TraDWin: An interactive Digital Twin for City Traffic

Shreyansh Yadav
Department of Computer Science Engineering
IIT BHU, Varanasi
India
shreyansh.yadav.cse19@itbhu.ac.in

Dr. Vignesh Sivaraman
Assistant Professor
Department of Computer Science and Engineering
IIT BHU, Varanasi
India
vignesh.cse@iitbhu.ac.in

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

***Abstract*—ChatGPT 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, a Traffic Digital Twin (TDT) 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. The TDT integrates comprehensive datasets encompassing road network configurations, traffic volumes at distinct junctures, and contextual parameters including road characteristics and meteorological conditions. It is specifically engineered and validated 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. 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.

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 interactive simulation of city traffic, continuously updating its internal state with real-time data from various sources. 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. These simulation engines are computationally intensive and require 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 [13], holidays [14], and other contextual factors.

We argue that traffic dynamics are shaped by both intrinsic data relationships and external factors such as weather [13] and holidays [14]. Moreover, there may be latent relationships among these variables that are not immediately evident. Thus, our proposed TDT 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 that first collects and aggregates real-time traffic data. 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.

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 [15] 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 [16]. (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 [17], which is a type of conditional GAN. This adaptation incorporates elements from the Wasserstein GAN (WGAN) [18], enhancing GAN training effectiveness. Additionally, we integrate a modified loss function from physics-informed deep learning techniques [19], 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) Our proposed framework considers contextual data like geographical, meteorological, temporal data that are aggregated according to physcial conservation laws.
(iii) We mathematically capture the dynamics involved in traffic modeling using physical laws.
(iv) 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.

The rest of the paper is organized as follows. The literature review is presented in Section I. We formulate our problem in Section II. Our proposed framework is presented in Section III. We present our empirical findings and performance evaluation of our proposed framework in Section IV. Finally, Section V concludes the paper.

I. Related Work

In this section, 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) [20] and its variants, such as time-oriented ARIMA [21] and Seasonal ARIMA [22], 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. integrated ARIMA with a genetic algorithm to estimate future traffic flow on roads. 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.

Other methods, such as Temporal Regularized Matrix Factorization (TRMF) [23], Bayesian Temporal Regularized Matrix Factorization (BTRMF) [24], and Bayesian Temporal Matrix Factorization (BTMF) [24], have also been explored in the context of traffic prediction. 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, pro-

posed 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. Graph Neural Networks (GNNs) [25] have emerged as a prominent tool for spatiotemporal modeling, demonstrating effectiveness in various applications such as traffic forecasting and imputation. Studies by Li et al. (2018) and Zhang et al. (2019) have highlighted their utility in accurately predicting traffic patterns and filling in missing data in traffic volume time series. 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) [26] 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. 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) [27] model and the Cell Transmission Model (CTM) [28].

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.

II. Problem Formulation

A road network can be represented as an undirected graph $G = (V, E)$, where V denotes the set of nodes and E denotes the set of edges. The nodes V consist of N detector nodes, formally expressed as $V = \{v_1, v_2, \ldots, v_n\}$, where $N = |V|$ and v_i represents the ith detector node. The set E represent the road links connecting the detector nodes with $N_E = |E|$. The adjacency matrix of G, denoted by $A \in \mathbb{R}^{N \times N}$, is defined as $A = \{e_{ij}\}$, where $i, j = 1, \ldots, N$, and $e_{ij} = 1$ 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 generated as explained later using node embeddings obtained from Node2Vec, other meta-information encoded, and traffic volume counts. 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 the number of timesteps denoted by T be a hyperparameter denoting the number of consecutive time steps under consideration. Then 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, the problem that we are tackling in this paper can be formally define as proposing as Traffic Digital Twin (TDT) that can perform the following tasks:

(i) **Traffic prediction:** Given graph $G(V, E)$ and a sequence $\boldsymbol{\chi}$ of feature vectors of observed historical traffic flow over T consecutive intervals, i.e., $\boldsymbol{\chi} = (\mathbf{X}_{t_0}, \mathbf{X}_{t_0+1}, \ldots, \mathbf{X}_{t_0+T-1})$, predict $\mathbf{Y}$, the traffic volume counts c_i for $i \in N$ at the next timestep $t_0 + T$, i.e.,

$\mathbf{Y} = \{c_1^{t_0+T}, c_2^{t_0+T}, \ldots, c_N^{t_0+T}\}$. Henceforth, we refer to this task as task (i).

(ii) **Imputation:** Given graph $G(V, E)$ and a sequence $\boldsymbol{\chi}$ of feature vectors of observed historical traffic flow over T consecutive intervals, i.e., $\boldsymbol{\chi} = (\mathbf{X}_{t_0}, \mathbf{X}_{t_0+1}, \ldots, \mathbf{X}_{t_0+T-1})$, a binary mask vector $\mathbf{M} \in \mathbb{R}^{N \times T}$ such that $\mathbf{M} = \{m_{ij}\}$ 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 its missing, impute the missing c_{ij} values. Henceforth, we refer to this task as task (ii).

(iii) **Traffic assignment on edge addition/removal:** Given original graph $G(V, E)$ and the new 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 the dist$(v_i, e) > \phi$ and 0 otherwise. Predict $\mathbf{Y}' = \{c'_1, c'_2, \ldots, c'_N\}$ where c'_i is the traffic volume of the detectors with the modified topology G'. Henceforth, we refer to this task as task (iii).

III. System Model

1. 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 (TDT), 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, we aim to keep the real-world collection infrastructure cheap, simple, and based on already existing systems around the world. In particular, we strive to cater to the following objectives:

(i) Hardware infrastructure should be cheap and easily replaceable.
(ii) Easily extendable to use a variety of different sources.
(iii) Should work with only absolute traffic counts, no other specific information like direction and vehicle types.
(iv) Should work with a medium frequency update.

There are various possible ways to collect the type of data mentioned. The most reliable would be to install inductive loop detectors embedded in road surfaces at city intersections. Such systems are already available and in use in several major cities. For example, the Sydney Coordinated Adaptive Traffic System (SCATS) [7] is available in over 180 cities across 28 countries, including New Zealand, Dublin, Shanghai, and Hong Kong [8]. New York City's Adaptive Traffic Control System (ATCS), Adaptive Signal Control Technology (ASCT), SCOOT, and ACS are other such systems which are used to control traffic signal timings and optimize congestion. Such systems can be easily extended to use with our proposed model since we only use absolute traffic volume counts which such systems are capable of collecting.

Another possible data source can be surveillance cameras assisted with computer vision models [29]. which is already used in practice at some intersections in Shenzhen, China. Several existing studies [30] using state-of-the-art object detection models like YOLO [31] demonstrate effectiveness of this method. Such a model based approach allows for cheap and effective traffic monitoring on top of existing surveillance camera infrastructure.

Other methods like using GPS-enabled mobile phones [10] to track urban traffic flow, as Google does in its Maps product, and the use of probe vehicles [9], which are vehicles equipped with detectors that may be taxis or public transport, can also be used to approximate traffic volume based on their data.

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

The deployed aggregator system will have an RPC and API interface can be interacted with by the remote sensors and other data sources. Each different service can have a separate interface build for interacting with the aggregator service. The aggregator then stores all the data with relevant meta information in a datalake which is a centralized repository that allows you to store all your structured and unstructured data. We make use of a datalake as different sources might have additional information apart from just traffic volume counts and we by allowing to store other complementary information we leave possible avenues for building up further capabilities in our TDT.

An example representing a simplied view of the aforementioned process is shown in Fig. 1.

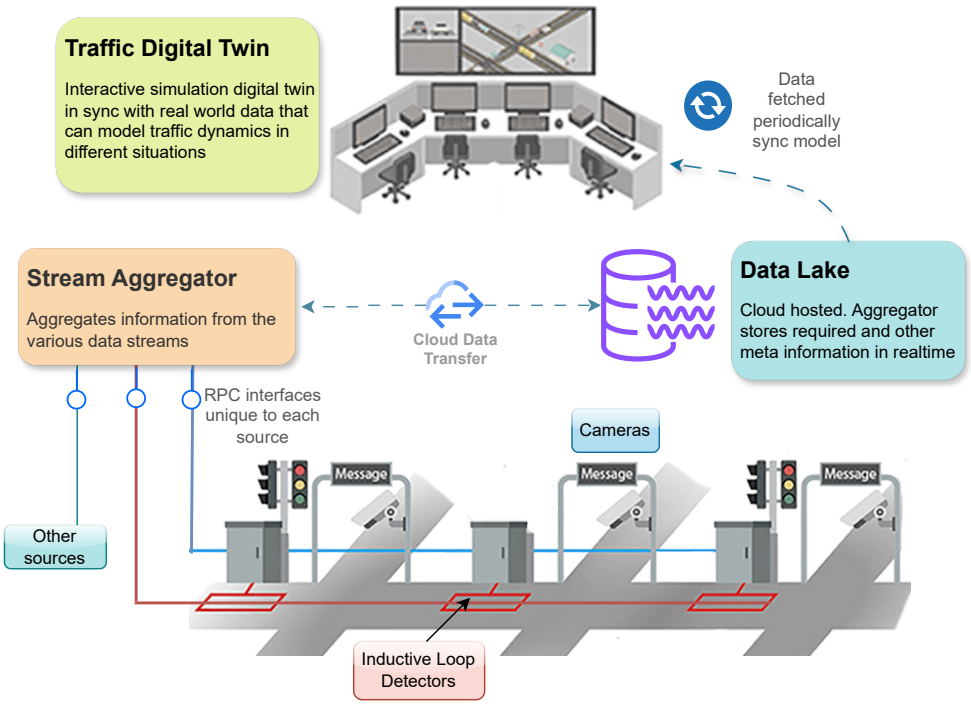


Fig. 1: Traffic Digital Twin (TDT) data processing

2. Model input

For each node, we create a feature vector that represents all the information related to that node. The feature vector for a node is composed of three feature vectors that are as defined ahead, concatenated:

(i) Graph embedding generation using Node2Vec:
In order for the model to effectively learn the spatial relationships between the detector nodes based on graph structure, we need a way to generate an appropriate representation of the graph nodes that capture the neighborhood structure effectively. $Node2Vec$ [15] is a feature learning algorithm that uses second-order biased random walks in a Skip-gram architecture to learn feature vectors for nodes in the graph. There are two key steps for feature generation:

(i) *Sampling using second order biased random walks*: Two hyper-parameters p and q are introduced which are the "return" and "in-out" parameters respectively. Suppose we simulate a random walk of fixed length l. Let c_i denote the ith node in the walk, starting with $c_0 = u$. Nodes c_i are generated by 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 walk that just traversed edge (t, v) and now resides at node v, we now needs to decide on the next step so it evaluates the transition probabilities π_{vx} on edges (v, x) leading from v. The transition probability $\pi_{vx} = \alpha_{pq}(t, x) \cdot w_{vx}$, where

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

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

(ii) *Optimizing objective function using Skip-gram*: Let for graph $G(V, E)$, $f : V \to \mathbb{R}^d$ be the d-dimensional feature representation function. Thus, f is a matrix of size $|V| \times d$ parameters. Now let for every node $u \in V$, we define $N_S(u) \subset V$ as a *network neighbor-*

hood of node u generated in step (i). Define similarity between two nodes u and v as:

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

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

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

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

$$\sum_{u \in V} \log P_f(N_s(u)|u) \quad (3)$$

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

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

(ii) Traffic volume normalised:
The input data should be normalised to increase the training speed and effectiveness. The traffic volume counts at different detectors is normalised as:

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

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

(iii) Word2Vec encoding of other exogeneous variables:
Skip-gram [32], a popular Word2Vec [16] architecture, aims to predict the context words given a target word. It learns to encode the semantic meaning of words by maximizing the probability of context words given the target word. Mathematically, the objective function for Skip-gram can be represented 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.
We use *Word2Vec* to encode the other possibly relevant exogenous variables like weather, road-type etc. as features to the model input, thus increasing its effectiveness. We would like to note that this approach is easily extensible to any number of other features that may prove to be relevant in the future, and we wish to add to the model input.

Concatenating the feature vectors from above subparts, we get the final input vector. Extending the notation as define in section III.A, for N nodes, we define the construction of $\mathbf{x}_i^t$, the feature vector of node i at time t. Assuming *Node2Vec* was used as defined in (i), to generate R^{D_g} dimensional graph embeddings for each node. Let g_i be the graph embedding for node i. Then, for n single-dimensional real-valued features $(r_1, r_2, \ldots, r_n)$, let them be normalized as defined in (ii) to $(r_1^{\text{norm}}, r_2^{\text{norm}}, \ldots, r_n^{\text{norm}})$ thus creating a $D_n = n$ dimensional vector. Let them be represented by h_i^t. Finally, for m "words" representing other semantic information related to the node, and using *Word2Vec* to output d_w dimensional vectors for each word, we get a R^{mxd_w} dimensioned vector that we flatten and represent as $k_i^t \in R^{D_w}$ where $D_w = m \times d_w$.
Finally, one concatenation of the above three, we get the feature vector of node i at time t

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

where $D = D_g + D_n + D_w$. Thus, from (i) and (ii), over T consecutive time steps and for N nodes, we get the final feature vector as $\chi \in \mathbb{R}^{T \times N \times D}$. Depending on the task, we also add other task-specific inputs like for the case of imputation a binary matrix M representing the missing and known values. Specific implementation details are discussed further in the experiments section.

3. Physics Informed Deep Learning

Physics-informed deep learning (PIDL) represents a distinct approach within deep learning (DL), where a neural network is trained to tackle learning tasks while adhering to the principles of physics. By integrating fundamental physical laws as prior knowledge, the resulting neural network serves as a data-efficient approximator, adept at processing input data and delivering accurate predictions.
In situations where systems are required to adhere to the physical laws, PIDL is specially useful as it biases the model to follow the physics models, this allows the model to generate more accurate results, specially GANs as this eliminates several possible predicted distributions that are not viable do to not adhering to the physical laws of the system.
In our situation, specifically with problem 3. of Traffic assignment on edge addition/removal, where we have to predict the changed volume counts on the same timestep but with modified topology, it is imperative, according to the *conservation law* of traffic, that in such a case that the total volume of traffic in a large enough region prior to and after redistribution will be same.

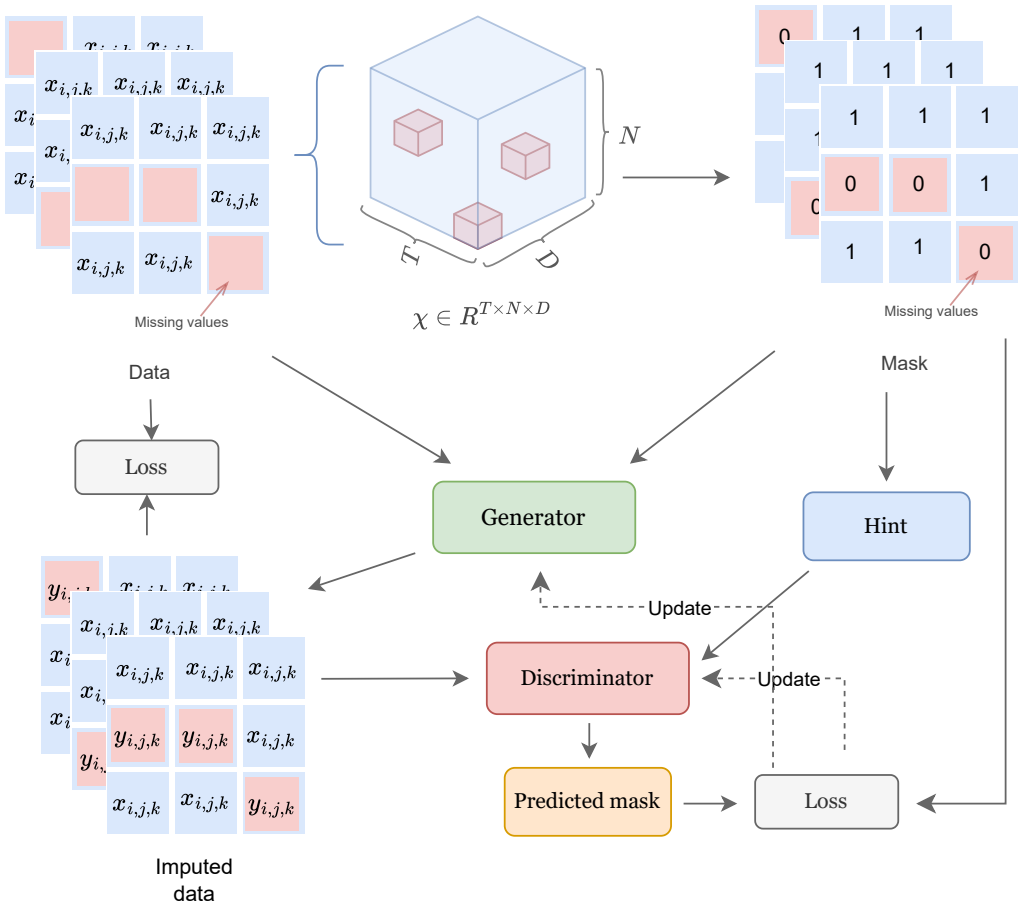


Fig. 2: Model architecture

Formally, for a particular fixed time, let $G(V, E)$ be the original graph, and $G'(V', E')$ be the graph obtained by adding or removing an edge e. Similarly, c_i represents the traffic volume at detector i in G for $i = 1$ to $N = |V|$, and c'_i represents the traffic volume at detector i in G' for $i = 1$ to $N' = |V'|$. Then from *conservation law*:

$$C = \sum_{i=1}^{N} c_i = C' = \sum_{i=1}^{N'} c'_i \tag{5}$$

In order to bias the model for conservation as described in Eq. (3), it needs to be integrated into the learning process of the PIDL network. The simplest way to do it is to define it as an additional term to the loss used to update the model, basically penalizing the model heavily for predictions that do not conform to the physical constraints. With traffic volumes c_i along with other graph information as training input, we establish two different measures for the discriminator to evaluate generator output:

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

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

ii) $\mathcal{L}_{PHY}$: Denoting the conservation loss. As defined in Eq. (3), for a given mask M, let S be the set of detectors such that $M(i) = 0$, specifically $S = \{i \mid M(i) = 0\}$. Then let C_0 be the sum of masked traffic volume counts, i.e., $C_0 = \sum_{i \in S} c_i$, and $Y = \{c'_i \mid i \in S\}$ be the generator output. Then define:

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

In order to control the dominance of the different components of the loss function, we introduce two new hyperparameters, μ and λ, to adjust the weights of $\mathcal{L}_{DL}$ and $\mathcal{L}_{PH}$ respectively. Thus, the final loss function can be defined as:

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

In conclusion, by leveraging hyperparameters, we can fine-tune the model's balance between different objectives. Biasing the discriminator to respect physical laws allows the generator to better capture underlying patterns in the data, a capability we found useful in our experiments as we describe ahead.

4. TraDWin: Traffic Model

In this section, we present the architecture of our TraDWinmodel, which is inspired from architectures of Generative Adversarial Imputation Nets (GAIN) [17] and Wasserstein Generative Adversarial Network (WGAN) [18]. GAIN is a new and popular type of conditional GAN that has been applied to a variety of missing value imputation tasks on different datasets with favorable results. WGAN is an improvement over traditional GANs proposed by M. Arjovsky et al., that is designed to improve training stability by focusing on using Wasserstein distance metric instead of the traditional Jensen-Shannon divergence, along with some other architecture changes like gradient clipping, removal of sigmoid, etc., that have been shown to result in more reliable convergence and better quality samples.

(i) ***Generator:***

The generator G accepts inputs of observed data, with missing values, denoted by $\mathbf{X}$, the binary mask vector $\mathbf{M}$ which indicates what are the missing values, and a noise vector denoted by $\mathbf{Z}$. It produces imputed data $\mathbf{X}_g$, representing a vector of imputations, which is then projected to form the complete imputed output $\mathbf{Y}_0$.

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

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

where $\odot$ denotes the Hadamard product or the element-wise multiplication and $\mathbf{Z}$ is the d-dimensional noise vector. $\mathbf{X}_g$ represents the missing values which are then filled back in $\mathbf{X}$ as per the mask $\mathbf{M}$, and this makes the final generator output that is then passed off to the discriminator to judge.

(ii) ***Discriminator:***

Similar to the discriminator in traditional GANs, we also use another model called the Discriminator, denoted by D, that acts as an adversary to the Generator G and trains it. The discriminator D defined in the original GAIN implementation as $D : \chi \rightarrow [0, 1]^d$, outputs a binary vector instead of a single real value, which denotes what components of the generator input are real (observed) and what are fake (imputed).

So, in contrast to the traditional discriminator as used in GANs, which predicts whether the generated result is entirely fake or real, the discriminator in GAIN predicts what components are real or fake. And the loss which is then used in SGD is the mean of losses for individual components.

(iii) ***Hint***

GAIN introduces a new concept called the hint mechanism, represented by a random variable $\mathbf{H}$ taking values in a space $\mathcal{H}$. The aim of this hint mechanism is to provide additional missing information to the discriminator about the mask. The following proposition was introduced and proved in the GAIN [17] original paper:

> *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 means that there may exist several possible distributions that G may generate that may seem valid to D, so $\mathbf{H}$ must provide enough information to uniquely identify the correct representation of the underlying data, which is what the hint mechanism aims to do.

The hint mechanism depends on the binary mask vector $\mathbf{M}$, and for each imputed sample $(\hat{\mathbf{x}}, \mathbf{m})$, we draw $\mathbf{h}$ from the distribution $\mathbf{H}|\mathbf{M} = \mathbf{m}$. We then add $\mathbf{h}$ as an extra input to the discriminator, resulting in a function $D : \mathcal{X} \times \mathcal{H} \rightarrow [0, 1]^d$, where each component of $D(\hat{\mathbf{x}}, \mathbf{h})$ represents the probability that the corresponding component of $\hat{\mathbf{x}}$ was observed given $\hat{\mathbf{X}} = \hat{\mathbf{x}}$ and $\mathbf{H} = \mathbf{h}$. By varying how $\mathbf{H}$ is defined, we control the level of information it provides about $\mathbf{M}$.

Finally, we take inspiration from the architecture of WGAN and incorporate some of those changes in our model. Traditional GANs suffer from training instability and model collapse, i.e., inability to learn the diverse representations and only generating repetitive samples. Wasserstein GAN (WGAN) proposed by M Arjovsky et. al, solves these problems and we highlight some of the architectural changes we made to GAIN based on WGAN in the following paragraphs.

In traditional GAN the generator and discriminator are trained using the following loss functions:

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

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

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

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

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

Another change in WGANs, that we incorporate in GAIN is the absence of a sigmoid activation function in the discriminator's output layer, unlike in traditional GANs. This alteration enables the discriminator to output unbounded real-valued scores instead of probabilities. Additionally, WGANs incorporate gradient clipping as a regularization technique to enhance training stability. Gradient clipping involves setting a threshold value for the gradients, and if they exceed this threshold, they are scaled down to ensure that their magnitude does not surpass it. This approach helps stabilize the training process and mitigates issues such as exploding gradients.

IV. Experiments

In this section we describe the experimental results of our model along with the necessary parameters required to reproduce those results. We also describe baselines that we compare our model with, along with their parameters.

1. Dataset Description

We use the real world traffic volumes of Dublin city streets collected using the SCATS (Sydney Coordinated Adaptive Traffic System) [7] system over a three month period. The SCATS system is used in Dublin as an adaptive urban traffic management system that synchronises traffic signals and optimises traffic flow across the entire city.

The dataset consists of traffic volumes from the 825 sensors at all key intersections in Dublin city, sampled at a frequency of 1 hour, during a time period of 3 months, from October 1, 2023 to December 31, 2023. The dataset contains the latitudinal and longitudinal geographical coordinates along with timestamped total volume of traffic detected by each detector in the one hour period.

We also verify the model for task (iii) on SUMO [11] TAPASCologne scenario [33], which is a simulation configuration that describes the traffic dynamics of Cologne city in Germany for a day. The scenario is publicly available on the official SUMO website and was created using mobility demand data based on information about travelling habits of citizens and on information about the infrastructure of the area they live in. Since this testing is manual, we run a limited number of tests, where we change an edge and process the simulation, based on the default minimal travel time routing before and after the change, and observe the deviation between simulation and our results.

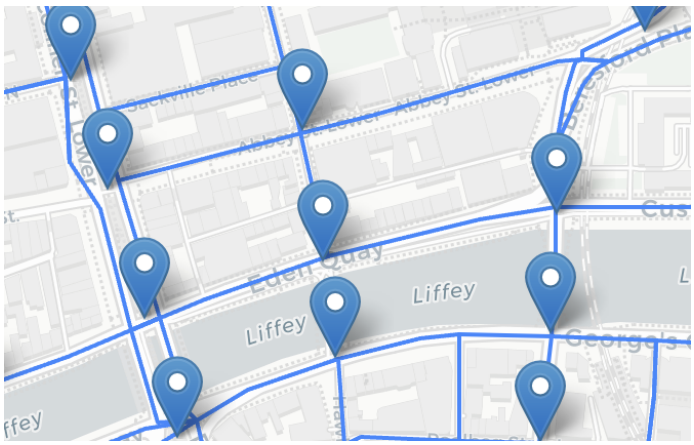


Fig. 3: Snapshot of sensor locations plotted along with road network from Overpass API

2. Model Parameters

For all experiments, we set hyperparameters $T = 3$, which is 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. Formally, during training, we train on sub-graphs consisting of the node clusters instead of the entire graph all 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, specifically for $G(V, E)$, $G_{\text{sub}} = G\{v \in V \mid \text{dist}(u, v) < \sigma\}$. For our experiments, we set σ as 2.5 km, which is a reasonable expectation as clusters usually contain 20 to 30 nodes within this range.

After sampling as stated before, we use a 80-20 test-train split, i.e., the model is trained on 80% of the samples and then performance is verified on new unseen data that comprises of 20% of the samples. The data is then modified based on the task at hand. For imputation, we followed MCAR (Missing Completely at Random) distribution and randomly masked values both spatially and temporally, following a parameter defined as the miss rate, we test 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 basically masking the values along the last timestep dimension. For re-assignment task, i.e., task (iii), we only consider one timestep and re-assign values for that same timestep but with modified spatial geometry.

3. Baselines

We compare our TraDWinmodel with a number of different existing approaches across different domains including different mathematical analysis and deep learning techniques used popularly for these tasks.

One of the simplest approaches for imputation is *KNN*. Similarly, for prediction, *ARIMA* [20] stands for AutoRegressive Integrated Moving Average is used. Matrix factorization methods like *TRMF* [23] (Temporal Regularized Matrix Factorization) use latent factors for predictions, and its modification *BTRMF* [24] (Bayesian Temporal Regularized Matrix Factorization) extends TRMF with a Bayesian framework. For both these methods, we use rank $= 10$, 1000 burn-in iterations, and 200 Gibbs iterations.

LRTC-TNN [34] (Low-Rank Tensor Completion with Truncated Nuclear Norm) is a method for tensor completion. It uses parameters $\rho = 1e-5$, $\theta = 0.25$, and $\epsilon = 1e-4$. *BGCP* (Bayesian Gaussian Process Factorization) is another method that utilizes Bayesian inference. For BGCP, we use similar burn-in and Gibbs iterations, and set rank $= 30$. It is worth noting that these model baselines are only applicable to task (i) and task (ii) and not for task (iii) since they were not built with that task as a consideration.

Finally, there are deep learning methods that we used. For imputation, we employed the *Denoising AutoEncoder* (DAE) [35] model, which is effective in learning meaningful representations of the data while handling missing values. For prediction tasks, we utilized an *LSTM* [36] (Long Short-Term Memory) model as LSTM networks are well-suited for capturing temporal dependencies in sequential data.

4. Evaluation metrics

In order 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:

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

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

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

5. Results

We evaluated the performance of our model on different tasks and compared them with other popular models used in the domain. For the prediction task, we tested the models for predicting traffic volumes at the next timestep, i.e., the forecasting $\delta = 1$, at the third timestep, i.e., $\delta = 3$ and the fifth timestep in future, i.e., $\delta = 5$. This scenario of predicting the next time step requires a mask consisting of all the T_{n+1} values missing, i.e., 0. We prepare a mask similarly for other time horizons. Thus, prediction here is treated as a strict MNAR (Missing Not At Random) subcategory of imputation.

The MAPE and RMSE plots of prediction task on different horizons is shown in Fig. 4 and Fig. 5 respectively. We observer that in general, as the δ i.e. the time horizon to predict in future, increases, both metrics worsen with MAPE falling 5-10% from $\delta = 1$ to $\delta = 3$, i.e., we see a stronger short term accuracy but face 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. We see that ARIMA performs worst, which is to be expected since its the simplest of mathematical models among the baselines we consider. The other matrix factorization and Bayesian models like TRMF, BTRMF and BTMF perform better but given that they are stricly mathematical models, they (i) fail to capture the intricate traffic dynamics that a deep learning model can, and (ii) use only the timeseries traffic data and not the graph topology and external factors that our model considers like weather etc. Finally, we see that LSTMs are close with slightly (2-3%) worse performance which could be attributed to lack of knowledge of graph topology that we incorporate in our model through $Node2Vec$.

Next, we evaluate performance on the imptation task. For this task, we have considered the MCAR (Missing Completely At Random) distribution and compare the models at five missing rates of 10%, 20%, 30%, 40% and 50%. The perfomance of different models measured using MAPE and RMSE is shown in Fig. 6. We see that the conventional methods like KNN, perform poorly to more specialised approaches, which is to be expected since KNN is not enough to capture the intricate traffic dynamics. The matrix factorisation and Bayesian methods which are TRMF, BTRMF and BGCP work slightly better, with deep learning methods like DAE following ahead, though since they deal with only traffic data timeseries with no information on topology they have a 8-13% worse off performane than out model. Tensor completion based model like LRTC-TNN is close to our TraDWinin for low missing rates, but we see that it does not

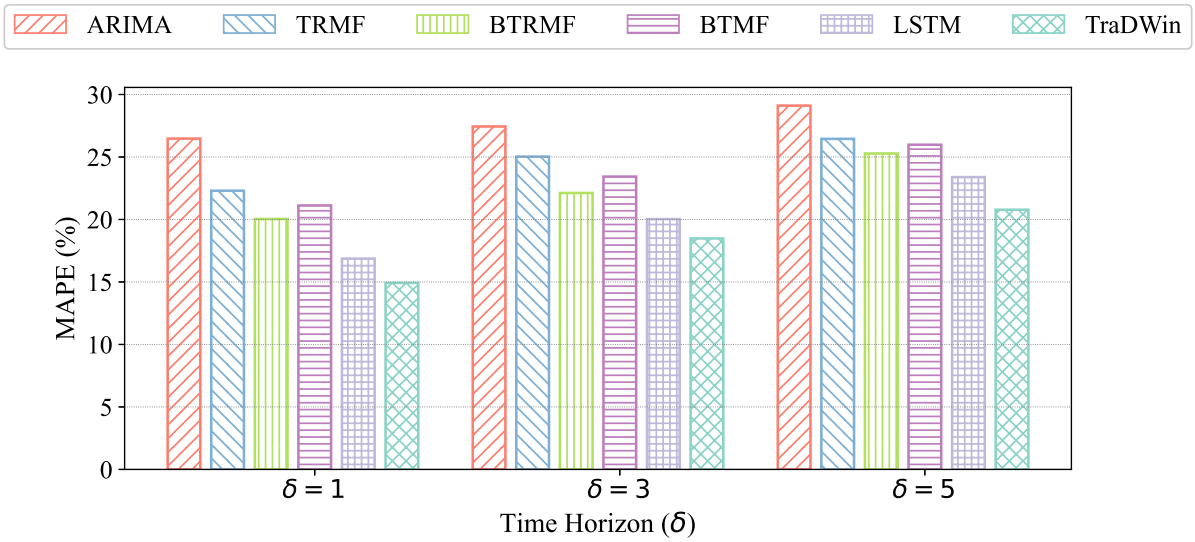


Fig. 4: MAPE on prediction task for different deltas (timesteps to predict ahead)

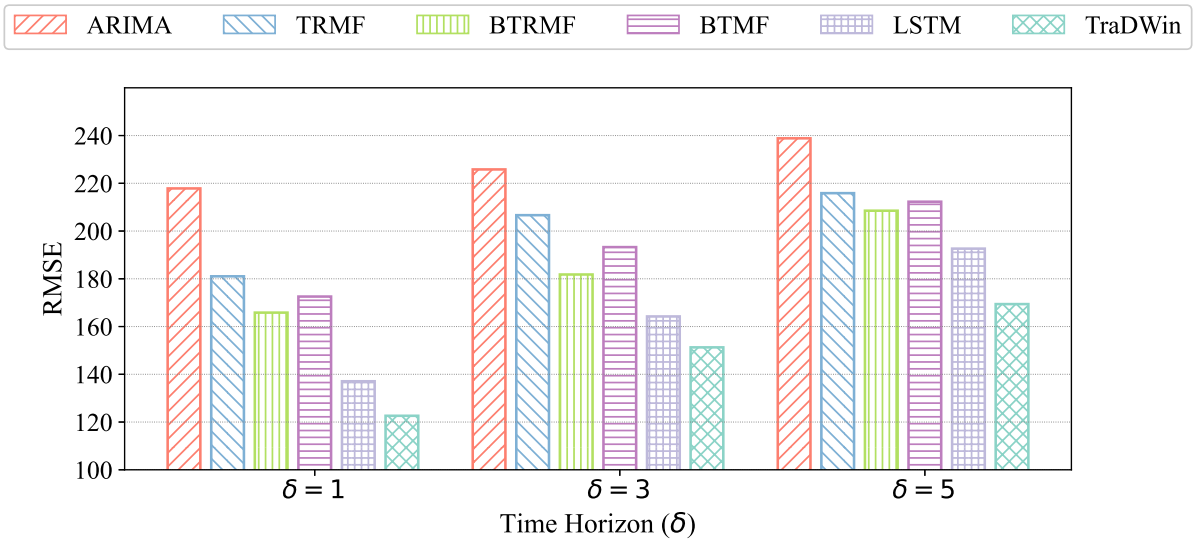
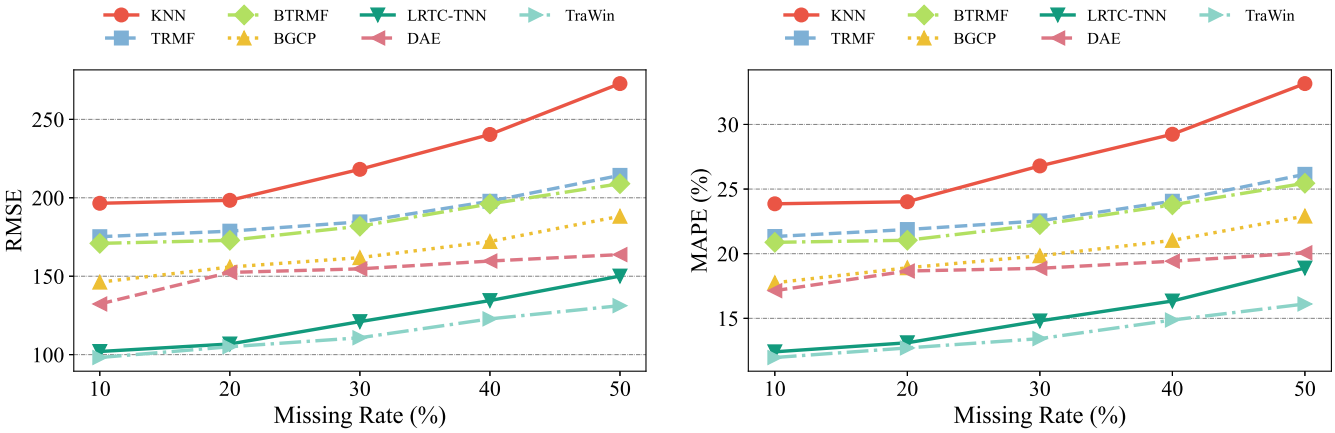


Fig. 5: RMSE on prediction task for different deltas (timesteps to predict ahead)

scale as well with increasing missing rate, possibly because too much data is lost for reliable tensor completion which a deep learning model can still handle, due to being able to learn more complex relationships in data and having knowledge of graph topology and other external factors. Moreover, it is worth nothing that LRTC-TNN is a MCAR imputer and not applicable to task (iii) so not generalised enough for our usecase. For all models, we see that the accuracy of imputations decreases with higher missing rates due to the reduced amount of contextual information available. This is a common challenge in data imputation, where fewer data points make it harder to discern underlying patterns accurately.

We also observe that TraDWinperforms worse on task (i) than task (ii) by around 4%. One primary reason for this, as we suspect, could be that GAIN that was originally designed for MCAR (Missing Completely At Random) imputation, does not generalise well to MNAR imputation scenarios. Another possible reason could be the subgraph-based



(a) RMSE of different models on imputation task

(b) MAPE of different models on imputation task

Fig. 6: Imputation comparison for different missing rates

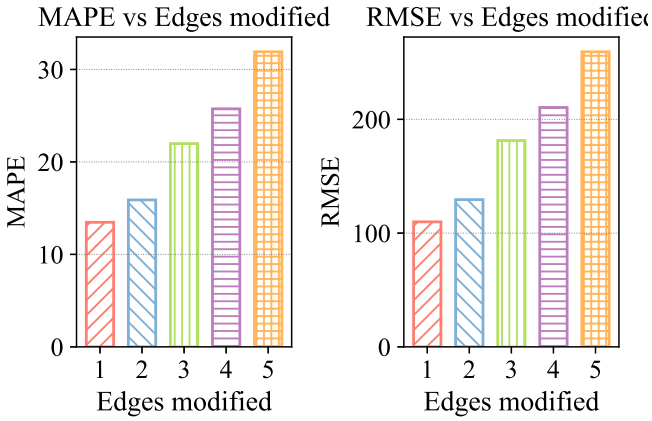


Fig. 7: MAPE and RMSE on re-assignment task vs number of edges modified

computation method that we use, where nodes at the corners of the subgraph do not have adequate spatial and temporal information of their neighbors. Still, we observe that our model performs reasonably and delivers comparable results for being a generalized model.

Finally, on the re-assignment task, which is a new task that we address in our paper and is not conventionally seen in contemporary literature, we essentially train the model for MNAR (Missing Not At Random) imputation scenarios with the central node cluster missing. This effectively makes the model learn how to assign traffic to node clusters given neighbor node information. Based on this training task, we achieve a MAPE of 13.47% and RMSE of 109.90. Note that the other baselines considered for prediction and imputation are not applicable to this without changes to their model architecture so there is not comparison with contemporary models. Moreover, the de-facto way to solve this problem, as we have seen in our literature review, has been simulations like SUMO [11] and Vissim [12]. We too use those to compare and test our model on the TAPASCologne scenario using SUMO, which is a traffic simulation engine, with the configuration as described previously. 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 Dublin SCATS dataset and 15.06% on the TAPASCologne scenario, which demonstrates that the model can, with reasonable accuracy, learn to solve the traffic assignment problem on node clusters. Further, we tried to evaluate how our model's performance scales with number of modifications, i.e., how much can we alter the original graph such that the model is still viable to use for re-assignment. In Fig. 7, we show a comparison of MAPE and RMSE with number of edges modified. We see that performance declines steadily from 13.47% at 1 change to 31.91% at 5 changes. So, altering more of graph leads to decline in performance, with drop being significant and sharp after 3 modifications. A possible reason for this could be the mass masking strategy we use to train the model for task (iii). For 5 modifications, too many of the neighbouring nodes are masked leading to large loss of contextual information that knowledge of graph representation does not compensate enough.

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

V. Conclusion

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

Future Work

It is worth noting that the core architecture of our TDT model is the same for all three underlying tasks. In our approach, we deal with each task as a separate problem for the model to be trained upon. Given that multi-task learning for models, where the model is able to 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] 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.

[14] ——, "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.

[15] 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.

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

[17] 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.

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

[19] 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.

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

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

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

[32] 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.

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

[34] 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.

[35] 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.

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