\alpha_{pq}(t, x)$ denotes the transition probability adjustment factor defined as:

$$\alpha_{pq}(t,x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases}$$

and d_{tx} denotes the shortest path distance between nodes t and x.

(ii) *Optimizing objective function using Skip-gram*: Let $G(V, E)$ be a graph where $f : V \to \mathbb{R}^d$ denotes the function mapping each node u to a d-dimensional feature representation. Hence, f forms a matrix of parameters sized $|V| \times d$. For each node $u \in V$, let $N_S(u) \subset V$ represent the *network neighborhood* generated in step (i). The similarity between nodes u and v is defined as:

$$P_f(v|u) = \frac{\exp(f(v)^T f(u))}{\sum_{w \in V} \exp(f(w)^T f(u))} \tag{1}$$

For neighborhood $N_s(v)$ of v, define the probability of the neighborhood of u as:

$$P_f(N_s(u)|u) = \prod_{v \in N_s(u)} P_f(v|u) \tag{2}$$

Further, the global neighborhood likelihood for a given f can be defined as:

$$\sum_{u \in V} \log P_f(N_s(u) \mid u) \tag{3}$$

which is the objective function that we want to maximize through the feature representation function f. Hence,

$$\max_{u \in V} \sum_{u \in V} \log \left(\prod_{v \in N_s(u)} \frac{\exp(f(v)^T f(u))}{\sum_{w \in V} \exp(f(w)^T f(u))} \right) \tag{4}$$

2) Normalizing traffic volume

The input data should be normalized to increase the training speed and effectiveness. The traffic volume counts at different detectors are normalized as follows:

$$x_{\text{norm}} = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}$$

Thus, x_{norm} is in the range $[0, 1]$ after normalization. This same approach is used for other continuous or discrete single-valued features.

3) Word2Vec encoding of other exogenous variables:

The Skip-gram architecture, a prominent model in natural language processing [52], operates under the Word2Vec framework [17]. Its primary objective is to predict surrounding context words based on a given target word. By doing so, Skip-gram effectively learns to encode semantic meanings of words by maximizing the conditional probability of context words given the target word. This objective function is mathematically formulated as:

$$J_\theta = \frac{1}{T} \sum_{t=1}^{T} \sum_{-n \le j \le n, j \neq 0} \log p\left(w_{j+1} \mid w_t\right)$$

where w_t represents the target word at position t, c is the size of the context window, and T is the total number of words in the corpus.

In our methodology, we employ *Word2Vec* [17] to encode various exogenous variables such as weather conditions, road types, and other potentially relevant factors into feature vectors for the model input. This enhancement aims to augment the model's effectiveness by capturing semantic relationships between these variables and the observed traffic patterns. Importantly, our approach is designed to accommodate future extensions, allowing for the inclusion of additional relevant features as they are identified and integrated into the model input.

Concatenating the feature vectors obtained from Sections IV-C1 to IV-C3, we derive the final input vector for our model. Extending upon the notation introduced in Section III, for N nodes, we define $\mathbf{x}_i^t$, the feature vector of node i at time t. Assuming the use of *Node2Vec* as described in (i) to generate R^{D_g}-dimensional graph embeddings g_i for each node i, and normalizing n single-dimensional real-valued features $(r_1, r_2, \ldots, r_n)$ as outlined in (ii) to $(r_1^{\text{norm}}, r_2^{\text{norm}}, \ldots, r_n^{\text{norm}})$, resulting in a $D_n = n$ dimensional vector h_i^t. Additionally, leveraging *Word2Vec* to produce d_w dimensional vectors for m semantic "words" related to each node, we obtain a $R^{m \times d_w}$-dimensional vector flattened and represented as $k_i^t \in R^{D_w}$, where $D_w = m \times d_w$.

Finally, by concatenating the feature vectors described in Section IV-C1 - IV-C3, we obtain the feature vector of node i at time t:

$$\mathbf{x}_i^t = (g_i || h_i^t || k_i^t) \in \mathbb{R}^D$$

where $D = D_g + D_n + D_w$. Thus, leveraging the methodologies outlined in Eq (1) and (2), across T consecutive time steps and for N nodes, we construct the final feature vector $\chi \in \mathbb{R}^{T \times N \times D}$. Depending on the specific task, additional inputs such as a binary matrix M representing missing and known values are included.

D. Physics Informed Deep Learning

Physics-informed deep learning (PIDL) represents a distinctive approach within deep learning (DL), where neural networks are trained to adhere to fundamental physical laws while performing learning tasks. By integrating these laws as prior knowledge, PIDL ensures that the resulting neural network functions as a data-efficient approximator, capable of accurately predicting outcomes by respecting physical principles.

In scenarios demanding adherence to physical laws, such as ensuring consistency with conservation laws, PIDL proves invaluable. This approach guides the model to produce more precise predictions by eliminating physically implausible outcomes, thereby enhancing the reliability of predictions, particularly in scenarios like GANs.

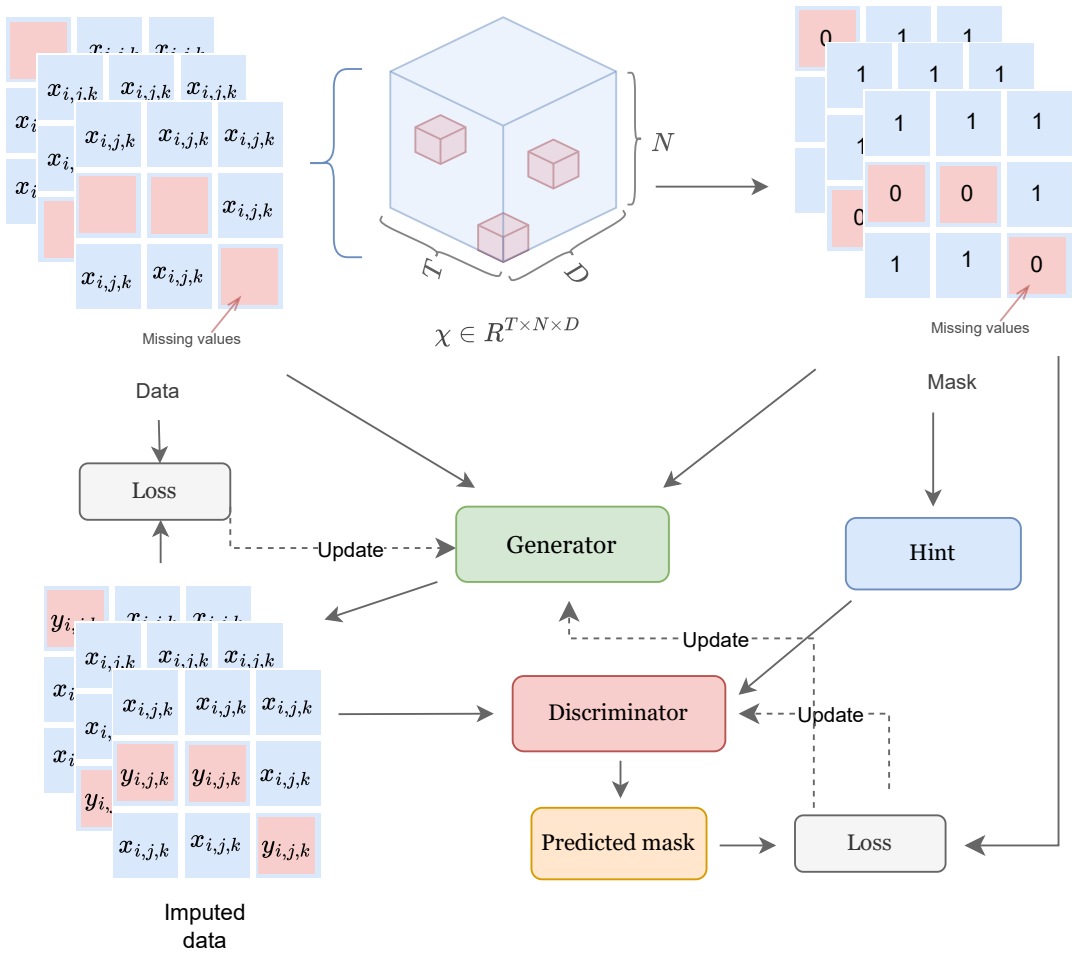


Fig. 3: TraDWin architecture: Generator fills in the missing values in the input data, while the discriminator distinguishes between real and generated data.

Specifically, for task(iii) where the goal is to predict traffic volume changes after edge addition/removal in the same time step, adherence to the *conservation law* of traffic is crucial. his principle dictates that the total volume of traffic remains consistent in a sufficiently large region before and after any redistribution.

Formally, at a specific fixed time, consider $G(V, E)$ as the original graph and $G'(V', E')$ as the graph obtained by adding or removing an edge e. Let c_i denote the traffic volume at detector i in G for $i = 1$ to $N = |V|$, and c'_i denote the traffic volume at detector i in G' for $i = 1$ to $N' = |V'|$. According to the *conservation law*:

$$C = \sum_{i=1}^{N} c_i = C' = \sum_{i=1}^{N'} c'_i \tag{7}$$

This principle ensures that the total traffic volume across the network remains constant despite the addition or removal of edges.

To enforce conservation principles as defined in Equation (7), it is essential to incorporate them into the learning framework of the Physics-Informed Deep Learning (PIDL) network. A straightforward approach involves integrating these principles as an additional term in the loss function used during model updates. This method effectively penalizes the model for predictions that violate physical constraints. Using traffic volumes c_i and other graph-related information as input for training, we establish two distinct metrics for evaluating the generator output by the discriminator:

1) $\mathcal{L}_{DL}$: Denoting the conventional discriminator loss defined as:

$$\mathcal{L}_{DL} = \mathbb{E}_{x \sim P_{\text{data}}}[D(x)] - \mathbb{E}_{z \sim P_z}[D(G(z))]$$

2) $\mathcal{L}_{PHY}$: The conservation loss, denoted as defined in Equation (3), incorporates a mask M where S represents the set of detectors i such that $M(i) = 0$, expressed as $S = \{i \mid M(i) = 0\}$. Here, C_0 denotes the sum of traffic volume counts masked by M, calculated as $C_0 = \sum_{i \in S} c_i$. The generator output $Y = \{c'_i \mid i \in S\}$ is defined accordingly. The conservation loss $\mathcal{L}_{PHY}$ is formulated as:

$$\mathcal{L}_{PHY} = (C_0 - \sum_{i \in S} c'_i)^2$$

To regulate the influence of various components within the loss function, we introduce two additional hyperparameters, μ and λ, to modulate the weights of $\mathcal{L}_{DL}$ and $\mathcal{L}_{PH}$ respectively. Consequently, the final loss function is defined as:

$$\mathcal{L} = \mathcal{L}_{DL} + \mu \cdot \mathcal{L}_{PH}$$

where $\mathcal{L}_{DL}$ represents the primary loss for deep learning tasks and $\mathcal{L}_{PH}$ denotes the conservation loss. Adjusting μ allows us to prioritize between maintaining physical consistency and optimizing for predictive accuracy within the model.

In summary, adjusting hyperparameters allows us to fine-tune how our model trade-offs between objectives. By ensuring the discriminator respects physical laws, the generator becomes better at identifying underlying data patterns. This approach proved effective in our experiments, as we detailed subsequently.

E. ***TraDWin: Traffic Model***

This section outlines the architecture of our *TraDWin* model, drawing inspiration from the Generative Adversarial Imputation Nets (GAIN) [18] and Wasserstein Generative Adversarial Network (WGAN) [19]. GAIN, a novel conditional GAN variant, has demonstrated effectiveness in various tasks involving imputation of missing values across diverse datasets. On the other hand, WGAN represents an advancement over traditional GANs, emphasizing training stability through the use of Wasserstein distance rather than Jensen-Shannon divergence. Architectural enhancements such as gradient clipping and elimination of the sigmoid function contribute to improved convergence and generation of higher-quality samples. Fig. 3 depicts the architecture of TraDWin.

1) Generator:

The generator G of our proposed model accepts inputs consisting of observed data with missing values, denoted as $\mathbf{X}$, a binary mask vector $\mathbf{M}$ that identifies the positions of missing values, and a noise vector $\mathbf{Z}$. It generates imputed data $\mathbf{X}_g$, which initially represents a vector of imputations, subsequently projected to form the complete imputed output $\mathbf{Y}_0$.

Formally, the output of the generator G can be given as:

$$\mathbf{X}_g = G(\mathbf{X}, \mathbf{M}, (\mathbf{1} - \mathbf{M}) \odot \mathbf{Z})$$

Here, $\odot$ denotes the Hadamard product, or element-wise multiplication, and $\mathbf{Z}$ represents the d-dimensional noise vector. The generator G produces $\mathbf{X}_g$, which fills in the missing values indicated by $\mathbf{M}$ within the observed data $\mathbf{X}$. The resultant output is then evaluated by the discriminator to assess its quality.

2) Discriminator:

Similar to the traditional GAN framework, our model incorporates a Discriminator D alongside the Generator G. The Discriminator D is designed to evaluate the imputations generated by G. In the context of our adaptation inspired by GAIN, D maps the input χ to a binary vector in $[0, 1]^d$, where each component signifies whether the corresponding element in the generator's output is real (observed) or fake (imputed).

In contrast to traditional Generative Adversarial Networks (GANs), where the discriminator distinguishes between entirely real and fake outputs, the discriminator in Generative Adversarial Imputation Nets (GAIN) serves to classify individual components as real or imputed. This nuanced approach allows for a more granular evaluation of the generator's imputation performance. The loss function employed is a stochastic gradient descent (SGD) that averages the losses across these individual components, aiming to enhance the fidelity of imputed data.

3) Hint

GAIN introduces an innovative concept known as the hint mechanism, characterized by a random variable $\mathbf{H}$ that ranges over a space $\mathcal{H}$. The primary objective of this mechanism is to furnish supplementary missing information to the discriminator concerning the binary mask. This concept was formally introduced and analyzed in [18], where the following proposition was articulated and proved:

> *Proposition 1: There exist distributions of* $\mathbf{X}$*,* $\mathbf{M}$*, and* $\mathbf{H}$ *for which solutions to* $\hat{p}(\mathbf{x}|\mathbf{h}, m_i = t) = \hat{p}(\mathbf{x}|\mathbf{h})$ *for each* $i \in \{1, \ldots, d\}$*,* $\mathbf{x} \in \mathcal{X}$*, and* $\mathbf{h} \in \mathcal{H}$ *such that* $p_h(\mathbf{h}|m_i = t) > 0$ *are not unique under the optimality criterion of GAN.*

This implies the existence of multiple potential distributions that the generator G might produce, appearing plausible to the discriminator D. Hence, the random variable $\mathbf{H}$ assumes the crucial role of furnishing adequate information to distinctly identify the accurate representation of the underlying data.

The hint mechanism relies on the binary mask vector $\mathbf{M}$. For each imputed sample $(\hat{\mathbf{x}}, \mathbf{m})$, a random variable $\mathbf{h}$ is drawn from the conditional distribution $\mathbf{H}|\mathbf{M} = \mathbf{m}$. $\mathbf{h}$ is then included as an additional input to the discriminator, altering its function to $D : \mathcal{X} \times \mathcal{H} \to [0, 1]^d$. Each component of $D(\hat{\mathbf{x}}, \mathbf{h})$ now indicates the probability that the corresponding component of $\hat{\mathbf{x}}$ was observed given $\hat{\mathbf{X}} = \hat{\mathbf{x}}$ and $\mathbf{H} = \mathbf{h}$. The definition of $\mathbf{H}$ governs the level of information it provides about $\mathbf{M}$.

Traditional GANs encounter challenges such as training instability and the generation of repetitive outputs. WGAN mitigates these issues by employing the Wasserstein distance metric in place of the Jensen-Shannon divergence. This approach also incorporates architectural improvements such as gradient clipping and the elimination of the sigmoid function. Now, we discuss how these enhancements from WGAN have been integrated into GAIN. In traditional GAN, the generator and discriminator are trained using the following loss functions:

$$\mathcal{L}^G_{\text{GAN}} = \log(1 - D(G(z)))$$

$$\mathcal{L}^D_{\text{GAN}} = -\log(D(x)) - \log(1 - D(G(z)))$$

In contrast, in WGANs, we remove the logarithms and use Wasserstein distance for loss instead.

$$\mathcal{L}^G_{\text{WGAN}} = -D(G(z))$$

$$\mathcal{L}^D_{\text{WGAN}} = D(G(z)) - D(x)$$

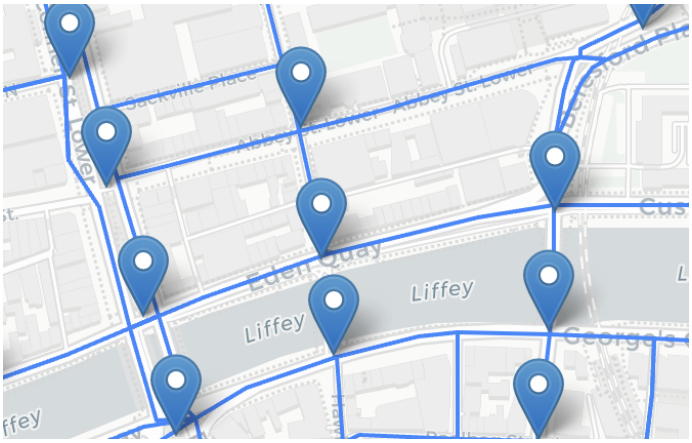


Fig. 4: Snapshot of sensor locations plotted along with road network from Overpass API

Another adaptation from WGANs that we integrate into GAIN is the omission of a sigmoid activation function in the discriminator's output layer, unlike in traditional GANs. This modification allows the discriminator to produce unbounded real-valued scores rather than probabilities. Additionally, WGANs utilize gradient clipping as a regularization method to improve training stability. Gradient clipping involves setting a threshold for gradients; if their magnitude exceeds this threshold, they are scaled down to prevent instability, such as exploding gradients, during training.

V. Experiments

In this section, we present the experimental results of our model, detailing the parameters necessary to replicate these outcomes. Additionally, we provide a description of the baseline models used for comparison, along with their respective parameters.

A. Dataset Description

We utilize real-world traffic volume data from Dublin city streets, collected using the SCATS (Sydney Coordinated Adaptive Traffic System) over a three-month period. SCATS is employed in Dublin as an adaptive urban traffic management system that synchronizes traffic signals and optimizes traffic flow across the entire city.

The dataset comprises traffic volumes from 825 sensors located at key intersections in Dublin city, sampled hourly from October 1, 2023, to December 31, 2023. This dataset includes the latitudinal and longitudinal geographical coordinates along with the timestamped total traffic volume detected by each sensor within each one-hour period. A visual representation is in Fig. 4

Weather data, used in conjunction with the traffic data, is sourced from the official website of Met Éireann, the state meteorological service of Ireland. This data includes hourly information on weather phenomena such as air temperature, relative humidity, visibility, and precipitation amount, collected from three weather stations within the city of Dublin.

We also verify the model for task (iii) using the SUMO [11] TAPASCologne scenario [53], which simulates the traffic dynamics of Cologne, Germany, over a day. This scenario is publicly available on the official SUMO website and was created using mobility demand data based on citizens' travel habits and information about the local infrastructure. Due to the manual nature of this testing, we conduct a limited number of tests where we alter an edge, process the simulation based on the default minimal travel time routing before and after the change, and observe the deviation between the simulation and our results.

B. Model Parameters

For all experiments, we set hyperparameters $T = 3$, representing the number of consecutive timesteps to consider for prediction and imputation tasks. For all tasks, we optimize the model input by assuming the locality of traffic within such a timespan and that any changes to a particular node are only dependent on nodes "close" to it. During training, we train on subgraphs consisting of node clusters instead of the entire graph at once. Specifically, a subgraph is generated by choosing a node u at random and then selecting all nodes that are at a distance less than σ to it. Formally, for $G(V, E)$, the subgraph is defined as $G_{\text{sub}} = G\{v \in V \mid \text{dist}(u, v) < \sigma\}$. For our experiments, we set σ to 2.5 km, which is reasonable as clusters usually contain 20 to 30 nodes within this range.

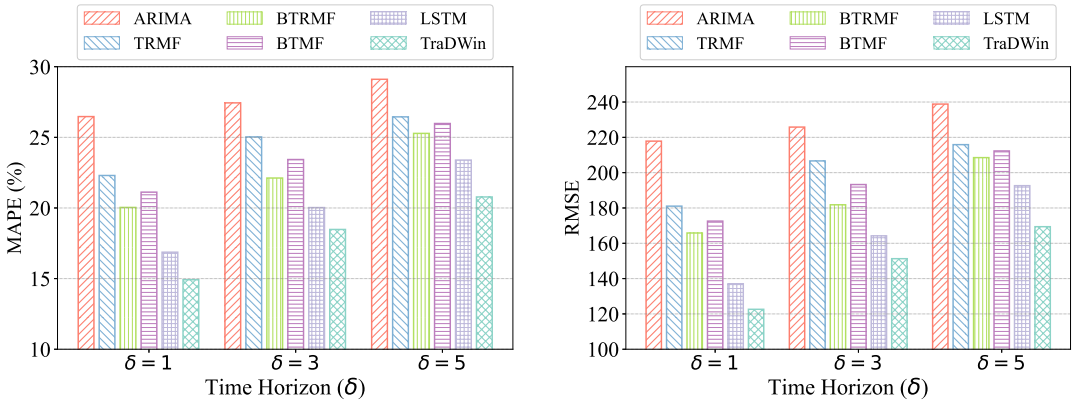
After sampling, as stated, we use an 80-20 test-train split, meaning the model is trained on 80% of the samples, and performance is verified on new unseen data comprising 20% of the samples. The data is then modified based on the task at hand. For imputation, we followed the MCAR (Missing Completely at Random) distribution and randomly masked values both spatially and temporally, based on a parameter defined as the miss rate. We tested our model on miss rates of 10%, 20%, 30%, and 40%. For prediction, we use the full data of three timesteps to predict the fourth step by masking the values along the last timestep dimension. For the reassignment task, i.e., task (iii), we only consider one timestep and re-assign values for that same timestep but with modified spatial geometry.

C. Baselines

We compare our TraDWin model with several existing approaches across different domains, including various mathematical analysis and deep learning techniques that are commonly used for these tasks.

For imputation, one of the simplest approaches is *KNN*. For prediction, *ARIMA* (AutoRegressive Integrated Moving Average) [25] is employed. Matrix factorization methods such as *TRMF* (Temporal Regularized Matrix Factorization) [31] use latent factors for predictions, with *BTRMF* (Bayesian Temporal Regularized Matrix Factorization) extending TRMF within a Bayesian framework. For both TRMF and BTRMF, we use a rank of 10, 1000 burn-in iterations, and 200 Gibbs iterations.

LRTC-TNN (Low-Rank Tensor Completion with Truncated Nuclear Norm) [54] is a method for tensor completion, using



(a) MAPE on prediction task for different deltas

(b) RMSE on prediction task for different deltas

Fig. 5: Performance metrics on prediction task for different deltas (timesteps to predict ahead)

parameters $\rho = 1e-5$, $\theta = 0.25$, and $\epsilon = 1e-4$. *BGCP* (Bayesian Gaussian Process Factorization) utilizes Bayesian inference. For BGCP, we use similar burn-in and Gibbs iterations and set the rank to 30. It is important to note that these model baselines are applicable only to task (i) and task (ii), and not to task (iii) since they were not designed for that task.

Additionally, we employed deep learning methods. For imputation, we used the *Denoising AutoEncoder* (DAE) [55] model, which is effective in learning meaningful representations of the data while handling missing values. For prediction tasks, we utilized an *LSTM* (Long Short-Term Memory) [56] model, as LSTM networks are well-suited for capturing temporal dependencies in sequential data.

D. Evaluation metrics

To evaluate the performance of the different methods and compare them, we use RMSE (Root Mean Squared Error) and MAPE (Mean Absolute Percentage Error). These metrics are defined as follows:

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

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

where y_i represents the true value, $\hat{y}_i$ represents the predicted value, and n is the number of samples.

E. Results

We evaluate the performance of our model on the following tasks, which cover various scenarios related to traffic modelling, along with comparing to other popular models in the domain. Note that for each of the three tasks, the model is trained independently reusing the same architecture and training setup.

1) Task (i): Traffic Prediction

For the prediction task, we test the models for predicting traffic volumes at the next timestep ($\delta = 1$), at the third timestep ($\delta = 3$), and at the fifth timestep in the future ($\delta = 5$). This scenario requires a mask consisting of all the T_{n+1} values missing, i.e., 0, for the next timestep prediction. We similarly prepare a mask for other time horizons, treating prediction as a strict MNAR (Missing Not At Random) subcategory of imputation.

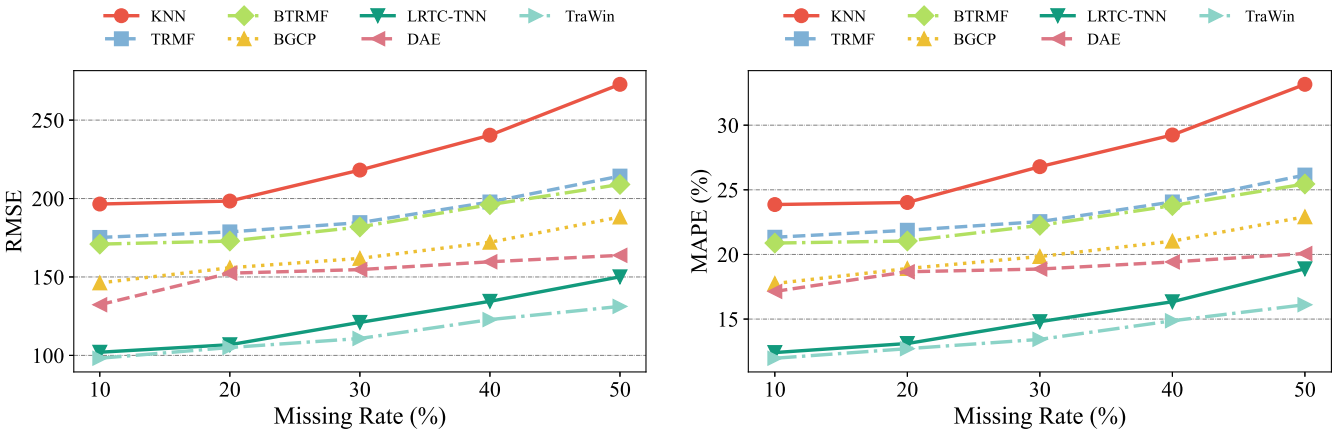
The MAPE and RMSE plots of prediction tasks across different horizons are shown in Fig. 5a and Fig. 5b, respectively. We observe that generally, as the time horizon δ increases, both metrics worsen, with MAPE falling 5-10% from $\delta = 1$ to $\delta = 3$. This indicates a stronger short-term accuracy but some challenges with long-term prediction across all models. This trend is common in time-series forecasting, where predictive accuracy decreases as the prediction horizon extends. The primary reasons for this decline include the increasing uncertainty and the influence of unpredictable external factors, such as accidents or weather changes.

ARIMA performs the worst, which is expected as it is the simplest of the mathematical models among the baselines we consider. Other matrix factorization and Bayesian models like TRMF, BTRMF, and BTMF perform better but, being strictly mathematical models, they (i) fail to capture the intricate traffic dynamics that a deep learning model can and (ii) use only the time-series traffic data and not the graph topology and external factors, such as weather, that our model considers. LSTMs perform close but slightly worse (by 2-3%), likely due to the lack of knowledge of graph topology that our model incorporates through Node2Vec.

Overall, our TraDWin model performs significantly better than most models, scoring roughly 3 to 7 percentage points better than the baselines.

2) Task (ii): Imputation

For the imputation task, we consider the MCAR (Missing Completely At Random) distribution and compare the models at five missing rates of 10%, 20%, 30%, 40%, and 50%. The performance of different models, measured using MAPE and



(a) RMSE of different models on imputation task

(b) MAPE of different models on imputation task

Fig. 6: Imputation comparison for different missing rates

RMSE, is shown in Fig. 6. Conventional methods like KNN perform poorly compared to more specialized approaches, as expected, since KNN is insufficient to capture intricate traffic dynamics. Matrix factorization and Bayesian methods, such as TRMF, BTRMF, and BGCP, work slightly better, with deep learning methods like DAE performing ahead. However, since they deal only with traffic data time series without topology information, their performance is 8-13% worse than our model. Tensor completion-based models like LRTC-TNN are close to our TraDWin for low missing rates but do not scale as well with increasing missing rates. This could be because too much data is lost for reliable tensor completion, which a deep learning model can handle due to its ability to learn complex data relationships and incorporate graph topology and other external factors. It is also worth noting that LRTC-TNN, an MCAR imputer, is not applicable to task (iii) and is not generalized enough for our use case. For all models, the accuracy of imputations decreases with higher missing rates due to the reduced amount of contextual information available, a common challenge in data imputation. Overall, our TraDWin scales well with more missing data and performs significantly (5-7%) better than the compared baselines.

We observe that TraDWin performs better on task (ii) than on task (i) by around 4%. One primary reason for this could be that GAIN, originally designed for MCAR imputation, does not generalize well to MNAR imputation scenarios. Another possible reason could be the subgraph-based computation method we use, where nodes at the corners of the subgraph lack adequate spatial and temporal information about their neighbors. Overall, our model performs reasonably and delivers comparable results as a generalized model on both tasks.

3) Task (iii): Re-assignment on Edge Addition or Removal

For the re-assignment task, which is a new task addressed in our paper and not commonly seen in contemporary literature, we train the model for MNAR imputation scenarios with the central node cluster missing. This training effectively teaches the model to assign traffic to node clusters based on neighbor node information. To elaborate, for a modified edge, nodes around the edge, based on a distance threshold, are masked (i.e., marked missing). The training input includes features for the neighbors of the cluster (nodes not masked out), graph embeddings for the masked nodes (not the traffic volume as that is marked missing), and the total volume of traffic for the masked-out nodes. The task is to assign traffic volumes to the masked-out nodes and compare them against the original values. The model learns ”traffic assignment” since real-world labeled data for traffic state before and after modification of edges is unavailable. Based on this training task, we achieve a MAPE of 13.47% and RMSE of 109.90. The other baselines considered for prediction and imputation are not applicable without changes to their model architecture, so there is no comparison with contemporary models.

The de-facto way to solve this problem, as observed in our literature review, has been through simulations like SUMO [11] and Vissim [12]. Using SUMO, we also compare and test our model on the TAPASCologne scenario. We train using the same approach as for the Dublin SCATS dataset but test against the simulation results after modifying the said edge. This is not possible for the Dublin dataset since detailed origin-destination pair data is unavailable. The results of our model on this task are shown in Table I. We observe that the model achieves a MAPE of 13.47% on the Dublin SCATS dataset and 15.06% on the TAPASCologne scenario, demonstrating the model's reasonable accuracy in solving the traffic assignment problem on node clusters. Furthermore, we evaluate how our model's performance scales with the number of modifications, i.e., how much we can alter the original graph while keeping the model viable for re-assignment. Fig. 7 shows a comparison of MAPE and RMSE with the number of edges modified. Performance declines steadily from 13.47% at one change to 31.91% at five changes, indicating that more graph alterations lead to a significant performance drop, especially after three modifications. A possible reason for this could be the mass masking strategy used to train the model for task (iii), where

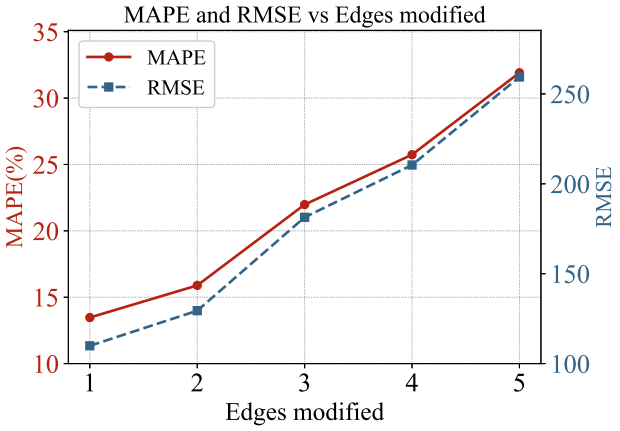


Fig. 7: MAPE and RMSE on re-assignment task vs number of edges modified

TABLE I: Performance on re-assignment task (1 edge change) for different datasets

Dataset	MAPE (%)	RMSE
Dublin SCATS	13.47	109.90
TAPASCologne	15.06	23.34

masking too many neighboring nodes leads to a large loss of contextual information that graph representation knowledge cannot fully compensate.

VI. Conclusion

In conclusion, we propose an end-to-end framework for an interactive traffic digital twin, including a model. We evaluate and compare our model's performance on the real-world Dublin SCATS dataset, and our experiments show that the model produces reasonable results compared to other contemporary approaches and techniques in different scenarios. We believe that such a system will be increasingly useful for managing and guiding the decision-making processes related to traffic infrastructure and planning, both by governments and private entities.

It is worth noting that the core architecture of TraDWinis the same for all three underlying tasks. In our approach, we treat each task as a separate problem for the model to be trained upon. Given that multi-task learning for models, where the model can better generalize by learning several different tasks, with output controlled by the parameters of the input itself, has shown promising results, we believe it will be worthwhile to investigate the application of that approach to our model.

References

[1] J. I. Levy, J. J. Buonocore, and K. von Stackelberg, "Evaluation of the public health impacts of traffic congestion: a health risk assessment," *Environmental health*, vol. 9, p. 65, 2010.

[2] R. Gorea, "Financial impact of road traffic accidents on the society," *IJETV*, vol. 2, no. 01, pp. 6–9, Jun. 2016.

[3] P. R. Anciaes, P. J. Metcalfe, and C. Heywood, "Social impacts of road traffic: perceptions and priorities of local residents," *Impact Assessment and Project Appraisal*, vol. 35, no. 2, pp. 172–183, 2017.

[4] Y. Guo, X. Hu, B. Hu, J. Cheng, M. Zhou, and R. Y. Kwok, "Mobile cyber physical systems: Current challenges and future networking applications," *IEEE Access*, vol. 6, pp. 12 360–12 368, 2017.

[5] M. Singh, E. Fuenmayor, E. P. Hinchy, Y. Qiao, N. Murray, and D. Devine, "Digital twin: Origin to future," *Applied System Innovation*, vol. 4, no. 2, p. 36, 2021. [Online]. Available: https://doi.org/10.3390/asi4020036

[6] E. VanDerHorn and S. Mahadevan, "Digital twin: Generalization, characterization and implementation," *Decision Support Systems*, vol. 145, p. 113524, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167923621000348

[7] A. Sims and K. Dobinson, "The sydney coordinated adaptive traffic (scat) system philosophy and benefits," *IEEE Transactions on Vehicular Technology*, vol. 29, no. 2, pp. 130–137, 1980.

[8] Wikipedia contributors, "Sydney coordinated adaptive traffic system — Wikipedia, the free encyclopedia," 2022, online; accessed 25-April-2024. [Online]. Available: https://en.wikipedia.org/wiki/Sydney_Coordinated_Adaptive_Traffic_System

[9] Y. Zhu, Z. Li, H. Zhu, M. Li, and Q. Zhang, "A compressive sensing approach to urban traffic estimation with probe vehicles," *IEEE Transactions on Mobile Computing*, vol. 12, no. 11, pp. 2289–2302, 2012.

[10] G. Rose, "Mobile phones as traffic probes: practices, prospects and issues," *Transport Reviews*, vol. 26, no. 3, pp. 275–291, 2006.

[11] D. Krajzewicz, "Traffic simulation with sumo–simulation of urban mobility," *Fundamentals of traffic simulation*, pp. 269–293, 2010.

[12] M. Fellendorf and P. Vortisch, "Microscopic traffic flow simulator vissim," *Fundamentals of traffic simulation*, pp. 63–93, 2010.

[13] K. Kušić, R. Schumann, and E. Ivanjko, "A digital twin in transportation: Real-time synergy of traffic data streams and simulation for virtualizing motorway dynamics," *Advanced Engineering Informatics*, vol. 55, p. 101858, 2023.

[14] M. Cools, E. Moons, and G. Wets, "Assessing the impact of weather on traffic intensity," *Weather, Climate, and Society*, vol. 2, no. 1, pp. 60–68, 2010.

[15] ——, "Investigating the variability in daily traffic counts through use of arimax and sarimax models: assessing the effect of holidays on two site locations," *Transportation research record*, vol. 2136, no. 1, pp. 57–66, 2009.

[16] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[17] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013.

[18] J. Yoon, J. Jordon, and M. Schaar, "Gain: Missing data imputation using generative adversarial nets," in *International conference on machine learning*. PMLR, 2018, pp. 5689–5698.

[19] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," 2017.

[20] A. J. Huang and S. Agarwal, "Physics-informed deep learning for traffic state estimation: Illustrations with lwr and ctm models," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 3, pp. 503–518, 2022.

[21] H. Wang, X. Chen, F. Jia, and X. Cheng, "Digital twin-supported smart city: Status, challenges and future research directions," *Expert Systems with Applications*, vol. 217, p. 119531, 2023.

[22] M. Jafari, A. Kavousi-Fard, T. Chen, and M. Karimi, "A review on digital twin technology in smart grid, transportation system and smart city: Challenges and future," *IEEE Access*, vol. 11, pp. 17 471–17 484, 2023.

[23] A. Matei and M. Cocoșatu, "Artificial internet of things, sensor-based digital twin urban computing vision algorithms, and blockchain cloud networks in sustainable smart city administration," *Sustainability*, vol. 16, no. 16, p. 6749, 2024.

[24] M. Batty, "Digital twins in city planning," *Nature Computational Science*, vol. 4, pp. 192–199, 2024.

[25] R. Adhikari and R. K. Agrawal, "An introductory study on time series modeling and forecasting," *arXiv preprint arXiv:1302.6613*, 2013.

[26] H. Dong, L. Jia, X. Sun, C. Li, and Y. Qin, "Road traffic flow prediction with a time-oriented arima model," in *2009 Fifth International Joint Conference on INC, IMS and IDC*. IEEE, 2009, pp. 1649–1652.

[27] S. V. Kumar and L. Vanajakshi, "Short-term traffic flow prediction using seasonal arima model with limited input data," *European Transport Research Review*, vol. 7, pp. 1–9, 2015.

[28] I. Kochetkova, A. Kushchazli, S. Burtseva, and A. Gorshenin, "Short-term mobile network traffic forecasting using seasonal arima and holt-winters models," *Future Internet*, vol. 15, p. 290, 2023.

[29] C. Xu, Z. Li, and W. Wang, "Short-term traffic flow prediction using a methodology based on autoregressive integrated moving average and genetic programming," *Transport*, vol. 31, no. 3, pp. 343–358, 2016.

[30] A. R. Sattarzadeh, R. J. Kutadinata, P. N. Pathirana, and V. T. Huynh, "A novel hybrid deep learning model with arima conv-lstm networks and shuffle attention layer for short-term traffic flow prediction," *Transportmetrica A: Transport Science*, pp. 1–23, 2023.

[31] H.-F. Yu, N. Rao, and I. S. Dhillon, "Temporal regularized matrix factorization for high-dimensional time series prediction," *Advances in neural information processing systems*, vol. 29, 2016.

[32] X. Chen and L. Sun, "Bayesian temporal factorization for multidimensional time series prediction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 9, pp. 4659–4673, 2021.

[33] Z. Yang, L. T. Yang, H. Wang, B. Ren, and X. Yang, "Bayesian tensor completion for network traffic data prediction," *IEEE Network*, vol. 37, no. 4, pp. 74–80, 2023.

[34] J. Lai, Z. Chen, J. Zhu *et al.*, "Deep learning based traffic prediction method for digital twin network," *Cognitive Computation*, vol. 15, pp. 1748–1766, 2023.

[35] W. Zhang, H. Zha, L. Gan, and Q. Li, "Ds-tfsn-based vehicle travel time prediction method for digital twin system of freeways," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 12, pp. 20 073–20 084, 2024.

[36] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI open*, vol. 1, pp. 57–81, 2020.

[37] W. Jiang, J. Luo, M. He, and W. Gu, "Graph neural network for traffic forecasting: The research progress," *ISPRS International Journal of Geo-Information*, vol. 12, no. 3, p. 100, 2023.

[38] A. Sharma, A. Sharma, P. Nikashina, V. Gavrilenko, A. Tselykh, A. Bozhenyuk, M. Masud, and H. Meshref, "A graph neural network (gnn)-based approach for real-time estimation of traffic speed in sustainable smart cities," *Sustainability*, vol. 15, no. 15, p. 11893, 2023.

[39] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," 2018. [Online]. Available: https://arxiv.org/abs/1707.01926

[40] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," 2019. [Online]. Available: https://arxiv.org/abs/1906.00121

[41] S. Messinis, O. E. Marai, N. E. Protonotarios, T. Taleb, and N. Doulamis, "Roads digital twin: Predictive situational awareness using 360° video streaming and graph neural networks," *IEEE Transactions on Vehicular Technology*, pp. 1–14, 2024.

[42] U. Kaytaz, S. Ahmadian, F. Sivrikaya, and S. Albayrak, "Graph neural network for digital twin-enabled intelligent transportation system reliability," in *2023 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*, 2023, pp. 1–7.

[43] Z. Shao, Z. Zhang, W. Wei, F. Wang, Y. Xu, X. Cao, and C. S. Jensen, "Decoupled dynamic spatial-temporal graph neural network for traffic forecasting," 2022. [Online]. Available: https://arxiv.org/abs/2206.09112

[44] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations," *arXiv preprint arXiv:1711.10561*, 2017.

[45] G. Wong and S. Wong, "A multi-class traffic flow model–an extension of lwr model with heterogeneous drivers," *Transportation Research Part A: Policy and Practice*, vol. 36, no. 9, pp. 827–841, 2002.

[46] C. F. Daganzo, "The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory," *Transportation research part B: methodological*, vol. 28, no. 4, pp. 269–287, 1994.

[47] J. Zhang, S. Mao, L. Yang, W. Ma, S. Li, and Z. Gao, "Physics-informed deep learning for traffic state estimation based on the traffic flow model and computational graph method," *Information Fusion*, vol. 101, p. 101971, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253523002877

[48] A. J. Saroj, S. Roy, A. Guin, and M. Hunter, "Development of a connected corridor real-time data-driven traffic digital twin simulation model," *Journal of Transportation Engineering, Part A: Systems*, vol. 147, 2021.

[49] N. K. Jain, R. Saini, and P. Mittal, "A review on traffic monitoring system techniques," *Soft computing: Theories and applications: Proceedings of SoCTA 2017*, pp. 569–577, 2019.

[50] C. Asha and A. Narasimhadhan, "Vehicle counting for traffic management system using yolo and correlation filter," in *2018 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*. IEEE, 2018, pp. 1–6.

[51] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[52] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, Eds., vol. 26. Curran Associates, Inc., 2013.

[53] "Tapascologne - sumo documentation," https://sumo.dlr.de/docs/Data/Scenarios/TAPASCologne.html, accessed on 14th May 2024.

[54] X. Chen, J. Yang, and L. Sun, "A nonconvex low-rank tensor completion model for spatiotemporal traffic data imputation," *Transportation Research Part C: Emerging Technologies*, vol. 117, p. 102673, 2020.

[55] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, and L. Bottou, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." *Journal of machine learning research*, vol. 11, no. 12, 2010.

[56] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.