TU Delft

Delft University of Technology

Document Version
Final published version

Licence
Dutch Copyright Act (Article 25fa)

Citation (APA)
Jiang, K., Yan, F., Qiao, Y., Li, M., Li, Y., & Conti, M. (2025). Network Traffic Matrix Imputation via Large Language Models. In *30th IEEE Symposium on Computers and Communications, ISCC 2025* IEEE. https://doi.org/10.1109/ISCC65549.2025.11326227

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
In case the licence states “Dutch Copyright Act (Article 25fa)”, this publication was made available Green Open Access via the TU Delft Institutional Repository pursuant to Dutch Copyright Act (Article 25fa, the Taverne amendment). This provision does not affect copyright ownership.
Unless copyright is transferred by contract or statute, it remains with the copyright holder.

Sharing and reuse
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.

Network Traffic Matrix Imputation via Large Language Models

Kaiwen Jiang[†‡], Fenglin Yan[†‡], Yan Qiao,[*‡] Meng Li[*‡], Yuxuan Li[§], Mauro Conti[¶‖]
[‡]School of Computer Science and Information Engineering, Hefei University of Technology, Hefei, China
[§]Faculty of Computer and Mathematical Science, The Hong Kong Polytechnic University, Hong Kong, China
[¶] Department of Mathematics and HIT Center, University of Padua, Padua, Italy
[‖]Faculty of Electrical Engineering, Mathematics and Computer Science Delft University of Technology, Delft, Netherlands
Emails: {2022212170, 2022212172}@mail.hfut.edu.cn,{qiaoyan, mengli}@hfut.edu.cn
24105846d@connect.polyu.hk, mauro.conti@unipd.it

2025 IEEE Symposium on Computers and Communications (ISCC) | 979-8-3315-2420-3/25/$31.00 ©2025 IEEE | DOI: 10.1109/ISCC65549.2025.11326227

***Abstract*—Large Language Models (LLMs) have demonstrated remarkable zero-shot capabilities across various domains. This paper pioneers the application of LLMs' outstanding knowledge and reasoning abilities to the challenging task of Traffic Matrix (TM) imputation. However, the application poses significant challenges due to the skewed TM distribution and the deficient traffic feature under low sampling rate. To address these issues, we propose TM-LLM, the first LLM-based model specifically designed for TM imputation. Our approach includes two critical designs: Firstly, we develop an adversarial training strategy to pre-impute TM data, allowing the LLM to understand the distributional features even when faced with extensive missing data. Secondly, we devise a TM-specific embedding scheme along with a crafted prompt template, which enables our approach to harness LLMs' exceptional inferential ability. Experimental results show that TM-LLM significantly outperforms state-of-the-art imputation methods, achieves a notable 16.5% - 44.8% improvement in accuracy over the current best baseline, while reduces measurement costs by 80% - 96%. It can accurately capture the traffic pattern even when the sampling rate is extremely low. The code for reproducing our experiments is publicly available[1]. These findings strongly indicate the breakthrough potential of LLMs in network TM analysis tasks.**

***Index Terms*—Network monitoring, traffic matrix imputation, adversarial learning, large language models.**

I. Introduction

Traffic Matrix (TM) is adopted to monitor the traffic volumes between all origin-destination (OD) pairs in a network over a given period [1]. Accurate observations of the TMs are essential for many network operation tasks such as network capacity planning, anomaly detection, and traffic engineering [2]. However, as the network increasingly expands, the network-wide OD pairs surge exponentially [3]. Frequently measuring these traffic volumes not only consumes vast amounts of ternary content addressable memory (TCAM) in routers, but also disrupts other network operations.

One promising solution is to measure a few traffic volumes in the TM, and impute the remaining volumes based on the observed ones, which is known as the TM imputation task [4] [5]. Traditional TM imputation methods apply low-rank matrix/tensor decomposition techniques to recover the incomplete TMs by leveraging the sparsity of TMs [6]. Despite the effectiveness of these methods, they can only capture the linear and simple correlations among OD flows while consuming high computational resources. With the rapid advancement of neural network technologies, a few neural-based TM imputation methods have emerged [7] [8]. On one hand, the neural-based methods can capture the nonlinear relationships within TM, enhancing the imputation accuracy [7] [8]. On the other hand, they can output the completed TMs within linear computational time after appropriately model training.

Recently, the remarkable progress in Large Language Models (LLMs) led to their application in various domains, achieving a significant progress [9] [10]. Currently, the application of LLMs for time-series (TS) analysis is at a nascent stage [11]. Many research works had revealed LLMs can better understand the TS patterns due to their zero-shot ability and the temporal sequence discern ability [12] [13]. Most of these works applied LLMs to address TS forecasting tasks and achieved remarkable breakthroughs. TM data, which records the traffic volumes over a time period, can also be considered as a type of TS data. Nevertheless, directly applying LLMs to deal with the TM imputation problem yet leads to unexpectedly low accuracy. This attributes to two reasons. First, LLMs were only shown to perform well within low-dimensional data with strong periodicity [13], while the TM data has much higher dimension than general TS data and presents significantly skew distribution. Second, different from TS forecasting, where the task pattern resembles natural language tasks, the TM imputation task involves filling in large gaps (e.g., often more than 90%) of missing data with only a few known entries. The missing data is random and compromise the pattern of the dataset, thereby posing challenges for LLMs to fully comprehend this task.

To address the above issues while harnessing the latent reasoning capabilities of LLMs for TM imputation, we make the first attempt to develop a LLM-based TM imputation model (named TM-LLM). Specifically, we first craft the input of the LLMs, integrating the embeddings of TM data and the prompt with TM side knowledge. Then we design a lightweight fine-tuning strategy for handling TM imputation. To enable the

[†] Kaiwen Jiang and Fenglin Yan contributed equally to this work.
[*] Yan Qiao and Meng Li are the corresponding authors.
[1]The experimental codes with our methods and the datasets are available at https://github.com/FILingK/TM-LLM

LLMs to well understand the TM data with a large proportion of missing data, we devise an adversarial learning strategy to preliminarily impute the missing data under the supervision of a discriminator. In this way, the incomplete TM will have the similar pattern with the authentic TM, thereby harnessing the full potential of LLMs for TM imputation. In summary, the main contributions of this paper are summarized as follows.

- We propose TM-LLM, the first LLM-based TM imputation approach, making a significant breakthrough in TM imputation performance. We develop both the LLM input and the alignment strategy, to fully activate the extensive knowledge base of LLM to deepen the understanding of TM imputation task.
- We develop an adversarial learning strategy to assist the LLM in understanding traffic patterns in severely incomplete TMs. We employ a discriminator to supervise a generator in producing complete TMs similar to authentic TMs. Then, we use the discrepancies between the generated and real values to enhance the LLM's proficiency in the task of TM imputation.
- We conduct extensive experiments on two real-world TM datasets. The results show that when there is 50% missing data, TM-LLM can improve the imputation accuracy by $20.6\% \sim 33.3\%$, whereas the improvement can achieve up to 44.8% when the missing rate increases to 98%. Besides, the ablation study demonstrates the efficacy of the distinctive designs in TM-LLM.

II. Related Work

We provide an overview of existing research on network TM imputation and highlight the key differences between our approach and previous studies.

A. Low-Rank Matrix/Tensor Decomposition-Based Methods

Low-rank matrix/tensor decomposition was the initial approach used for TM imputation, which applies matrix decomposition to access the principle components of TMs [4] [6]. Sparsity Regularized Matrix Factorization (SRMF) proposed in [4] used K-Nearest Neighbors (KNN) to capture the similarities in traffic volumes across different OD pairs, and constructed a spatial constraint matrix to enhance accuracy. Combined Optimized Layered Design (COLD) proposed in [6] recovered the TMs using maximum entropy principle. Tensor-based methods are regarded as more effective in capturing multi-dimensional correlations [14]. Xie et al. [15] exploited local data correlations by constructing sub-tensors with lower ranks for precise TM imputation. Tian et al. [16] proposed a tensor-based TM imputation approach that structures continuous distance data into sliding-window-based tensors and incorporates a lightweight sample selection algorithm with a lower sampling bound, along with a fast and accurate inference algorithm for unmeasured data. However, the above methods often cost high computational resources to perform the decomposition operation. They also struggle to capture the complex correlations between OD pairs through linear decompositions, resulting in limited imputation performance.

B. Neural-Based TM Imputation

To better model the complex correlations among the traffic volumes, a few recent works adopt neural networks to TM imputation task. Neural Tensor Completion (NTC) proposed in [17] schemed to model three-order interaction among traffic features with the outer product and built a 3D interaction map which can significantly reduce the error in missing data recovery. Deep Adversarial Tensor Completion (DATC) proposed in [7] built the TM data into tensors and applied a generative adversarial network (GAN) to perform the tensor imputation task. LightNestle [18] used a meta-learning-based framework and spatiotemporal knowledge transfer to perform the imputation efficiently. D-NORM [19] performed a distributed neural tensor completion using convolutional neural networks (CNNs).

C. LLM-based Time Series (TS) Analysis

In recent years, LLMs have been applied to the analysis of TS data, achieving state-of-the-art results. Zhou et al. [20] proposed the first LLM-based TS analysis approach, which can be applied to various TS tasks, such as classification, anomaly detection, and forecasting. Chang et al. [21] proposed LLM4TS, a pre-trained LLM with two-stage fine-tuning, which is used for predicting TS data in fields of economic planning and weather forecasting. Yu et al. [22] used LLMs to predict financial TS data. Amine Elhafsi et al. [23] applied LLMs for semantic anomaly detection. Luca Zanella et al. [24] adopted LLMs to video anomaly detection tasks. However, TMs in computer networks exhibit more skewed distribution along with much higher missing rate, thereby posing challenges to the comprehension capacity of LLMs. Currently, the LLMs have not been used in network TM imputation task.

In this paper, we aim to overcome the challenges in applying LLMs to TM imputation task, and leverage the powerful capacities inherent in LLMs to enhance the imputation performance.

III. Methodology

In this section, we present our TM-LLM, which leverages LLMs to recover the severely missing TMs in networks.

A. Problem Formulation

We model the network as graph $G = (V, E)$, where V and E represent the nodes and links in the network, respectively. Generally, the TM is defined as a $|V| \times |V|$ matrix, with each entry representing an OD flow between a pair of nodes in the network. For ease of computation, we reshape a series of TM data into a matrix $\mathbf{X} = \{X_1, X_2, \ldots, X_\Omega\} \in \mathbb{R}^{N \times \Omega}$, where $X_i = \{x_{1i}, x_{2i}, \ldots, x_{Ni}\}^T$, $i = \{1, 2, \ldots, \Omega\}$, is a flattened TM that contains $N = |V| \times |V|$ OD flows. Ω corresponds to the total number of measurement time points. We use a sampling indication matrix $\mathbf{M} \in \mathbb{R}^{N \times \Omega}$ to indicate the observed measurements in $\mathbf{X}$. If x_{ij} is observed, $m_{ij} = 1$; otherwise, $m_{ij} = 0$. $\mathbf{M} \odot \mathbf{X}$ indicates the TM with only the observed values, where $\odot$ represents the element-wise product. In this paper, our goal is to recover all missing values in $\mathbf{X}$ given the observed values.

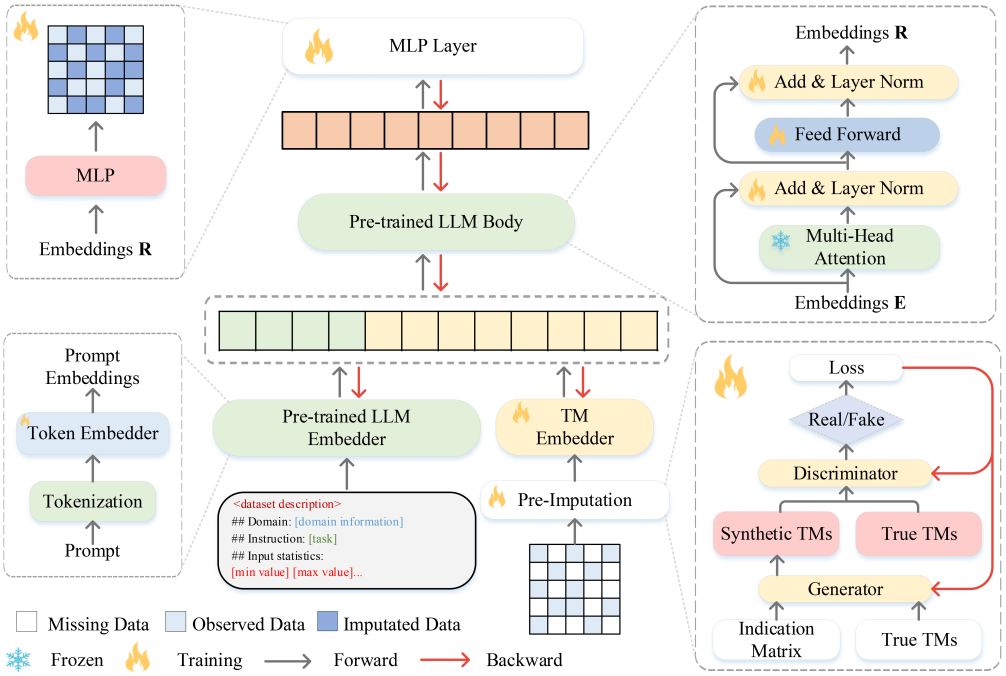


Fig. 1. The framework of TM-LLM: Imcomplete TMs are pre-imputed and embedded via TM Embedder, while prompts are embedded through LLM Embedder. The two embeddings are concatenated and processed by Pre-trained LLM Body. The outputs are projected onto complete TMs via MLP layer. Multi-Head attention layers in Pre-Train LLM are frozen while the embedding layer, normalization layers and feedforward layers require fine-tuning.

B. Method Overview

The framework of TM-LLM is depicted in Fig 1. The framework can be divided into two stages: the pre-imputation through adversarial generating and the comprehensive imputation through LLM. In the first stage, we employ a 2-D Generator to construct preliminary TMs based on the partially observed TMs under the supervision of a Discriminator. In the second stage, we first embed the preliminary TMs and combine it with the task-specific prompt as the input of the LLM. Then we add a Multi-Layer Perceptron after the output of LLM, and fine-tune the LLM with the designed imputation objective.

C. Pre-Imputation with Adversarial Learning

Since the missing data in TMs may severely compromise the spatiotemporal correlation of OD flows in TMs, thereby impeding LLM's comprehension of TMs pattern. We design a pre-imputation module using adversarial learning to produce the synthetic TMs, where the missing values are synthesized to enforce the pre-imputed TMs have the similar spatiotemporal characteristics with the real TMs.

The pre-imputation module contains a Generator and a Discriminator. The Generator first encodes $\mathbf{M}\odot\mathbf{X}$ into a latent representation $\mathbf{H}$ through a series of 2-D convolutional layers. Then, it decodes $\mathbf{H}$ through another series of convolutional layers to produce a dense reconstruction $\hat{\mathbf{X}}$. Let f_e and f_d represent the non-linear functions corresponding to the encoding and decoding processes, respectively, with parameters Θ_e and Θ_d. The latent representation of $\mathbf{X}$ is given by $\mathbf{H} = f_e(\mathbf{M}\odot\mathbf{X};\Theta_e)$. While the dense reconstruction $\hat{\mathbf{X}}$ is obtained as:

$$\hat{\mathbf{X}} = f_d(\mathbf{H};\Theta_d) = f_d(f_e(\mathbf{M}\odot\mathbf{X};\Theta_e);\Theta_d). \tag{1}$$

After $\hat{\mathbf{X}}$ is outputted by the Generator, we replace the observable entries by their measurement values, i.e.,

$$\hat{\mathbf{X}} = \mathbf{M}\odot\mathbf{X} + (\mathbf{I}-\mathbf{M})\odot\hat{\mathbf{X}}, \tag{2}$$

where $\mathbf{I}\in\mathbb{R}^{N\times\Omega}$ denotes the all-one matrix.

The Discriminator takes the synthetic and true TMs as input, and outputs an indicator that represents Fake/Real. Similar to the Generator, the Discriminator initially increases the number of channels through two convolutional layers, followed by a pooling layer that reduces both the temporal steps and feature dimensions. Subsequently, two additional convolutional layers are employed to progressively reduce the number of channels, followed by another pooling layer to further diminish the other two dimensions. Finally, the output is passed through six fully connected layers and a Sigmoid function to determine the probability of the input being real or fake.

The pre-imputation module is trained in an adversarial manner, where the Generator and the Discriminator are trained alternately. We define two loss functions $Loss_{Dis}$ and $Loss_{Ge}$ for Discriminator and Generator, respectively. $Loss_{Dis}$ includes two parts: One part guides Discriminator to identify the synthesized TM $\hat{\mathbf{X}}$ as Fake (denoted by 0), whereas the other part forces it to recognize the true TM $\mathbf{X}$ as Real (denoted by 1). We employ Binary Cross Entropy (BCE) to measure the discrepancy between the output and the target. Then $Loss_{Dis}$ can be formally expressed as:

$$\begin{aligned}\text{Loss}_{Dis} &= \text{BCE}(0, f_{dis}(\hat{\mathbf{X}})) + \text{BCE}(1, f_{dis}(\mathbf{X}))\\ &= -\log(1-f_{dis}(\hat{\mathbf{X}})) - \log(f_{dis}(\mathbf{X})),\end{aligned} \tag{3}$$

where f_{dis} represents the process function of Discriminator.

$Loss_{Ge}$ also contains two components: One is designed to produce indistinguishable TMs that fools the Discriminator to identify the synthesized TMs as real, while the other penalizes the discrepancies between the synthesized TMs and the real TMs. The formula for $Loss_{Ge}$ can be written as:

$$\begin{aligned} \text{Loss}_{Ge} &= \text{BCE}(1, f_{dis}(\hat{\mathbf{X}})) + \gamma||\hat{\mathbf{X}} - \mathbf{X}||_2^2 \\ &= -\log(f_{dis}(\hat{\mathbf{X}})) + \gamma||\hat{\mathbf{X}} - \mathbf{X}||_2^2, \end{aligned} \tag{4}$$

where γ is the weight parameter, and $||\cdot||_2^2$ denotes L2 norm.

D. Comprehensive Imputation Using LLMs

We focus on using an embedding-visible language foundation model, such as DeepSeek-R1 [25], Llama-3 [26] and GPT-2 [27], for comprehensive TM imputation. In this stage, we consider the following problem: Given the pre-imputed TM and a sampling matrix, we aim to let LLMs well understand the TM pattern and refine the missing data in TMs. The comprehensive imputation involves three elements: TM embeddings, a TM imputation prompt template, and lightweight fine-tuning for task alignment.

1) TM Embeddings: In order to feed the pretrained LLMs with the TM data, it is essential to project the TM data into a vector space that the LLMs can interpret. In TM-LLM, we design an embedding layer with Long Short-Term Memory (LSTM) that maps $\hat{\mathbf{X}} \in \mathbb{R}^{N\times\Omega}$ into its embeddings $E_{TM} \in \mathbb{R}^{N\times D}$. The embedding process can be denoted by f_{Emb}, where the parameters in the embedding layer (denoted by Θ_{emb}) will be trained during the fine-tuning.

> The Abilene dataset contains 12 routers, 30 directed inner links, and 24 outside links. The dataset collected the volumes of all OD flows in the network every 5 minutes from March to September 2004. Below is the information about the input time series:
>
> [BEGIN DATA]
> [Domain]: The dataset consists of OD flow pairs between 12 routers (v1 to v12), with a total of 144 OD flows (x1 to x144). Each OD pair has a time step length of [seq len]. Each flow x represents the volume of traffic between two routers in the network, where x1 is the flow from v1 to v1 …
> [Instruction]: Learn the temporal relationships within the complete data of each row, as well as the traffic relationships between different rows, to infer and fill in the missing data.
> [Statistics]: The input has a minimum if [min value], a maximum of [max value], and a median of [median value]. The missing rate of data is [unobserved rate].
> [END DATA]

Fig. 2. Example of a prompt template for TMs in the Abilene dataset: [] represents task-specific configurations and data statistics. [] marks the start and end of the Load. [] indicates the start of a specific type of prompt.

2) Prompt Design: Given our goal of applying pre-trained language models for TM imputation task, we need to craft an appropriate prompt to elicit the reasoning and understanding of LLMs. The designed prompt contains a Head that describes the dataset information, and a Load that contains the following three parts - Domain, Instruction, and Statistics. Domain specifies the data format, Instruction describes the task requirements, and Statistics provide the mathematical information about the data. Such design allows the LLMs to understand the correlations among the OD flows as well as the key features of TMs, thereby enhancing the accuracy of imputation. Taking the Abilene dataset as an example, as shown in Fig. 2, the Head includes the dataset name, the network topology, and the sampling method. The Load comprises Domain (including the size and properties of TM), Instruction (including task guidelines) and Statistics (including minimum, maximum, median, and the missing rate of TM).

For each TM dataset, we employ the tokenizer built into the specific model to tokenize the prompt into tokens. Then these tokens are embedded into a $P \times D$ matrix, denoted by $E_{prompt} \in \mathbb{R}^{P\times D}$. Finally, we concatenate the TM embeddings E_{TM} with the prompt embeddings E_{prompt}, resulting in the integrated embeddings $E \in \mathbb{R}^{(P+N)\times D}$ as the input to LLMs.

3) Lightweight Alignment: To preserve the pre-trained knowledge of LLMs while enabling them to handle the imputation task, we perform a lightweight fine-tuning to align the LLMs with the specific requirements of the imputation task. We froze the multi-head attention layers of LLMs and only adjust the small among of the weight matrix of positional encoding in the embedding layer (denoted by Θ_{wpe}), the parameters in normalization layers (denoted by Θ_{norm}), and the parameters in feed-forward layer (denoted by Θ_{ff}).

In the output of LLMs, we discard the prefix portion and retain the remaining part as the representation for TM. To reconstruct the TM from its representation, we flatten the representing vector and apply a MLP layer, denoted as f_{MLP}, to produce the final predictions $\tilde{\mathbf{X}}$. The set of parameters in the MLP layer is denoted by Θ_{mlp}, which will be trained during the fine-tuning process. The objective of the fine-tuning can be formulated by

$$\min_{\Theta_{emb},\Theta_{wpe},\Theta_{norm},\Theta_{ff},\Theta_{mlp}} ||(\mathbf{I} - \mathbf{M}) \odot (\tilde{\mathbf{X}} - \mathbf{X})||_2^2. \tag{5}$$

After fine-tuning, the framework can be used to impute the missing data in the TMs efficiently. The complete imputation algorithm is presented in Alg. 1. The algorithm takes a group of TMs $\mathcal{X}$ and a group of sampling matrices $\mathcal{M}$ as inputs, and outputs the complete TMs $\tilde{\mathcal{X}}$. For each $\mathbf{X}_t$ in $\mathcal{X}$, a initial imputation is performed to obtain a preliminary $\hat{\mathbf{X}}_t$ (line 2). Then, $\hat{\mathbf{X}}_t$ is transformed into embeddings $\mathbf{E}_{TM}^t$ through TM embedding layer (line 3). Next, a prompt for $\hat{\mathbf{X}}_t$ is generated using the designed prompt template and embedded into $\mathbf{E}_{prompt}$ through LLM Embedder (line 4). The two embeddings are then concatenated to $\mathbf{E}$ as the input to LLM (line 5). After processing $\mathbf{E}$ through LLM, it generates a representation of the embeddings $\mathbf{R}$ (line 6). Finlly, $\mathbf{R}$ is tailored and projected to the final output $\tilde{\mathbf{X}}$ through the MLP layer (line 7).

IV. Experiments

In this section, we compare the performance of TM-LLM with five state-of-the art baselines on two benchmark datasets.

A. Experimental settings

1) Data Descriptions: We utilize the real-world network TM data from two sources: the U.S. Internet2 Network (Abilene [28]) and the European Research and Educational Network (GÉANT [29]). Both of them have been widely used in TM-related tasks [4] [30]. The Abilene dataset records TM data every 5 minutes over a span of 168 days, while the GÉANT

Algorithm 1 TM Imputation via Fine-Tuned TM-LLM

Input: $\mathcal{X} = \{\mathbf{X}_1, \ldots, \mathbf{X}_T\}, \mathcal{M} = \{\mathbf{M}_1, \ldots, \mathbf{M}_T\}$
Output: Imputed $\tilde{\mathcal{X}} = \{\tilde{\mathbf{X}}_1, \tilde{\mathbf{X}}_2, \ldots, \tilde{\mathbf{X}}_T\}$

for t from 1 to T **do**
  //Pre-Imputation
  $\hat{\mathbf{X}}_t \leftarrow f_d(f_e(\mathbf{M}_t \odot \mathbf{X}_t; \Theta_e); \Theta_d)$
  //Comprehensive Imputation
  $\mathbf{E}^t_{TM} \leftarrow f_{Emb}(\hat{\mathbf{X}}_t, \Theta_{emb})$;
  $\mathbf{E}^t_{prompt} \leftarrow LLMEmbedder(Prompt(\hat{\mathbf{X}}_t), \Theta_{wpe})$;
  $\mathbf{E} \leftarrow concatenate(\mathbf{E}^t_{TM}, \mathbf{E}^t_{prompt})$;
  $\mathbf{R} \leftarrow f_{LLM}(\mathbf{E}, \Theta_{norm}, \Theta_{ff})$;
  $\tilde{\mathbf{X}}_t \leftarrow f_{MLP}(\mathbf{R}, \Theta_{mlp})$;
end for
return $\tilde{\mathcal{X}}$

dataset collects TM data every 15 minutes over 112 days. Referring to the experimental settings in [7], we select the first 3000 rows of the two datasets, and split them into training, validation, and testing sets in an 8:1:1 ratio.

2) Metrics: We apply four metrics to evaluate the imputation performance: Normalized Mean Absolute Error (NMAE), Normalized Root Mean Square Error (NRMSE), Kullback-Leibler Divergence (KL), and Normalized Discounted Cumulative Gain (NDCG@k). The formula of each metric is listed in Table I. NMAE and NRMSE are used to assess the recovery accuracy. Achieving smaller values for these two metrics results in higher accuracy. KL is used to evaluate whether the recovered data follows the distribution of the real TM data. A smaller KL value indicates the recovered data is closer to the true distribution. NDCG@k is applied to evaluate how accurately the top-k elephant flows can be inferred. A larger NDCG@k indicates a more accurate recovery of the top-k flows.

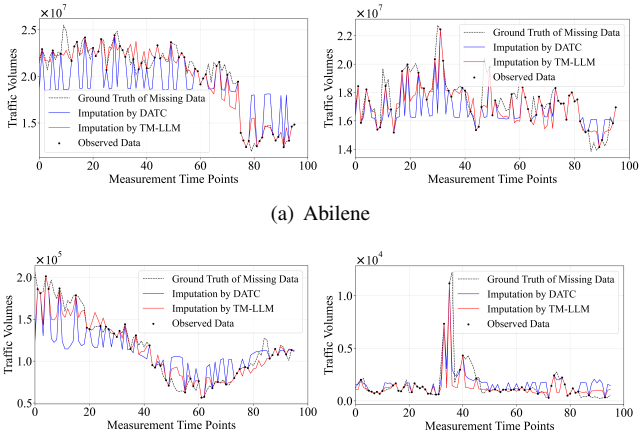
In Table I, x_{ij} and $\hat{x}_{ij}$ denote the real and the predicted values of the j-th OD flow at the i-th time interval in $\mathbf{X}$ and $\hat{\mathbf{X}}$, respectively. Φ and $\overline{\Phi}$ denote the index set of the observed and un-observed data, respectively. Only the un-observed data is considered in the calculation of these metrics. In the formula of NDCG@k, x represents a flattened vector of TM in a particular time slot, and $\hat{\pi}$ is a permutation of x. $\hat{\pi}_i$ indicates the i-th element in $\hat{\pi}$. π is derived by sorting x in decreasing order. After computing NDCG@k for each time slot, the final result is the average value across all time slots. Given the skewed distribution of network TM data, we set $k = 10$, following the settings in reference [7].

3) Baselines: Besides TM-LLM, we implement 5 state-of-the-art imputation methods as baselines, including DATC [7], NTC [17], NTM [31], NTF [32], and CoSTCo [33]. Among them, DATC and NTC are used for network TM imputation, while NTM, NTF, and CoSTCo are designed for imputation tasks in recommendation systems.

Our method is implemented in PyTorch, and all experiments are conducted on a single NVIDIA Tesla A800 80G GPU. We employ GPT-2 as our base pre-trained model with only 10 epochs of fine-tuning. The records below are averaged over 10 runs. The detailed settings of our experiments are available at https://github.com/FILingK/TM-LLM.

TABLE I
PERFORMANCE METRICS

Method	Formula
NMAE	$\frac{\sum_{(i,j)\in\overline{\Phi}} \lvert x_{ij} - \hat{x}_{ij} \rvert}{\sum_{(i,j)\in\overline{\Phi}} \lvert x_{ij} \rvert}$
NRMSE	$\frac{\sqrt{\sum_{(i,j)\in\overline{\Phi}} \left(x_{ij} - \hat{x}_{ij}\right)^2}}{\sqrt{\sum_{(i,j)\in\overline{\Phi}} x_{ij}^2}}$
KL	$\sum_{(i,j)\in\overline{\Phi}} x_{ij} \log\left(\frac{x_{ij}}{\hat{x}_{ij}}\right)$
NDCG@k	$\text{NDCG@k}(x, \hat{\pi}) = \frac{\sum_{i=1}^{k} \frac{2^{x[\hat{\pi}_i]}-1}{\log(i+1)}}{\sum_{i=1}^{k} \frac{2^{x[\pi_i]}-1}{\log(i+1)}}$



(a) Abilene

(b) GÉANT

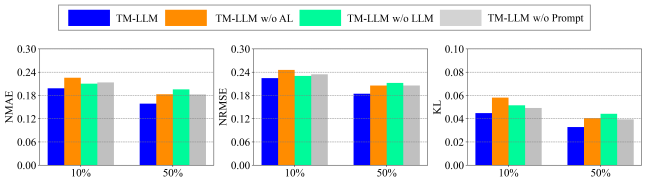
Fig. 3. Visualizations of imputation performance.

B. Results

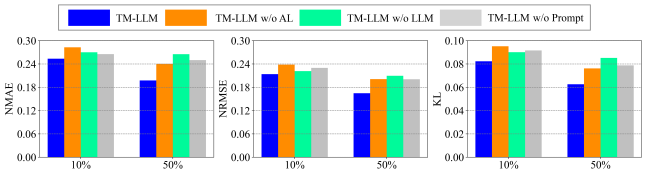
1) Comparison with baselines: Table II summarizes the results of the six TM imputation methods on the two datasets. From the table, TM-LLM significantly outperforms the other five baselines on all metrics. Specifically, comparing with the optimal baseline DATC, TM-LLM shows a $22.0\% \sim 33.3\%$ improvement on Abilene and a $16.5\% \sim 44.8\%$ improvement on GÉANT in NMAE, as well as a $28.1\% \sim 40.1\%$ improvement on Abilene and a $5.5\% \sim 37.3\%$ improvement on GÉANT in NRMSE, across sampling rates ranging from 2% to 50%. It is particularly noted that in Abilene, when the sampling rate is 2%, the NMAE and NRMSE of TM-LLM are even better than those of DATC under the sampling rate of 50%, thereby reducing the sampling cost by 96%. That indicates TM-LLM is especially suitable for TM imputation under very low sampling rate. In addition, TM-LLM also remarkably improves the other baselines in terms of KL divergence. This demonstrates the TMs produced by TM-LLM closely align with the real TM pattern. In both datasets, TM-LLM shows stably good performance on NDCG@10 under various sampling rates. This indicates our method excels in the task of traffic imputation for skewed traffic distributions. We visualize the imputation performance of TM-LLM and DATC in Fig. 3 under a sampling rate of 50%. From the figures, TM-LLM demonstrates a distribution more closely aligned with the ground truth compared to DATC.

TABLE II
IMPUTATION PERFORMANCE ON ABILENE AND GÉANT

Dataset		Abilene						GÉANT					
Metric	Model	SR 2%	SR 4%	SR 6%	SR 8%	SR 10%	SR 50%	SR 2%	SR 4%	SR 6%	SR 8%	SR 10%	SR 50%
NMAE	TM-LLM	**0.2171**	**0.2101**	**0.2051**	**0.2024**	**0.1984**	**0.1589**	**0.2722**	**0.2713**	**0.2699**	**0.2648**	**0.2536**	**0.1976**
	DATC	0.2954	0.2748	0.2631	0.2596	0.2541	0.2381	0.4939	0.4332	0.3651	0.3215	0.3037	0.2487
	NTC	0.3702	0.3362	0.3243	0.3115	0.3073	0.2497	0.7042	0.6796	0.6239	0.5781	0.4751	0.2871
	NTM	1.2214	0.8818	0.7633	0.6686	0.6178	0.3758	1.4547	1.3646	1.3266	1.2003	1.1978	0.5150
	NTF	0.7529	0.4147	0.3637	0.3486	0.3353	0.2639	0.9061	0.7783	0.7189	0.5958	0.5125	0.3208
	CoSTCo	0.3857	0.3739	0.3381	0.3224	0.3145	0.2971	0.8588	0.7826	0.7291	0.7241	0.7042	0.6851
NRMSE	TM-LLM	**0.2367**	**0.2358**	**0.2320**	**0.2276**	**0.2247**	**0.1844**	**0.2393**	**0.2294**	**0.2290**	**0.2187**	**0.2138**	**0.1647**
	DATC	0.3453	0.3312	0.3226	0.3202	0.3178	0.3076	0.3818	0.3415	0.2979	0.2547	0.2368	0.1742
	NTC	0.4388	0.4072	0.3951	0.3858	0.3766	0.3436	0.6412	0.6266	0.5888	0.5450	0.4310	0.2319
	NTM	0.9720	0.7573	0.7234	0.6613	0.6197	0.4473	0.9230	0.9057	0.8877	0.7854	0.7629	0.3976
	NTF	0.6856	0.4201	0.3882	0.3848	0.3723	0.3667	0.7110	0.6621	0.5838	0.4959	0.4128	0.2728
	CoSTCo	0.4081	0.3929	0.3736	0.3666	0.3600	0.3579	0.7215	0.6579	0.6252	0.6037	0.5927	0.5827
KL	TM-LLM	**0.0521**	**0.0488**	**0.0465**	**0.0463**	**0.0449**	**0.0329**	**0.0960**	**0.0948**	**0.0938**	**0.0938**	**0.0823**	**0.0626**
	DATC	0.0994	0.0881	0.0794	0.0767	0.0736	0.0688	0.4495	0.3338	0.2522	0.2327	0.2103	0.2076
	NTC	0.1578	0.1326	0.1202	0.1157	0.1095	0.0981	1.2776	1.0643	0.8378	0.8306	0.4765	0.3472
	NTM	2.5215	0.9318	0.8276	0.7838	0.6946	0.2604	2.0707	1.9729	1.8230	1.7226	1.5882	0.6466
	NTF	0.4744	0.2114	0.1635	0.1591	0.1228	0.0805	1.1552	0.7186	0.6736	0.6624	0.5734	0.5242
	CoSTCo	0.3566	0.4370	0.3305	0.2851	0.2461	0.1757	5.2273	3.7190	3.4680	3.3872	3.0307	2.7457
NDCG @10	TM-LLM	**0.9757**	**0.9762**	**0.9766**	**0.9774**	**0.9777**	**0.9874**	**0.9877**	**0.9875**	**0.9874**	**0.9872**	**0.9875**	**0.9907**
	DATC	0.9618	0.9642	0.9659	0.9660	0.9666	0.9688	0.9453	0.9611	0.9682	0.9733	0.9812	0.9894
	NTC	0.9345	0.9403	0.9447	0.9452	0.9465	0.9509	0.7291	0.7594	0.7842	0.8207	0.8969	0.9546
	NTM	0.1920	0.5957	0.6065	0.7288	0.7906	0.8091	0.2558	0.2955	0.3807	0.6294	0.6602	0.7417
	NTF	0.6909	0.9538	0.9561	0.9568	0.9580	0.9497	0.7498	0.7531	0.8123	0.8796	0.9263	0.9649
	CoSTCo	0.9522	0.9537	0.9547	0.9551	0.9555	0.9564	0.7663	0.7882	0.7993	0.7961	0.7996	0.8166



(a) Abilene



(b) GÉANT

Fig. 4. Results of ablation study.

2) Ablation study: To verify the effectiveness of the distinctive designs in TM-LLM, we implement three variants of TM-LLM, denoted as TM-LLM w/o AL, TM-LLM w/o LLM and TM-LLM w/o Prompt. TM-LLM w/o AL performs imputation directly using LLM, without the pre-imputation with adversarial learning. TM-LLM w/o LLM performs the imputation solely using the adversarial learning, without LLM. TM-LLM w/o Prompt follows the design of two-stage imputation, but does not incorporate the crafted prompt template for LLM.

Fig. 4 reports the experimental results of the ablation study at 10% and 50% sampling rates, representing low and high sampling rates, respectively. From the comparison, we can draw the following conclusions: 1) The original form of TM-LLM outperforms the three variants across all three metrics on the two datasets, validating the effectiveness of our design. 2) TM-LLM w/o AL presents a remarkable performance drop in lower-sampling-rate scenarios (i.e., 10% sampling rate), indicating the pre-imputation module plays a significant role in enabling the LLMs to better understand the TM pattern when the data in TM is severely missing. 3) TM-LLM w/o LLM shows a more significant performance drop in higher-sampling-rate scenario (i.e., 50% sampling rate). This indicates LLMs can achieve a deeper understanding of the imputation task, thereby substantially enhancing the imputation performance beyond that of former deep learning models. 4) TM-LLM w/o Prompt undergoes a performance reduction across all scenarios, indicating the crafted prompts play a crucial role in eliciting the LLM's ability to understand the complex features of TM data.

V. CONCLUSION

In this paper, we introduce TM-LLM, the first LLM-based approach for network TM imputation. By leveraging LLMs' pre-trained knowledge and zero-shot reasoning capabilities, TM-LLM significantly enhances the imputation accuracy over the state-of-the art. We address challenges posed by skewed

TM distributions and extensive missing data through a two-stage strategy: pre-imputation via adversarial learning, followed by comprehensive imputation using LLMs. With crafted input embeddings and task-specific prompts, the lightweight fine-tuning ensures the LLMs adapt effectively to the imputation task. Experiments on two real-world datasets show TM-LLM significantly outperforms the state-of-the-art in terms of both accuracy and measurement costs.

ACKNOWLEDGMENT

This research is supported by the Key Laboratory of Knowledge Engineering with Big Data (the Ministry of Education of China), under grant number NO. BigKEOpen2025-04. It is supported by the Fundamental Research Funds for the Central Universities of China (PA2024GDSK0095). It is supported by the National Natural Science Foundation of China (NSFC) under the grant No. 62372149 and No. U23A20303. It is also supported by the China Scholarship Council (CSC).

REFERENCES

[1] Y. Qiao, K. Wu, and X. Yuan, "Autotomo: Learning-based traffic estimator incorporating network tomography," *IEEE/ACM Transactions on Networking*, 2024.

[2] Y. Qiao, X. Yuan, and K. Wu, "Routing-oblivious network tomography with flow-based generative model," in *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*. IEEE, 2024, pp. 2139–2148.

[3] X. Yuan, Y. Qiao, Z. Wei, Z. Zhang, M. Li, P. Zhao, R. Hu, and W. Li, "Diffusion models meet network management: Improving traffic matrix analysis with diffusion-based approach," *IEEE Transactions on Network and Service Management*, 2025.

[4] M. Roughan, Y. Zhang, W. Willinger, and L. Qiu, "Spatio-temporal compressive sensing and internet traffic matrices (extended version)," *IEEE/ACM Transactions on Networking*, vol. 20, no. 3, pp. 662–676, 2011.

[5] S. E. Leurgans, R. T. Ross, and R. B. Abel, "A decomposition for three-way arrays," *SIAM Journal on Matrix Analysis and Applications*, vol. 14, no. 4, pp. 1064–1083, 1993.

[6] P. Tune and M. Roughan, "Spatiotemporal traffic matrix synthesis," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 2015, pp. 579–592.

[7] K. Xie, Y. Ouyang, X. Wang, G. Xie, K. Li, W. Liang, J. Cao, and J. Wen, "Deep adversarial tensor completion for accurate network traffic measurement," *IEEE/ACM Transactions on Networking*, vol. 31, no. 5, pp. 2101–2116, 2023.

[8] D. Jiang, Z. Zhao, Z. Xu, C. Yao, and H. Xu, "How to reconstruct end-to-end traffic based on time-frequency analysis and artificial neural network," *AEU-International Journal of Electronics and Communications*, vol. 68, no. 10, pp. 915–925, 2014.

[9] M. Jin, S. Wang, L. Ma, Z. Chu, J. Y. Zhang, X. Shi, P.-Y. Chen, Y. Liang, Y.-F. Li, S. Pan *et al.*, "Time-llm: Time series forecasting by reprogramming large language models," *arXiv preprint arXiv:2310.01728*, 2023.

[10] Y. Wang, Z. Gu, S. Zhang, S. Zheng, T. Wang, T. Li, H. Feng, and Y. Xiao, "Llm-gan: Construct generative adversarial network through large language models for explainable fake news detection," *arXiv preprint arXiv:2409.01787*, 2024.

[11] M. Jin, Y. Zhang, W. Chen, K. Zhang, Y. Liang, B. Yang, J. Wang, S. Pan, and Q. Wen, "Position: What can large language models tell us about time series analysis," in *Forty-first International Conference on Machine Learning*, 2024.

[12] N. Gruver, M. Finzi, S. Qiu, and A. G. Wilson, "Large language models are zero-shot time series forecasters," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[13] H. Tang, C. Zhang, M. Jin, Q. Yu, Z. Wang, X. Jin, Y. Zhang, and M. Du, "Time series forecasting with llms: Understanding and enhancing model capabilities," *arXiv preprint arXiv:2402.10835*, 2024.

[14] M. E. Kilmer, L. Horesh, H. Avron, and E. Newman, "Tensor-tensor algebra for optimal representation and compression of multiway data," *Proceedings of the National Academy of Sciences*, vol. 118, no. 28, p. e2015851118, 2021.

[15] K. Xie, X. Wang, X. Wang, Y. Chen, G. Xie, Y. Ouyang, J. Wen, J. Cao, and D. Zhang, "Accurate recovery of missing network measurement data with localized tensor completion," *IEEE/ACM Transactions on Networking*, vol. 27, no. 6, pp. 2222–2235, 2019.

[16] J. Tian, K. Xie, X. Wang, J. Wen, G. Xie, J. Cao, W. Liang, and K. Li, "Reducing network distance measurement overhead: A tensor completion solution with a new minimum sampling bound," *IEEE/ACM Transactions on Networking*, 2024.

[17] K. Xie, H. Lu, X. Wang, G. Xie, Y. Ding, D. Xie, J. Wen, and D. Zhang, "Neural tensor completion for accurate network monitoring," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 1688–1697.

[18] Y. Li, W. Liang, K. Xie, D. Zhang, S. Xie, and K. Li, "Lightnestle: quick and accurate neural sequential tensor completion via meta learning," in *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE, 2023, pp. 1–10.

[19] C. Liu, K. Xie, T. Wu, C. Ma, and T. Ma, "Distributed neural tensor completion for network monitoring data recovery," *Information Sciences*, vol. 662, p. 120259, 2024.

[20] T. Zhou, P. Niu, L. Sun, R. Jin *et al.*, "One fits all: Power general time series analysis by pretrained lm," *Advances in neural information processing systems*, vol. 36, pp. 43 322–43 355, 2023.

[21] C. Chang, W.-C. Peng, and T.-F. Chen, "Llm4ts: Two-stage fine-tuning for time-series forecasting with pre-trained llms," *arXiv preprint arXiv:2308.08469*, 2023.

[22] X. Yu, Z. Chen, Y. Ling, S. Dong, Z. Liu, and Y. Lu, "Temporal data meets llm–explainable financial time series forecasting," *arXiv preprint arXiv:2306.11025*, 2023.

[23] A. Elhafsi, R. Sinha, C. Agia, E. Schmerling, I. A. Nesnas, and M. Pavone, "Semantic anomaly detection with large language models," *Autonomous Robots*, vol. 47, no. 8, pp. 1035–1055, 2023.

[24] L. Zanella, W. Menapace, M. Mancini, Y. Wang, and E. Ricci, "Harnessing large language models for training-free video anomaly detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 18 527–18 536.

[25] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi *et al.*, "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," *arXiv preprint arXiv:2501.12948*, 2025.

[26] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan *et al.*, "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.

[27] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[28] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg, "Fast accurate computation of large-scale ip traffic matrices from link loads," *ACM SIGMETRICS Performance Evaluation Review*, vol. 31, no. 1, pp. 206–217, 2003.

[29] S. Uhlig, B. Quoitin, J. Lepropre, and S. Balon, "Providing public intradomain traffic matrices to the research community," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, pp. 83–86, 2006.

[30] K. Xie, X. Li, X. Wang, G. Xie, J. Wen, and D. Zhang, "Graph based tensor recovery for accurate internet anomaly detection," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 1502–1510.

[31] H. Chen and J. Li, "Neural tensor model for learning multi-aspect factors in recommender systems," in *International Joint Conference on Artificial Intelligence (IJCAI)*, vol. 2020, 2020.

[32] X. Wu, B. Shi, Y. Dong, C. Huang, and N. V. Chawla, "Neural tensor factorization for temporal interaction learning," in *Proceedings of the Twelfth ACM international conference on web search and data mining*, 2019, pp. 537–545.

[33] H. Liu, Y. Li, M. Tsang, and Y. Liu, "Costco: A neural tensor completion model for sparse tensors," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 324–334.