Strada-LLM: Graph LLM for traffic prediction

Seyed Mohamad Moghadas, Yangxintong Lyu, Bruno Cornelis
Alexandre Alahi *Member, IEEE*, Adrian Munteanu *Member, IEEE*

***Abstract*—Traffic prediction is a vital component of intelligent transportation systems. By reasoning about traffic patterns in both the spatial and temporal dimensions, accurate and interpretable predictions can be provided. A considerable challenge in traffic prediction lies in handling the diverse data distributions caused by vastly different traffic conditions occurring at different locations. LLMs have been a dominant solution due to their remarkable capacity to adapt to new datasets with very few labeled data samples, i.e., few-shot adaptability. However, existing forecasting techniques mainly focus on extracting local graph information and forming a text-like prompt, leaving LLM-based traffic prediction an open problem. This work presents a probabilistic LLM for traffic forecasting with three highlights. We propose a graph-aware LLM for traffic prediction that considers proximal traffic information. Specifically, by considering the traffic of neighboring nodes as covariates, our model outperforms the corresponding time-series LLM. Furthermore, we adopt a lightweight approach for efficient domain adaptation when facing new data distributions in few-shot fashion. The comparative experiment demonstrates the proposed method outperforms the state-of-the-art LLM-based methods and the traditional GNN-based supervised approaches. Furthermore, Strada-LLM can be easily adapted to different LLM backbones without a noticeable performance drop.**

***Index Terms*—spatio-temporal transformers, graph neural networks, traffic prediction, LLM.**

I. Introduction

WITH the development of Intelligent Transportation Systems, traffic forecasting has received increasing attention. It is a key component of advanced traffic management systems and is crucial in traffic planning, traffic management, and traffic control. Traffic forecasting involves analyzing traffic conditions on roads, including flow, speed, and density, mining traffic patterns, and predicting traffic trends [1]. This capability provides a scientific foundation for any traffic management department to anticipate and mitigate congestion, implement preemptive vehicle restrictions, and enable urban tourists to select safer and more efficient travel routes.

However, traffic forecasting is a challenging task due to complex spatial and temporal dependencies:

1) Spatial Dependency: The variation in traffic volume is influenced by the topological structure of the urban road network. In particular, traffic conditions at upstream roads impact downstream roads through transfer effects, which refers to the impact that a change in one section of a traffic network (e.g., a closure of a road, a change in signal timing, or new infrastructure) has on other parts of the network. On the other hand, downstream roads affect upstream roads through feedback effects, which occur when the changes in traffic patterns resulting from transfer effects influence the original cause of the change, creating a loop of cause and effect.
2) Temporal Dependency: Traffic volume changes over time, exhibiting periodicity and trends, which are often influenced by factors such as holidays, working hours, and other social events.

Seyed Mohamad Moghadas, Yangxintong Lyu, Bruno Cornelis, Adrian Munteanu are with the Department of Electronics and Informatics, Vrije Universiteit Brussel, Pleinlaan 2, B-1050 Brussels, Belgium.

Seyed Mohamad Moghadas, Yangxintong Lyu, Adrian Munteanu are with the IMEC, Kapeldreef 75, B-3001 Leuven, Belgium.

Bruno Cornelis is with Macq, Rue de l'Aéronef 2, 1140, Brussels, Belgium.

Alexandre Alahi is with the VITA Laboratory, EPFL, 1015 Lausanne, Switzerland.

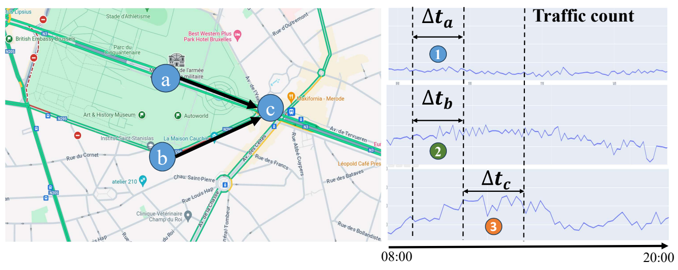

Fig. 1. An example of the spatio-temporal dependency between the traffic flows for different traffic states ①,②,③. Nodes (a), (b), and (c) are placed in a crowdedrea of Brussels. The corresponding traffic patterns between 8:00 am and 8:00 pm are illustrated. The short-term traffic flow similarity is altered by the significant impact of neighboring roads, resulting in varying flows.

As illustrated in Figure 1, the strong influence among adjacent roads caused by the traffic flow alongside the edges changes the short-term state ① to state ② and state ③. Indeed, node *(a)* has a steady traffic count trend, and node *(b)* has short spikes in volume ②, so the spatio-temporal correlation caused state ③ in node *(c)* after a couple of hours. Furthermore, traffic data sources exhibit a variety of data distributions, posing a significant challenge for the domain in finding generalizable models.

In recent years, the emergence of Large Language Models (LLM) have revolutionized various fields due to the generalization capabilities of extracting efficient features from different modalities [2], [3]. Research has been conducted in multiple domains, such as computer vision and natural language processing. Specifically, in [4], a unified representation space is proposed, namely shared vocabulary, across varied data sources such that the model can learn domain-invariant features. This characteristic is particularly appealing for traffic prediction, where data distributions can vary significantly across different regions and times. In [5], [6], [7], the authors have tried to address this problem by casting the graph (and/or the corresponding time-series) to a prompt input. However, this branch of methods, in terms of predictability, is sub-optimal because traditional time series forecasting models use numerical inputs and outputs, whereas prompt-based methods transform these into text prompts, which may not be as effective for capturing the nuances of time series trends [8]. It will lead to a performance gap compared to classical GNNs [9], [10]. Moreover, the complexity and scalability of the prompt-based approaches are unexplored. For instance, in Spatio-Temporal Graphs(STG) with 1 million nodes, feeding the nodes to an LLM with limited context length is a challenging operation. Consequently, the prompting operation loses the global topological information. In [11], the structural information of the underlying graph is encoded as prompts for molecular analysis, but the effectiveness of this approach in urban contexts remains uncertain.

Parameter-Efficient Fine-Tuning (PEFT) has emerged as a favored method for integrating domain-specific expertise into Large Language Models (LLMs), particularly in the realms of language and vision processing [12]. This technique involves modifying a limited set of model parameters—either by targeting existing weights or introducing additional ones—while preserving the bulk of the model's architecture. PEFT enables the retention of valuable pre-trained feature representations while concurrently tailoring the model to particular domains through targeted adjustments. This approach effectively strikes a balance between maintaining general knowledge and incorporating specialized insights. Among the popular PEFT strategies are LoRA [13] and prompt-tuning [14]. However, the optimal selection of an efficient method for an expert urban LLM, along with a comparative analysis of these techniques in this specific context, remains unexplored. Consequently, the question of how best to apply LLMs to graph-based traffic prediction using these methods is still open for investigation.

To address these issues, we propose a new traffic forecasting method called Strada-LLM, a graph-aware LLM for traffic prediction. To the best of our knowledge, our model is the first attempt to propose a specialized parameter-efficient LLM model in traffic forecasting. Our contributions are five-fold:

1) Strada-LLM is the first probabilistic LLM, which is compliant with classical LLMs and specializes in spatio-temporal traffic forecasting. Specifically, Strada-LLM has the capability to predict traffic, even when faced with a data distribution that differs from its training data, particularly in the few-shot setting.
2) Strada-LLM is graph-aware, taking spatial dependencies into account by encoding the network graph implicitly, especially without any prompting.
3) Strada-LLM adopts a lightweight approach to perform domain adaptation. Specifically, adopting low-ranked approaches showcases its effectiveness in adapting to new datasets.
4) We provide a comprehensive evaluation and superior accuracy. We evaluate our approach using numerous real-world traffic datasets. Our results show a significant reduction in prediction error, ranging from approximately 5% to 18%, compared to baseline methods. This demonstrates the superiority of our graph-aware LLM model in traffic forecasting. We delve into the comparison between low-rank adaptation and prompt-tuning as two cogent approaches for LLM fine-tuning.
5) Strada-LLM is flexible and can be used with different known LLMs.

The paper is organized as follows. Section II examines pertinent studies related to traffic forecasting. Section III introduces the details of our method. In section IV, we evaluate the predictive performance of the Strada-LLM using real-world traffic datasets.

II. Related Work

The related approaches can be categorized into two main groups: those specifically designed for traffic prediction and those developed for broader timeseries forecasting. The latter category includes methods for both univariate and multivariate time series analysis, which have been adapted to handle traffic-related variables.

A. Statistical models

This class of models served as a fundamental pillar in traffic forecasting for many years, progressively evolving to tackle sophisticated forecasting challenges. Statistical models, such as ARIMA (Autoregressive Integrated Moving Average) [15] established the groundwork by utilizing auto-correlation to predict future outcomes. Subsequently, ETS (Error, Trend, Seasonality) models [16], [17] break down a time series into essential elements such as trends and seasonal variations, thus enabling more detailed forecasts. Theta models [18], [19] marked another notable progression by partially addressing non-linear behavior. Moreover, [18], [20] often necessitates significant manual adjustment and specialized knowledge to choose suitable models and parameters for particular forecasting problems.

B. Neural Traffic forecasting

Deep learning-based traffic forecasting is a rapidly developing research area [21]. Various architectures have been developed, including CNN-based [22], [10], RNN-based [23], and LSTM-based models[24], [25]. Inspired by the vanilla transformer mechanism, introduced in [26], PatchTST [27] highlights the importance of extracting local semantic information from time series data, which is often overlooked in previous models that use point-wise tokens, then re-process the output of a transformer for a point forecast or a probabilistic forecast. On the other hand, various other works propose alternative strategies to vanilla attention and build on the transformer architecture, leading to better models tailored for time series forecasting [28], [29], [30]. Specifically, models such as [31] propose spatio-temporal attention, others leverage trend-aware decomposition in the definition of the attention layer [32], [33], [34], [35], [3], [36]. On the other hand, there are copula-based transformer models in the literature [37], better at capturing volatilities, which are defined as the degree of variation in the time-series value over time. We selected transformer-based models due to their powerful ability to extract features and their effectiveness in few-shot learning scenarios. Specifically, we adopt Mistral [38] as our backbone model.

C. LLMs

LLM models are an emerging paradigm within deep learning [39]. A key factor contributing to their popularity is the utilization of spatio-temporal attention [40], as highlighted by [31]. Many such models [41], [12] have showcased the power of robust feature extraction on a large scale. However, fine-tuning an LLM is time and energy-consuming, leading to the proposal of efficient methods [13], [42] with differing tuned parameters. LLMs have also been extended to scientific domains such as protein design [43], [44]. However, their application in urban traffic forecasting is still being investigated. Notably, several LLMs have been developed for traffic forecasting, including ST-LLM [45], which partially freezes the transformer block deteriorating the performance on a large scale. Moreover, Time-LLM [46], LLM4TS [47], and UniTime [48] cast the graph to textual prompt, that is, encode nodes, edges, attributes, etc., and forecast based on the prompts. Moreover, prompt-tuning emerged as a prominent technique for adapting large models [49]. FlashST [14] extends the prompt-tuning to spatio-temporal data. They adjust to unseen datasets by applying prompt-tuning in the context of spatio-temporal data. This category of models neglects the graph topology which hampers the prediction efficiency of the LLM [50]. Furthermore, data diversity has formed an additional challenge for LLMs in few-shot learning on novel datasets. [51]. The main goal of our work is to apply the LLM approach to traffic data forecasting and to investigate to what extent LLMs can adapt across a wide range of traffic datasets.

III. Methodology

A. Problem Definition

In this paper, we aim to predict traffic conditions at a specific timestamp T' based on historical traffic data over period T. Particularly, the traffic metric broadly refers to any traffic speed, flow, or density.

Before further elaborating on our method, we define the following notations. In Strada-LLM, a road network is represented as a graph G, consisting of a set of nodes V and a set of edges E. Each node presents a traffic measuring sensor, and an edge connects two nodes if they are geographically adjacent. That is, $G = (V, E, A)$ where V is a set of N nodes, E is a set of edges and $A \in \mathbb{R}^{N\times N}$ is the corresponding adjacency matrix. We assume that the topology of the traffic graph G is static. At each timestamp t, the graph G is associated with a dynamic feature matrix $\boldsymbol{X}_t \in \mathbb{R}^{N\times F}$, where F is the number of traffic metrics.

In this paper, we formulate traffic forecasting as a discrete multi-variate point prediction problem. In particular, Strada-LLM takes the T past observations $\boldsymbol{X} = [\boldsymbol{X}_{t_1}, \boldsymbol{X}_{t_2}, ..., \boldsymbol{X}_{t_T}] \in \mathbb{R}^{T\times N\times F}$ from G as input and predicts the traffic for horizon T', where $\hat{\boldsymbol{X}} = \left[\hat{\boldsymbol{X}}_{t_{T+1}}, \hat{\boldsymbol{X}}_{t_{T+2}}, ..., \hat{\boldsymbol{X}}_{t_{T+T'}}\right]$.

B. Proposed Method

Strada-LLM consists of a Hierarchical Feature Extractor (HFE), an LLM-based backbone, and a T-student distribution head. The distribution head includes three learnable parameters related to this distribution: degrees of freedom, mean, and scale, with suitable non-linearities to ensure the relevant parameters remain positive, as shown in Figure 2. In the HFE module, a sub-graph extractor and a global graph feature extractor hierarchically learn features while the LLM-based block maps the graph features into the latent spatio-temporal feature space to be learned by the distribution head. In the following sections, we will elaborate on each component.

1) Hierarchical Feature Extractor:

- **Subgraph extractor:** To enhance the LLM's understanding of graph structures while accommodating its limited context length, we tokenize the spatio-temporal traffic signal regarding the existing subgraphs of the road network. Overall, the corresponding traffic signals for neighboring nodes will be aggregated together to be tokenized in the further steps. This block is responsible for extracting k-hop subgraphs from the input graph. For a node $v \in G$, the k-hop operator [52] is defined as:

$$\mathcal{N}_k(v) = \{u \in V | d(u, v) \leq k\}, \tag{1}$$

  where $d(\cdot, \cdot)$ is the hop-based distance function between two nodes. By doing so, each node is equipped with a sub-graph of G, including local topological information. As a result, for each node v, we concatenate the features of $\mathcal{N}_k(v)$, leading to an $N \times T \times (M \times F)$ feature map, where M denotes $|\mathcal{N}_k(v)|$. Similarly, Subgraph-1-WL [53] and MixHop [54] can also be used as the k-hop operator, but it is beyond the scope of this research.
- **Global Graph Feature Extractor:** While prompt-based networks are widely used in the existing methods [44], they lose the global structure information [14]. In order to keep the global graph structure information, we compute Laplacian embeddings [55]. The Laplacian operation leverages the eigenvectors of the complete graph and creates unique positional encodings [26] for each node. First, we define the graph Laplacian matrix $L = D - A$ where $D = diag(d_1, \ldots, d_N)$ is the degree matrix. We then compute the eigen decomposition of the normalized Laplacian as follows $L_{\text{norm}} = D^{-1/2}LD^{-1/2} = U\Lambda U^T$.
- **Lag Features** The tokenization strategy employed by Strada-LLM revolves around the creation of lagged features derived from historical traffic data. These features are constructed using predetermined sets of lag indices, such as quarterly, monthly, etc., also referred to as second-level frequencies, carefully chosen to capture various temporal patterns. Mathematically, this lagging operation can be expressed as a mapping $\boldsymbol{x}_t \longmapsto \boldsymbol{\lambda_t} \in \mathbb{R}^{|\mathcal{L}|\times F}$, where $\mathcal{L} = \{1, ..., L\}$ represents the set of lag indices. In this formulation, $\lambda_t[j] = x_{t-\mathcal{L}[j]}$, meaning that $\lambda_t[j]$ corresponds to the value of x at j time steps before t, as specified by the j-th element of $\mathcal{L}$. To put it another way, $\lambda_t[j]$ represents the $\mathcal{L}[j]$-th lagged feature when considering the time series $\boldsymbol{x}$ in reverse chronological order. This approach allows for a comprehensive encoding of temporal dependencies across various time scales. Therefore, to generate lag features for a specific

context-length window $x_{1:C}$, where C is the length of the context window, it is necessary to consider an expanded window that includes L additional historical data points. Besides the lagged features, we incorporate date-time features covering all temporal features in our dataset, including second-of-minute, hour-of-day, and so forth, up to quarter-of-year, based on the time index t. It is important to note that the main purpose of these date-time features is to enrich the information set. In any time series, all but one of the date-time features will stay the same from one time step to another, allowing the model to intuitively understand the time series frequency. We use a total of C date-time features, making the size of each token $|\mathcal{L}| + C$.

2) LLM-based-Backbone: Our backbone architecture is inspired by Mistral [38], a decoder-only transformer architecture. The extracted tokens by the sub-graph extractor are transformed by a shared linear projection layer that maps the features to the hidden dimension of the attention module. Inspired by [56], Strada-LLM includes pre-normalization by utilizing the RMSNorm [57] and Rotary Positional Encoding (RoPE) [58] for the representation of query and key blocks at each attention layer, similar to the approach described for Mistral [38].

After passing through the causally masked transformer layers, the proposed method incorporates Flash-Attention-2 [59] to predict the parameters of the forecast distribution for the next timestamp, denoted as ϕ. These parameters are generated by a parametric distribution head, as shown in Figure 2. The objective is to minimize the negative log-likelihood of the predicted distribution across all prediction times.

At the inference stage, given a time series with the size of at least $L + C$, a feature vector can be formed and supplied to the model to determine the distribution of the subsequent timestamps. In this fashion, we can obtain many simulated future trajectories up to our chosen prediction horizon T' via greedy auto-regressive decoding [60]. In doing so, uncertainty intervals and confidence scores could also be valuable for downstream decision-making tasks.

C. Domain Adaptation

In this section, we describe the techniques we adopt to adapt the Strada-LLM model to a new dataset, which might contain a graph of a new city. One of the important aspects of such a model is that it should be domain-agnostic. In the rapidly evolving landscape of deep learning, domain adaptation has emerged as a crucial technique for enhancing the generalizability of models across different but related domains. More specifically, domain adaptation focuses on the ability of a model trained in one domain (source) to adapt and perform well in another domain (target) with potentially different distributions. A prominent approach within this realm is low-rank adaptation [13], which aims to refine pre-trained models by adapting their parameters in a computationally efficient manner; indeed, a nearly linear operation will replace the $\mathcal{O}(n^2)$ operation. This technique is particularly beneficial when working with LLMs, which, despite their impressive performance and versatility, pose significant challenges regarding computational resources and training time. In this paper, we adapt to the new distribution by tuning the distribution head, query, key, and value attention blocks.

1) Low-Rank Matrix Adaptation: By leveraging domain adaptation strategies and Low-Rank Matrix Adaptation (LoRA) for fine-tuning,researchers and practitioners can effectively bridge the gap between domains, ensuring that powerful deep learning models maintain their efficacy across diverse applications [61]. Effectiveness of LoRA in the NLP domain has already been proven [13]. One of the purposes of this research is a demonstration of LoRA in the traffic forecasting problem. As illustrated in Figure III-C, we apply LoRA for the fine-tuning process. The query, key, and value matrices will be fine-tuned in a low-rank approximation manner. Specifically, according to Figure III-C, suppose the query, key, and value tensors are computed as follows:

$$q = W_q h, k = W_k h, v = W_v h, \tag{2}$$

where h is the output of the last layer of the LLM backbone with dimension $\mathbb{R}^{d \times k}$. W_q, W_k and W_v are the query, key, and the projection weight matrices respectively. The low-rank matrices for fine-tuning the forward step can be defined as:

$$\begin{aligned} q &= W_q h + \Delta W_q h = W_q h + B_q A_q h \\ k &= W_k h + \Delta W_k h = W_k h + B_k A_k h \\ v &= W_v h + \Delta W_v h = W_v h + B_v A_v h, \end{aligned} \tag{3}$$

where h is the output of the previous layer, the low-ranked approximated update matrices B_q, B_k, B_v are of dimensions $\mathbb{R}^{d \times r}$, A_q, A_k, A_v are of dimensions $\mathbb{R}^{r \times k}$, and where the rank hyperparameter $r \ll min(d, k)$. By introducing these matrices, the online inference latency reduction is achieved [13]. For the initialization of the matrices A we choose Gaussian initialization while the B matrices are initialized to zero [13]. Following this approach enables us to perform domain adaptation for different datasets consisting of diverse distributions.

TABLE I
THE COMPARISON BETWEEN PEFT AND PROMPT-TUNING APPROACHES

	# trainable params	Memory	Performance	Flexibility to different architectures
Prompt-Tuning [14]	+			
LoRA [13]		+	+	+

Table I presents a comparative analysis between Parameter-Efficient Fine-Tuning (PEFT) methods and prompt-tuning approaches, providing insights into their strengths and trade-offs. The table clearly highlights that prompt-tuning offers an advantage in terms of the number of trainable parameters, making it a lightweight alternative for model adaptation. However, it appears that parameter-efficient methods such as LoRA outperform prompt-tuning when considering memory efficiency, performance, and flexibility across different architectures. LoRA, in particular, demonstrates a clear edge in terms of both adaptability and overall resource management, suggesting its suitability for more complex and varied machine learning environments. These observations indicate that, while prompt-tuning may be appealing for its simplicity and parameter efficiency, methods like LoRA are likely more robust for scenarios requiring higher performance and architectural versatility.

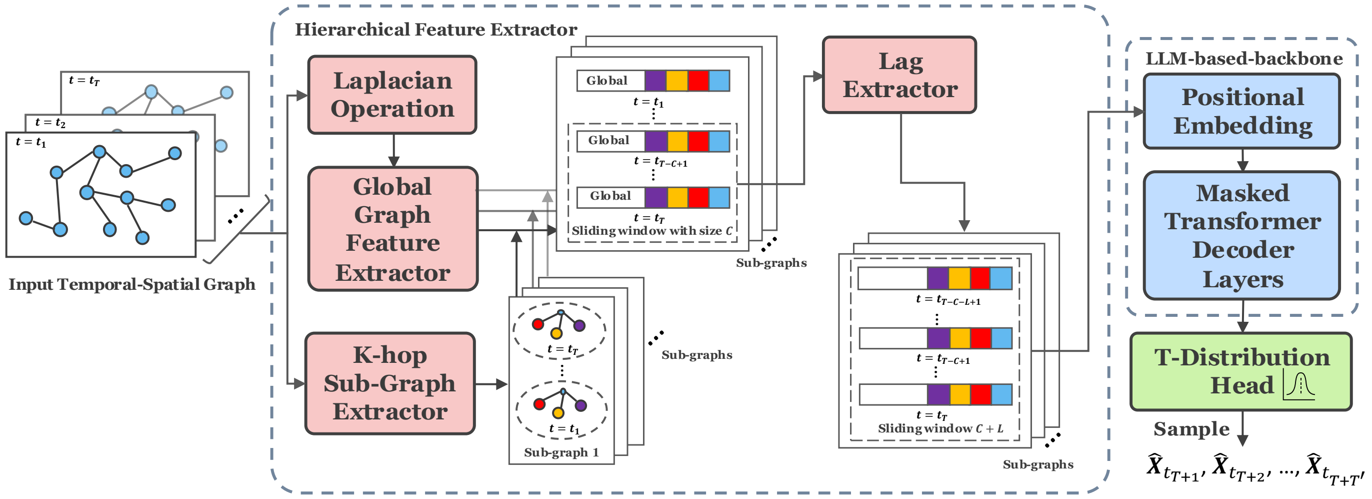
Hierarchical Feature Extractor
Laplacian Operation
Global Graph Feature Extractor
K-hop Sub-Graph Extractor
Lag Extractor
Input Temporal-Spatial Graph
Global
Sliding window with size C
Sub-graphs
Sub-graph 1
Sliding window C + L
LLM-based-backbone
Positional Embedding
Masked Transformer Decoder Layers
T-Distribution Head
Sample

Fig. 2. The proposed Strada-LLM architecture. The novelty lies in the K-hop sub-graph extractor module, which takes lag features in both the spatial and temporal dimensions. The transformer backbone is inspired by [56].

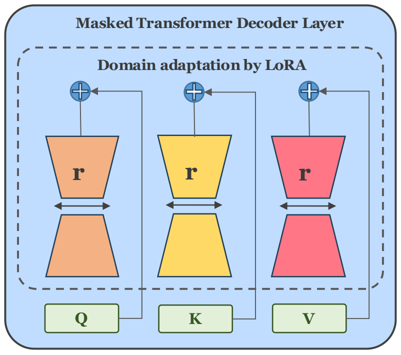

Fig. 3. Application of the LoRA method for domain adaptation. **Frozen** query, key, and value tensors will be fine-tuned by approximating low-rank matrices.

D. Loss Function

The Negative Log-Likelihood (NLL) is a loss function commonly used in probabilistic time-series forecasting. It is calculated as the negative of the log of the probability of the true value given the predicted value. The formula for negative log-likelihood is:

$$\text{NLL} = -\sum_{x\in\mathcal{D}} log(P(y|x)), \tag{4}$$

where y is the true value, x is the input data, $\mathcal{D}$ is the training set, and $P(y|x)$ is the predicted conditional probability of y given x. In contrast to other loss functions, such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE) [14], which measure the difference between the predicted and true values directly, the negative log-likelihood measures the conditional probability of the true value under the model's predictions. The network is fully optimized on the basis of the NLL loss during training. This probabilistic approach makes NLL particularly advantageous for models designed to capture the underlying probability distribution of the data. In the context of few-shot learning for a new geographical area, this characteristic enables the model to learn the intrinsic graph distribution effectively.

IV. EXPERIMENTS

A. Dataset Details

In this section, we assess the Strada-LLM model's predictive capabilities using eight real-world datasets. The description of these datasets is presented in Table II. Most of these datasets pertain to traffic speed. For the simplicity, traffic speed is utilized as the primary traffic metric in our experimental analysis; thus one can presume $F = 1$.

TABLE II
DATASET DESCRIPTION

Dataset	Data Type	Number of Nodes	Data Split Ratio(%)
Pretraining Datasets			
PeMS03 [50]	Traffic Speed	170	70, 20, 10
PeMS04 [50]	Traffic Speed	307	70, 20, 10
PeMS07 [50]	Traffic Speed	883	70, 20, 10
PeMS08 [50]	Traffic Speed	170	70, 20, 10
Domain Adaptation			
METR-LA [50]	Traffic Speed	207	70, 20, 10
PEMS-Bay [50]	Traffic Speed	325	70, 20, 10
PeMS07(M) [50]	Traffic Speed	228	70, 20, 10
Brussels [62]	Traffic Count	207	70, 20, 10

B. Training Strategy

1) Augmentation Techniques: We adopted the time series augmentation techniques $FreqMix$[63] and $FreqMask$ [63] to prevent over-fitting.

2) Hyperparameters: The hyperparameters of the Strada-LLM model mainly include: learning rate, batch size, training epochs, the number of layers, context length, embedding dimension per head, number of heads, rope scaling, and number of parallel samples for greedy decoding. We adopt Adam as the optimizer [64]. In the experiment, we follow the weight decay $1e-8$ and set the learning rate to 0.001 [64], the batch size to 32, and the training epochs to 150 for each dataset. We set $k = 3$ in the sub-graph extractor. Furthermore, based on the fact that Strada-LLM is a probabilistic model, we take samples and compute medians to compare with point-wise models. The number of samples is set to 100. In terms of complexity, our model contains 16 million parameters. The experiments are conducted on 2 Nvidia RTX 4090 GPUs.

C. Baselines

We compare the performance of the Strada-LLM model with the following baseline methods:

- STSGCN [22]: Spatial-Temporal Synchronous Graph Convolutional Layers are used to process the traffic as a spatio-temporal tensor.
- GMAN [23]: The gating fusion mechanism redefines the spatio-temporal attention block.
- MTGNN [65]: Alternating use of graph convolution and temporal convolution modules.
- GTS [25]: The underlying graph structure is learned among multiple time series, and the corresponding time series is simultaneously predicted with DCRNN [1].
- STEP [66]: Adopts the graph wavenet model [10], integrated into the transformer backbone and pre-training scheme.
- FlashST [14]: The data distribution shift is learned by leveraging prompt-tuning.

To compare with the baselines, we adopt MAE, RMSE, and MAPE prediction error [14].

D. LLM Model Results

The long-term traffic prediction accuracy was evaluated across four datasets: PeMS-Bay, METR-LA, PEMS07(M), and Brussels. The last dataset is private. Strada-LLM, as an LLM should be benchmarked jointly for all datasets. So, we report the performance in two manners:

1) **knowledge adaptation**: Like every LLM, we pre-trained Strada-LLM on the datasets PeMS03, PeMS04, PeMS07, and PeMS08 then fine tuned on the PeMS-Bay, METR-LA, PEMS07(M), and Brussels datasets in few-shot fashion; we refer to this approach as *LLM* setup.
2) **fully-supervised**: We trained our proposed model on a single dataset in a fully-supervised fashion (referred to as $Strada - LLM\,[Solo]$ in this section).

The prediction results for the datasets METR-LA and PEMS-Bay are demonstrated in Tables III and IV, correspondingly.

The ability to perform long-term forecasting and being robust to new domains are important qualities for an LLM. As shown in Tables III and IV, the proposed LLM has competitive performance compared to Step [66], which is the corresponding transformer-based fully-supervised method. On the other hand, compared to the best prompt-tuning method FlashST [14], taking the global structure of the graph into account brings forth a definite advantage. Moreover, although Strada-LLM and STEP [66] are both transformer-based models, the introduction of normalization blocks like RMSNorm [57] and Rotary Positional Encoding (RoPE) differentiates them. Additionally, while STEP [66] is an encoder-decoder model, Strada-LLM is a decoder-only model, which makes it optimized in terms of performance for inference [67]. As we can see in Tables III–IV, the Strada-LLM manages to outperform the baseline models on the aforementioned datasets. Specifically, on the PeMS-Bay dataset, our proposed model achieves an RMSE of 3.94 for 1-hour prediction, corresponding to an improvement of approximately 16% compared to the prompt-tuning baseline [14].

E. Domain Adaptation Results

To rigorously assess the efficacy of Strada-LLM compared to prompt-tuning techniques, particularly in few-shot learning scenarios, we conduct a comprehensive evaluation. Our experimental design involves fine-tuning Strada-LLM for a duration equivalent to that used in the FlashST approach, as proposed in [14] on simple universal prompt tuning. The results of this comparative analysis are meticulously documented in Table V, which presents a detailed breakdown of performance metrics for both the Brussels and PeMS07(M) datasets. Upon careful examination of these results, it becomes evident that our Strada-LLM model demonstrates superior performance across the majority of the evaluation metrics when compared to the FlashST method. This improvement in performance serves as a strong indicator of the inherent advantages of our pre-trained, graph-aware LLM in few-shot learning scenarios. The observed superiority can be attributed to Strada-LLM's ability to effectively leverage its pre-existing knowledge of graph structures, allowing it to adapt more efficiently to new domain with limited training examples. These findings not only validate the effectiveness of our approach but also underscore the potential of graph-aware LLM compared to prompt-based forecasters.

To further validate the adaptability of Strada-LLM, we compare the performance of our proposed method with the most relevant transfer learning approaches from the literature, including:

- Full fine-tuning: All of the trainable parameters of the model will be fine-tuned.
- Top^k fine-tuning [13]: Just fine-tune the last k layers.
- LoRA [13]: Using low-rank matrices to fine-tune the model.

There are some other methods like instruction tuning [68], which here are not applicable because of the used modality. Moreover, we intertwine LoRA [13] with our proposed model in order to be adapted to other datasets. We experiment with varying sample percentages of the target dataset, in this case PEMS07(M), to identify the saturation point in the few-shot learning process. The results are shown in Figure 4. The plot shows that training on at least **55%** of the target data in a few-shot manner is sufficient to achieve good performance. This makes Strada-LLM a versatile LLM for urban data forecasting. Based on the superior performance on the Brussels city dataset, presented in Table V, one can conclude that Strada-LLM is robust when applied to a new city with a different data distribution.

TABLE III
TRAFFIC FORECASTING COMPARISON WITH THE STATE-OF-THE-ART METHODS ON THE METR-LA DATASET

	Baselines	Training Time(H)[a]	15 min			30 min			60 min		
			MAE↓	RMSE↓	MAPE(%)↓	MAE↓	RMSE↓	MAPE(%)↓	MAE↓	RMSE↓	MAPE(%)↓
Supervised	STSGCN [22]	-	3.31	7.62	8.06	4.13	9.77	10.29	5.06	11.66	12.91
	GMAN [23]	-	2.80	5.55	7.41	3.12	6.49	8.73	3.44	7.35	10.07
	MTGNN [65]	3	2.69	5.18	6.88	3.05	6.17	8.19	3.49	7.23	9.87
	GTS [25]	6	2.67	5.27	7.21	3.04	6.25	8.41	3.46	7.31	9.98
	STEP[66]	18	2.61	4.98	6.6	2.96	5.97	7.96	3.37	6.99	9.61
LLM	FlashST [14]	0.5	2.84	5.57	7.45	3.19	6.43	9.04	3.60	7.44	10.68
	Strada-LLM(ours)	0.5	**2.73**	**4.95**	**6.54**	**2.99**	**5.84**	**7.87**	**3.43**	**6.73**	**9.47**

[a] The reported numbers were reproduced on our hardware. For the LLM-based models, the entire duration of the few-shot learning process is reported.

TABLE IV
TRAFFIC PREDICTIONS COMPARISON WITH THE STATE-OF-THE-ART METHODS ON PEMS-BAY DATASET

	Baselines	Training Time(H)[a]	15 min			30 min			60 min		
			MAE↓	RMSE↓	MAPE(%)↓	MAE↓	RMSE↓	MAPE(%)↓	MAE↓	RMSE↓	MAPE(%)↓
Supervised	STSGCN [22]	-	1.44	3.01	3.04	1.83	4.18	4.17	2.26	5.21	5.40
	GMAN [23]	-	1.34	2.91	2.86	1.63	3.76	3.68	1.86	4.32	4.37
	MTGNN [65]	3	1.32	2.79	2.77	1.65	3.74	3.69	1.94	4.49	4.53
	GTS [25]	6	1.34	2.83	2.82	1.66	3.78	3.77	1.95	4.43	4.58
	STEP[66]	18	1.26	2.73	2.59	1.55	3.58	3.43	1.79	4.20	4.18
LLM	FlashST [14]	0.5	1.37	2.96	2.88	1.72	3.98	3.89	2.02	4.69	4.79
	Strada-LLM (Ours)	0.5	**1.28**	**2.77**	**2.55**	**1.63**	**3.14**	**3.27**	**1.83**	**3.94**	**4.09**

[a] The reported numbers were reproduced on our hardware. For the LLM-based models, the entire duration of the few-shot learning process is reported.

TABLE V
FEW-SHOT LEARNING EXPERIMENT ON THE PEMS07(M) AND BRUSSELS DATASET(p-VALUE < 0.05)

Baselines	PEMS07(M)									Brussels								
	15 min			30 min			60 min			15 min			30 min			60 min		
	MAE↓	RMSE↓	MAPE(%)↓	MAE↓	RMSE↓	MAPE(%)↓	MAE↓	RMSE↓	MAPE(%)↓	MAE↓	RMSE↓	MAPE(%)↓	MAE↓	RMSE↓	MAPE(%)↓	MAE↓	RMSE↓	MAPE(%)↓
FlashST [14]	**2.37**	5.14	5.93	**2.59**	5.25	6.54	**2.83**	6.81	7.46	4.73	7.54	**20.34**	5.24	8.59	**21.09**	6.06	10.24	**22.14**
Strada-LLM(ours)	2.38	**4.13**	**4.92**	2.60	**4.86**	**6.51**	2.84	**6.43**	**7.41**	**4.53**	**7.16**	20.38	**5.16**	**8.14**	21.15	**5.86**	**10.12**	22.18

In few-shot scenarios, the convergence plateau is particularly crucial because the model is learning from a limited amount of data, making it prone to overfitting or failing to generalize beyond the training set. To demonstrate the experiment, Figure 5 can verify our effectiveness after 35 epochs. As shown in Table III, the few-shot learning time for both models are identical. Additionally, both LLMs outperform Step (black line in Figure 5) confirming the superiority of LLMs compared to mono-dataset models.

Notably, as shown in Figure 4, LLM models (i.e., the red and blue plots) outperform the fully supervised method (the black horizontal line); consequently, it shows the capability of LLMs compared to the traditional supervised models.

F. Perturbation Analysis and Robustness

To evaluate obustness of the Strada-LLM model against noise, we conduct perturbation analysis experiments. We added two types of common random noise to the validation data during the experiment. Random noise obeys the Gaussian distribution $\mathcal{N}(0, \sigma^2)$ where $\sigma \in (0.2, 0.4, 0.8, 1, 2)$. Then, we normalize the values of the noise matrices to be between 0 and 1. The results are shown in Figure 6, where the horizontal axis represents σ, the vertical axis represents the error, and different colors indicate different evaluation metrics. This benchmark is evaluated for the dataset PEMS04 but other datasets follow the same pattern. According to this experiment, our model is robust to Gaussian noise.

G. Ablation Study

1) Domain Adaptation: In this ablation study, we examine the domain adaptation capabilities of Strada-LLM by evaluating its performance when different datasets are used as source and target domains. Specifically, we trained Strada-LLM on one dataset (e.g., PeMS-04) and adapted to another (e.g., METR-LA) to assess its generalization across different types of traffic patterns. Table VI presents the results for the domain adaptation process when pre-trained on a source dataset and adapted to the corresponding target dataset. The results presented in Table VI indicate that there is a performance drop when the model is applied to a different target domain. This table can also address selecting efficient pre-training datasets to gain transferability advance for the targeted LLM. Take

TABLE VI
DOMAIN ADAPTATION COMPARISON FOR DEMONSTRATING SELECTION OF EFFECTIVE SOURCE DATASET

Source	→ Target	Top^k Fine-tuning		LoRA[13]		Fully-Finetune		Fully-Supervised	
		MAE↓	RMSE↓	MAE↓	RMSE↓	MAE↓	RMSE↓	MAE↓	RMSE↓
Metr-LA	Pems-BAY	3.31	5.76	2.64	4.83	2.01	3.89	1.54	3.53
Pems-BAY	Metr-LA	6.12	8.10	5.28	7.06	4.31	6.18	3.08	5.79
Metr-LA	Pems04	22.36	28.72	22.12	27.58	20.41	26.43	19.26	25.47
Pems04	Metr-LA	5.20	6.56	4.83	6.27	4.01	6.13	3.08	5.79
Pems-BAY	Pems04	25.76	30.11	24.71	28.96	23.96	29.83	19.26	25.47
Pems04	Pems-BAY	2.73	5.02	1.96	4.15	1.67	3.67	1.54	3.53

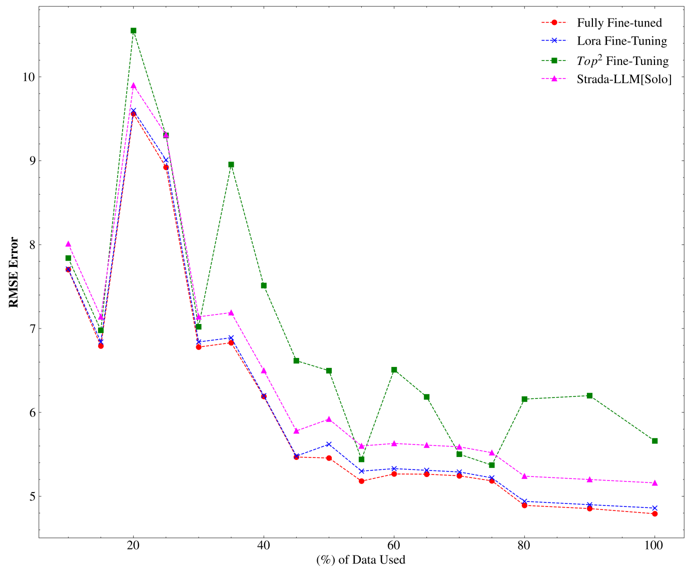

Fig. 4. Domain adaptation plateau for Strada-LLM by applying different approaches on the PEMS07(M) dataset in the few-shot setting. The LLM models(highlighted in red and blue) consistently outperform the fully-supervised model (pink), which demonstrates the added value of an LLM.

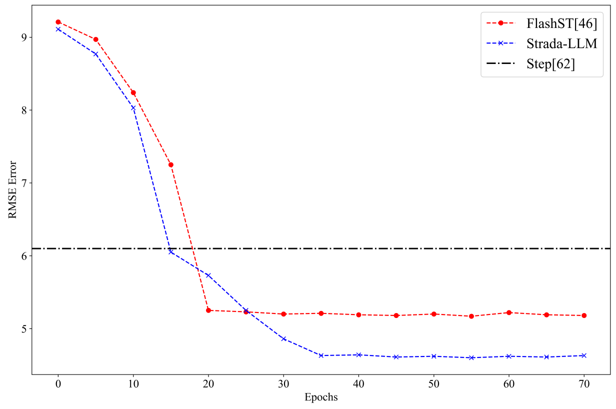

Fig. 5. Epoch convergence plateau comparison on PEMS07(M) dataset.

Pems04 as an example, selecting Metr-LA as the pertaining dataset for LLM ends up the 11.39% superior performance in the adaptation phase compared to Pems-Bay, because of the inherent closer distribution. Therefore, considering **Pems-bay and Metr-LA** as the pertaining datasets will have more distribution coverage compared to Metr-LA and Pems04.

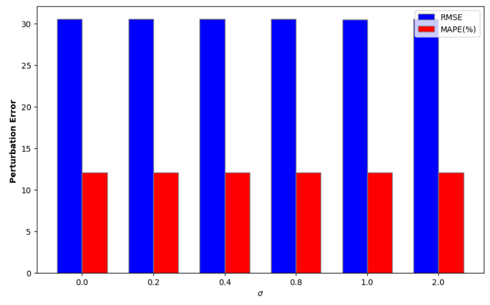

Fig. 6. Evaluation metrics for the perturbation analysis. Different σ values were examined to test the model robustness.

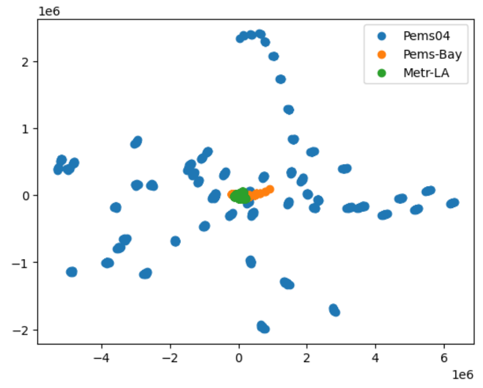

Fig. 7. Latent space embedding of the last layer (before the distribution head) visualized in a 2D space.

2) Model Interpretation: To better understand Strada-LLM, we visualize the latent space of the LLM representation for different datasets. By doing so, the disparity of dataset distributions and the reason for the difficulty of adapting to PEMS04 are revealed. The visualization is shown in Figure 7 which is obtained with T-SNE [69] in 2D dimensions. Our variational posterior distribution $P(\theta \mid X)$ can truly depict the underlying distribution.

3) LoRA: We evaluate the performance of our LoRA adoption with different widths. To do so, we try various values for $R \in (2, 4, 8)$ and the results are demonstrated in Table VII. As can be seen, the lower error comes with the cost of involving more parameters. For real-time applications, the lower bandwidths are more applicable.

4) LLM Cross-compatibility: We evaluate our model's flexibility by deploying it across multiple LLM architectures.

TABLE VII
LoRA RANK BENCHMARK RESULTS ON PeMS04 DATASET

R	MAE↓	RMSE↓	MAPE(%)↓
2	24.76	29.37	12.80
4	<u>22.12</u>	<u>27.58</u>	<u>12.09</u>
8	**21.87**	**26.62**	**11.8**

Specifically, we utilized Mistral and Llama 2 as base models to assess the adaptability of our proposed method. The results, shown in Table VIII, indicate that the framework is adaptable to different LLM backbones without losing too much performance

TABLE VIII
ABLATION STUDY ON LLM BACKBONE ADAPTABILITY FOR PEMS07(M)

Backbone	MAE↓	RMSE↓	MAPE(%)↓
Llama2 [70]	2.81	5.63	6.81
Mistral [38]	**2.60**	**4.86**	**6.51**

V. CONCLUSION

This research proposes a probabilistic-based LLM for traffic forecasting called Strada-LLM, which injects the graph structure implicitly into the input tokens. We utilize a subgraph extraction procedure to inject the graph structure into the transformer token inputs. On the one hand, we adopt a probabilistic transformer to predict the traffic data by sampling from the underlying distribution. On the other hand, we utilize a low-rank method for the transfer learning task. Consequently, we showcase its success in adapting to datasets with significant differences compared to the datasets used for pre-training. Finally, the model is robust to noise, interpretable, and scalable. Strada-LLM is compliant to a variety of different known LLM backbones, which make its architecture suitable to process spatio-temporal graphs. In summary, the Strada-LLM model successfully captures the spatial and temporal features from traffic data so that they can be applied to other spatio-temporal tasks. A potential avenue for future research could involve evaluating the expressiveness of the distribution head.

ACKNOWLEDGMENT

This work is funded by Innoviris within the research project TORRES. The authors thank Leandro Di Bella, Weijiang Xiong, and Bert Van Hauwermeiren for comments that greatly improved the manuscript.

REFERENCES

[1] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *International Conference on Learning Representations*, 2018.

[2] Y. Liang, H. Wen, Y. Nie, Y. Jiang, M. Jin, D. Song, S. Pan, and Q. Wen, "Foundation models for time series analysis: A tutorial and survey," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, (New York, NY, USA), p. 6555–6565, Association for Computing Machinery, 2024.

[3] Z. Ni, H. Yu, S. Liu, J. Li, and W. Lin, "Basisformer: Attention-based time series forecasting with learnable and interpretable basis," 2024.

[4] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[5] S. Dooley, G. S. Khurana, C. Mohapatra, S. V. Naidu, and C. White, "Forecastpfn: Synthetically-trained zero-shot forecasting," in *Advances in Neural Information Processing Systems* (A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, eds.), vol. 36, pp. 2403–2426, Curran Associates, Inc., 2023.

[6] Y. Yuan, J. Ding, J. Feng, D. Jin, and Y. Li, "Unist: A prompt-empowered universal model for urban spatio-temporal prediction," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, (New York, NY, USA), p. 4095–4106, Association for Computing Machinery, 2024.

[7] Z. Li, L. Xia, J. Tang, Y. Xu, L. Shi, L. Xia, D. Yin, and C. Huang, "Urbangpt: Spatio-temporal large language models," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, (New York, NY, USA), p. 5351–5362, Association for Computing Machinery, 2024.

[8] H. Xue and F. D.Salim, "Promptcast: A new prompt-based learning paradigm for time series forecasting," *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[9] Y. Li, Z. Li, P. Wang, J. Li, X. Sun, H. Cheng, and J. X. Yu, "A survey of graph meets large language model: Progress and future directions," 2024.

[10] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 1907–1913, International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[11] H. Zhao, S. Liu, C. Ma, H. Xu, J. Fu, Z.-H. Deng, L. Kong, and Q. Liu, "GIMLET: A unified graph-text model for instruction-based molecule zero-shot learning," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[12] OpenAI, "Gpt-4 technical report," 2024.

[13] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," 2021.

[14] Z. Li, L. Xia, Y. Xu, and C. Huang, "Flashst: A simple and universal prompt-tuning framework for traffic prediction," 2024.

[15] J. D. Hamilton, *Time series analysis*. Princeton, NJ: Princeton University Press, Jan. 1994.

[16] P. H. Franses, "Seasonality, non-stationarity and the forecasting of monthly time series," *International Journal of forecasting*, vol. 7, no. 2, pp. 199–208, 1991.

[17] Q. Wen, J. Gao, X. Song, L. Sun, H. Xu, and S. Zhu, "Robuststl: A robust seasonal-trend decomposition algorithm for long time series," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 5409–5416, Jul. 2019.

[18] V. Assimakopoulos and K. Nikolopoulos, "The theta model: a decomposition approach to forecasting," *International Journal of Forecasting*, vol. 16, no. 4, pp. 521–530, 2000. The M3- Competition.

[19] R. J. Hyndman and B. Billah, "Unmasking the theta method," *International Journal of Forecasting*, vol. 19, no. 2, pp. 287–290, 2003.

[20] J. A. Fiorucci, T. R. Pellegrini, F. Louzada, F. Petropoulos, and A. B. Koehler, "Models for optimising the theta method and their relationship to state space models," *International journal of forecasting*, vol. 32, no. 4, pp. 1151–1161, 2016.

[21] K. Benidis, S. S. Rangapuram, V. Flunkert, Y. Wang, D. Maddix, C. Turkmen, J. Gasthaus, M. Bohlke-Schneider, D. Salinas, L. Stella, F.-X. Aubet, L. Callot, and T. Januschowski, "Deep learning for time series forecasting: Tutorial and literature survey," *ACM Computing Surveys*, vol. 55, p. 1–36, Dec. 2022.

[22] C. Song, Y. Lin, S. Guo, and H. Wan, "Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 914–921, Apr. 2020.

[23] C. Zheng, X. Fan, C. Wang, and J. Qi, "Gman: A graph multi-attention network for traffic prediction," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 1234–1241, Apr. 2020.

[24] D. Salinas, M. Bohlke-Schneider, L. Callot, R. Medico, and J. Gasthaus, "High-dimensional multivariate forecasting with low-rank gaussian copula processes," 2019.

[25] C. Shang, J. Chen, and J. Bi, "Discrete graph structure learning for forecasting multiple time series," in *International Conference on Learning Representations*, 2021.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[27] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, "A time series is worth 64 words: Long-term forecasting with transformers," 2023.

[28] B. Lim, S. O. Arik, N. Loeff, and T. Pfister, "Temporal fusion transformers for interpretable multi-horizon time series forecasting," 2020.

[29] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-beats: Neural basis expansion analysis for interpretable time series forecasting," 2020.

[30] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," 2021.

[31] S. M. Moghadas, A. Gheibi, and A. Alahi, "Statnet: Spatial-temporal attention in the traffic prediction," in *hEART 2022: 10th Symposium of the European Association for Research in Transportation*, 2022.

[32] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," 2022.

[33] G. Woo, C. Liu, D. Sahoo, A. Kumar, and S. Hoi, "Etsformer: Exponential smoothing transformers for time-series forecasting," 2022.

[34] S. Liu, H. Yu, C. Liao, J. Li, W. Lin, A. X. Liu, and S. Dustdar, "Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting," in *International Conference on Learning Representations*, 2022.

[35] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, "Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting," 2022.

[36] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan, "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," 2020.

[37] A. Ashok, Étienne Marcotte, V. Zantedeschi, N. Chapados, and A. Drouin, "Tactis-2: Better, faster, simpler attentional copulas for multivariate time series," 2024.

[38] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, "Mistral 7b," 2023.

[39] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, and et al., "On the opportunities and risks of foundation models," 2022.

[40] S. M. Moghadas, A. Gheibi, and A. Alahi, "Statnet: Spatial-temporal attention in the traffic prediction," 2022.

[41] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[42] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," *arXiv preprint arXiv:2104.08691*, 2021.

[43] R. Verkuil, O. Kabeli, Y. Du, B. I. M. Wicky, L. F. Milles, J. Dauparas, D. Baker, S. Ovchinnikov, T. Sercu, and A. Rives, "Language models generalize beyond natural proteins," *bioRxiv*, 2022.

[44] Y. Liang, H. Wen, Y. Nie, Y. Jiang, M. Jin, D. Song, S. Pan, and Q. Wen, "Foundation models for time series analysis: A tutorial and survey," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, (New York, NY, USA), p. 6555–6565, Association for Computing Machinery, 2024.

[45] C. Liu, S. Yang, Q. Xu, Z. Li, C. Long, Z. Li, and R. Zhao, "Spatial-temporal large language model for traffic prediction," *arXiv preprint arXiv:2401.10134*, 2024.

[46] M. Jin, S. Wang, L. Ma, Z. Chu, J. Y. Zhang, X. Shi, P.-Y. Chen, Y. Liang, Y.-F. Li, S. Pan, and Q. Wen, "Time-llm: Time series forecasting by reprogramming large language models," 2024.

[47] T. Zhou, P. Niu, X. Wang, L. Sun, and R. Jin, "One fits all:power general time series analysis by pretrained lm," 2023.

[48] X. Liu, J. Hu, Y. Li, S. Diao, Y. Liang, B. Hooi, and R. Zimmermann, "Unitime: A language-empowered unified model for cross-domain time series forecasting," 2024.

[49] D. Shrivastava, H. Larochelle, and D. Tarlow, "Repository-level prompt generation for large language models of code," 2023.

[50] Z. Fang, Q. Long, G. Song, and K. Xie, "Spatial-temporal graph ode networks for traffic flow forecasting," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, (New York, NY, USA), p. 364–373, Association for Computing Machinery, 2021.

[51] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), vol. 33, pp. 1877–1901, Curran Associates, Inc., 2020.

[52] G. Nikolentzos, G. Dasoulas, and M. Vazirgiannis, "k-hop graph neural networks," 07 2019.

[53] C. Zhou, R. Yu, and Y. Wang, "On the theoretical expressive power and the design space of higher-order graph transformers," in *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics* (S. Dasgupta, S. Mandt, and Y. Li, eds.), vol. 238 of *Proceedings of Machine Learning Research*, pp. 2179–2187, PMLR, 02–04 May 2024.

[54] S. Abu-El-Haija, B. Perozzi, A. Kapoor, N. Alipourfard, K. Lerman, H. Harutyunyan, G. V. Steeg, and A. Galstyan, "Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing," 2019.

[55] N. G. Trillos, F. Hoffmann, and B. Hosseini, "Geometric structure of graph laplacian embeddings," *Journal of Machine Learning Research*, vol. 22, no. 63, pp. 1–55, 2021.

[56] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," 2023.

[57] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 11106–11115, May 2021.

[58] J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," 2023.

[59] T. Dao, "Flashattention-2: Faster attention with better parallelism and work partitioning," 2023.

[60] J. M. Bernardo, M. J. Bayarri, J. O. Berger, A. P. Dawid, D. Heckerman, A. F. M. Smith, and M. West, *Bayesian Statistics 8: Proceedings of the Eighth Valencia International Meeting June 2–6, 2006*. Oxford University Press, 07 2007.

[61] N. Filatov and M. Kindulov, "Low rank adaptation for stable domain adaptation of vision transformers," *Optical Memory and Neural Networks*, vol. 32, no. Suppl 2, pp. S277–S283, 2023.

[62] S. Moghadas, Y. Lyu, B. Cornelis, and A. Munteanu, "STRADA: Spatial-Temporal Dashboard for traffic forecasting," in *2024 25th IEEE International Conference on Mobile Data Management (MDM)*, (Los Alamitos, CA, USA), pp. 251–254, IEEE Computer Society, jun 2024.

[63] M. Chen, Z. Xu, A. Zeng, and Q. Xu, "Fraug: Frequency domain augmentation for time series forecasting," 2023.

[64] D. P. Kingma, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[65] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "Connecting the dots: Multivariate time series forecasting with graph neural networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, (New York, NY, USA), p. 753–763, Association for Computing Machinery, 2020.

[66] Z. Shao, Z. Zhang, F. Wang, and Y. Xu, "Pre-training enhanced spatial-temporal graph neural network for multivariate time series forecasting," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, (New York, NY, USA), p. 1567–1577, Association for Computing Machinery, 2022.

[67] Z. Fu, W. Lam, Q. Yu, A. M.-C. So, S. Hu, Z. Liu, and N. Collier, "Decoder-only or encoder-decoder? interpreting language model as a regularized encoder-decoder," 2023.

[68] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," 2023.

[69] T. T. Cai and R. Ma, "Theoretical foundations of t-sne for visualizing high-dimensional clustered data," 2022.

[70] K. Rasul, A. Ashok, A. R. Williams, H. Ghonia, R. Bhagwatkar, A. Khorasani, M. J. D. Bayazi, G. Adamopoulos, R. Riachi, N. Hassen, M. Biloš, S. Garg, A. Schneider, N. Chapados, A. Drouin, V. Zantedeschi, Y. Nevmyvaka, and I. Rish, "Lag-llama: Towards foundation models for probabilistic time series forecasting," 2024.

Seyed Mohamad Moghadas received the M.S. degree of Computer Science from Amirkabir University of Technology Tehran, Iran, in 2023. He was a Software Engineer at Mahsan Company. He is currently PhD student in the Department of Electronics and Informatics, (VUB), and IMEC, Brussels, Belgium. He investigated the use of various graph neural networks in the transportation domain. His research interests include spatio-temporal data analysis and sparse representations in time-series analysis.

Yangxintong Lyu is a Post-Doctoral Researcher at the Electronics and Informatics department of the Vrije Universiteit Brussel. She received the M.Sc. degree in Applied Computer Science from Vrije Universiteit Brussel, Belgium, in 2018, and the Doctorate degree in Engineering Sciences (Summa Cum Laudae) from Vrije Universiteit Brussel, Belgium, in 2024. Prior to her current role, she worked as an R&D engineer at an autonomous driving startup. Her research focuses on Machine Learning, 2D/3D Computer Vision, and their applications in Intelligent Transportation Systems.

Bruno Cornelis received the M.S. degree of Industrial Engineer in electronics from Erasmushogeschool Brussel, Belgium, in 2005, the M.S. degree in electrical engineering and the Ph.D. degree in applied sciences and engineering from Vrije Universiteit Brussel (VUB), Belgium, in 2007 and 2014, respectively. He was a Visiting Assistant Professor with the Mathematics Department, Duke University. He is currently professor with the Department of Electronics and Informatics, (VUB), and iMinds, Ghent, Belgium. He investigated the use of various image processing tools in support of art scholarship. His research interests include statistical data analysis and sparse representations in image processing.

Alexandre Alahi (Member, IEEE) is currently an Assistant Professor with EPFL, where he is leading the Visual Intelligence for Transportation Laboratory (VITA). Before joining EPFL in 2017, he spent multiple years as a Post-Doctoral Researcher and a Research Scientist with Stanford University. His research interests include computer vision, machine learning, and robotics applied to transportation and mobility.

Adrian Munteanu (M'07) is professor at the Electronics and Informatics (ETRO) department of the Vrije Universiteit Brussel (VUB), Belgium. He received the MSc degree in Electronics and Telecommunications from Politehnica University of Bucharest, Romania, in 1994, the MSc degree in Biomedical Engineering from University of Patras, Greece, in 1996, and the Doctorate degree in Applied Sciences (Maxima Cum Laudae) from Vrije Universiteit Brussel, Belgium, in 2003. In the period 2004-2010 he was post-doctoral fellow with the Fund for Scientific Research – Flanders (FWO), Belgium, and since 2007, he is professor at VUB. His research interests include image, video and 3D graphics compression, deep learning, distributed visual processing, error-resilient coding, and multimedia transmission over networks. He is the author of more than 400 journal and conference publications, book chapters, and contributions to standards, and holds 7 patents in image and video coding. He is the recipient of the 2004 BARCO-FWO prize for his PhD work, the (co-)recipient of the Most Cited Paper Award from Elsevier for 2007, and of 10 other scientific prizes and awards. He served as Associate Editor for IEEE Transactions on Multimedia and currently serves as Associate Editor for IEEE Transactions on Image Processing.