Strada-LLM: Graph LLM for traffic prediction

Seyed Mohamad Moghadas
Department of Electronics and Informatics, Vrije Universiteit Brussel
B-1050 Brussels, Belgium
Seyed.Mohamad.Moghadas@vub.be

Bruno Cornelis*
Department of Electronics and Informatics, Vrije Universiteit Brussel
B-1050 Brussels, Belgium
bcorneli@etrovub.be

Alexandre Alahi†
VITA Lab, EPFL
Lausanne, Switzerland
alexandre.alahi@epfl.ch

Adrian Munteanu‡
Department of Electronics and Informatics, Vrije Universiteit Brussel
B-1050 Brussels, Belgium
Adrian.Munteanu@vub.be

Abstract

Traffic forecasting is pivotal for intelligent transportation systems, where accurate and interpretable predictions can significantly enhance operational efficiency and safety. A key challenge stems from the heterogeneity of traffic conditions across diverse locations, leading to highly varied traffic data distributions. Large language models (LLMs) show exceptional promise for few-shot learning in such dynamic and data-sparse scenarios. However, existing LLM-based solutions often rely on prompt-tuning, which can struggle to fully capture complex graph relationships and spatiotemporal dependencies—thereby limiting adaptability and interpretability in real-world traffic networks.

We address these gaps by introducing Strada-LLM, a novel multivariate probabilistic forecasting LLM that explicitly models both temporal and spatial traffic patterns. By incorporating proximal traffic information as covariates, Strada-LLM more effectively captures local variations and outperforms prompt-based existing LLMs. To further enhance adaptability, we propose a lightweight distribution-derived strategy for domain adaptation, enabling parameter-efficient model updates when encountering new data distributions or altered network topologies—even under few-shot constraints.

Empirical evaluations on spatio-temporal transportation datasets demonstrate that Strada-LLM consistently surpasses state-of-the-art LLM-driven and traditional GNN-based predictors. Specifically, it improves long-term forecasting by 17% in RMSE error and 16% more efficiency. Moreover, it maintains robust performance across different LLM backbones with minimal degradation, making it a versatile and powerful solution for real-world traffic prediction tasks.

*Also with Macq.
†Also with VITA Lab, EPFL.
‡Member, IEEE

KDD '25,

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/XXXXXXX.XXXXXXX

CCS Concepts

• **Computing methodologies**; • **Machine learning approaches**;

Keywords

spatio-temporal transformers, graph neural networks, traffic prediction, LLM

ACM Reference Format:
Seyed Mohamad Moghadas, Bruno Cornelis, Alexandre Alahi, and Adrian Munteanu. 2025. Strada-LLM: Graph LLM for traffic prediction. In *Proceedings of August 03–07, 2025 (KDD '25).* ACM, New York, NY, USA, 13 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

With the development of Intelligent Transportation Systems, spatio-temporal traffic prediction has received increasing attention. It is a key component of advanced urban management systems and is crucial in urban planning, mobility management, and resource allocation. Spatio-temporal prediction involves analyzing urban conditions on various dimensions, including flow, speed, and density, mining their patterns, and predicting trends [14]. This capability provides a scientific foundation for any urban management department to anticipate and mitigate congestion, implement preemptive restrictions, and enable urban residents to select safer and more efficient travel routes.

However, spatio-temporal prediction is a challenging task due to complex spatial and temporal dependencies:

1) Spatial Dependency: The variation in urban patterns is influenced by the topological structure of the urban network. In particular, conditions at upstream locations impact downstream locations through transfer effects, which refers to the impact that a change in one section (e.g., a closure of an area, a change in service timing, or new infrastructure) has on other parts of the network. On the other hand, downstream locations affect upstream locations through feedback effects, which occur when the changes in patterns resulting from transfer effects influence the original cause of the change, creating a loop of cause and effect.

2) Temporal Dependency: Urban conditions change over time, exhibiting periodicity and trends, which are often influenced by factors such as holidays, working hours, and other social events. As illustrated in Figure 1, the strong influence among adjacent locations caused by the flow alongside the edges changes the short-term

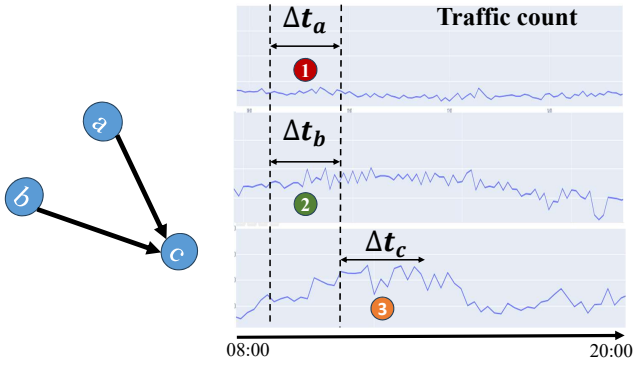


Figure 1: An example of the spatio-temporal dependency between the traffic flows for different traffic states ①,②,③. Nodes (a), (b), and (c) are placed in a crowded area of Brussels. The corresponding traffic patterns between 8:00 am and 8:00 pm are illustrated for nodes (a), (b), and (c). The short-term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

states. Indeed, location (a) has a steady count trend, and location (b) has short spikes in volume, so the spatio-temporal correlation yielded a different state in location (c) after a couple of hours. Furthermore, urban data sources exhibit a variety of data distributions, posing a significant challenge for finding generalizable models.

In recent years, the emergence of Large Language Models (LLM) has revolutionized various fields due to the generalization capabilities of extracting efficient features from different modalities [34]. Research has been conducted in multiple domains, such as computer vision and natural language processing. Specifically, in [34], 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 urban prediction, where data distributions can vary significantly across different regions and times.

Previous works [15, 16] 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 suboptimal 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 characteristics and trends of time series [22]. It leads to a performance gap compared to classical Graph Neural Networks [24]. 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.

Effective LLM adaptation approach have a noticeable impact on their Zero-Shot performance. There are three prominent approaches to LLM adaptation[2]. Namely, Prompt-Tuning[20], Adapter-Layer[2], Parameter-Efficient Fine-Tuning (PEFT)[11]. The adapter-layer approach [2] tends to be more applicable in the multi-task setup for foundational models[2]. In contrast, the effectiveness of Prompt-based approaches and PEFT remains a point of contention in the literature. Both methods have emerged as favored for integrating domain-specific expertise into Large Language Models (LLMs). However, in the realms of existing urban LLMs, UrbanGPT[15] and UniST[34] have focused on Prompt-based approaches. Specifically, they include a **learnable** prompt network (PN) into the LLM. Essentially, the PN is fine-tuned in the adaptation stage. Prompt networks could impose extra complexity and neglect scalable features in the spatio-temporal target domain. Specifically, PN relies on the memory network[26], which has been proven in graph-based spatio-temporal data to cause exponential error propagation[3]. On the other hand, PEFT[11] 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. However, the competence and scalability of PEFT in the LLM-based urban traffic forecasting domain remain unexplored.

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

(1) Strada-LLM is a probabilistic LLM compliant with classical LLMs and specializes in spatio-temporal prediction. Specifically, Strada-LLM has the capability to predict urban traffic patterns, even when faced with a data distribution that differs from its training data, particularly in the few-shot setting.
(2) To the best of our knowledge, Strada-LLM is the first non-prompt-based probabilistic LLM that has lightweight adaptation capability, effectively capturing both local and global spatial dependencies.
(3) Strada-LLM is graph-aware, taking spatial dependencies into account by encoding the network graph implicitly and without any prompting. This allows the model to maintain crucial topological information throughout the prediction process.
(4) Strada-LLM adopts a lightweight approach to perform domain adaptation. Specifically, adopting low-ranked approaches showcases its effectiveness in adapting to new datasets while maintaining computational efficiency.
(5) We provide a comprehensive evaluation and demonstrate superior accuracy compared to existing urban LLMs. We evaluate our approach using numerous real-world datasets. Our results show a 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 urban forecasting.

The remainder of this paper is organized as follows: Section 2 presents the preliminaries, Section 3 describes the methodology, Section 4 discusses the results, and Section 5 concludes.

2 Preliminaries

Spatio-Temporal Forecasting. 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 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 multivariate point prediction problem. In particular, Strada-LLM takes the T past observations $\boldsymbol{X} = \left[\boldsymbol{X}_{t_1}, \boldsymbol{X}_{t_2}, ..., \boldsymbol{X}_{t_T}\right] \in \mathbb{R}^{T\times N\times F}$ from G as input and predicts the traffic for horizon T', where:
$\hat{\boldsymbol{X}} = \left[\hat{\boldsymbol{X}}_{t_{T+1}}, \hat{\boldsymbol{X}}_{t_{T+2}}, ..., \hat{\boldsymbol{X}}_{t_{T+T'}}\right]$.

3 Methodology

Strada-LLM consists of a Hierarchical Feature Extractor (HFE), an LLM-based backbone, and a T-student distribution head, as shown in Figure 2. The distribution head includes three learnable parameters related to the output: degrees of freedom, mean, and scale, to ensure the relevant parameters remain positive. 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.

3.1 Hierarchical Feature Extractor

3.1.1 k-hop Sub-Graph Extractor: To enhance the LLM's understanding of graph structures while accommodating its limited context length, we tokenize the spatio-temporal traffic signal by taking into account 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 [19] 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 $|\mathcal{N}_k(v)|$, cardinality of neighbors. Similarly, Subgraph-1-WL [36] and MixHop [1] can also be used as the k-hop operator, but it is beyond the scope of this research.

3.1.2 Global Graph Embedding. While prompt-based networks are widely used in the literature [17], they lose the global structure information [16]. As demonstrated by [28], in order to keep the global graph structure information, we compute Laplacian embeddings. The Laplacian operation leverages the eigenvectors of the complete graph and creates unique positional encodings [29] for each node. First, we define the graph Laplacian matrix $\mathbf{L} = D - A$ where $D = \text{diag}(d_1, \ldots, d_N)$ is the degree matrix. We then compute the eigendecomposition of the normalized Laplacian as follows $L_{\text{norm}} = D^{-1/2}\mathbf{L}D^{-1/2} = \mathbf{U}\Lambda\mathbf{U}^T$. Then we compute global embedding by non-linear mapping of $\mathbf{U}$ matrix. Reader can refer to Section 4.4.3 to discern the contribution of this representation extraction on the LLM forecasting ability. We denote F' as the embedding dimension.

3.1.3 Lag Extractor. Mathematically, this lagging operation can be expressed as a mapping $\boldsymbol{x_t} \longmapsto \boldsymbol{\lambda_t} \in \mathbb{R}^{|\mathcal{H}|\times F}$, where $\mathcal{H} = \{h_1, h_2, ..., h_H\}$ represents the set of historical lag indices. In this formulation, $\lambda_t[j] = x_{t-\mathcal{H}[j]}$, meaning that $\lambda_t[j]$ corresponds to the value of x at h_j time steps before t, as specified by the j-th element of $\mathcal{H}$.

In our approach, we extend beyond simple lag feature extraction by employing a structured tokenization scheme to convert spatio-temporal slices of the road network graph into discrete tokens for the LLM-based backbone. Specifically, for each time step t, we first derive both global and local (k-hop) graph embeddings. These representations are then segmented within a sliding window, yielding embeddings that encapsulate the node-level or sub-graph-level features alongside their temporal contexts. Crucially, we add positional embeddings to each token so that the masked Transformer decoder can effectively attend to different time steps in a sequence-like manner. While lag features help capture short-term correlations, the tokenization scheme formalizes how each time window becomes a coherent input unit, thus ensuring the LLM's attention mechanisms can leverage meaningful traffic patterns and dependencies across time and space.

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 H 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. In the end, the HFE block's output is a tensor in shape $\mathbb{R}^{C\times(1+M)\times|\mathcal{H}|\times(F+F')}$.

3.2 LLM-based-Backbone

Our backbone architecture is inspired by Mistral [12], 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 [27], Strada-LLM includes pre-normalization by utilizing the RMSNorm [37] and Rotary Positional Encoding (RoPE) [25] for the representation of query and key blocks at each attention layer, similar to the approach described for Mistral [12].

After passing through the causally masked transformer layers, the proposed method incorporates Flash-Attention-2 [7] 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 $H + C$, 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 auto-regressive decoding [4]. In doing so, uncertainty intervals and confidence scores could also be valuable for downstream decision-making tasks.

3.3 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 [11], which aims to refine pre-trained models by adapting their parameters in a computationally efficient manner; indeed, a nearly linear operation will replace the $O(n^2)$ 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.

3.3.1 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 [9]. One of the purposes of this research is to propose a LoRA-based method for the temporal graph forecasting problem. As illustrated in Figure 3, we apply LoRA for the LLM fine-tuning process such as spatio-temporal patterns of the target data being learned. The query, key, and value matrices will be fine-tuned in a low-rank approximation manner. Specifically, according to Figure 3, 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, online inference latency reduction is achieved [11]. For the initialization of the matrices A we choose Gaussian initialization while the B matrices are initialized to zero [11]. Following this approach enables us to perform domain adaptation for different datasets consisting of diverse distributions. As shown in Figure 3(a), our adaptation approach is conceptually different than prompt-based models like UniST [34]. Based on the distribution-driven alignment loss function in the adaptation phase, global graph embedding, query, key, value projections, and distribution head are targeted to be tuned in the low-rank fashion. As reader can see in the experiment section 4.4.4, tuning the mentioned blocks is more lightweight than tuning prompt-based models like Figure 3(b). One might critique not applying low-rank tuning for other modules like Global Graph Embedding in Figure 3(a), which is addressed in Appendix A.4.

Recent theoretical work shows that domain alignment can be bounded using the optimal transport distance functions [33]. For instance, unsupervised distance adaptation approaches [21] have been adopted in modern LLMs. As shown in Figure 3(a), In Strada-LLM, the distribution head will be tuned. So, as another novel pillar of our model, during the adaptation phase, we minimize Kullback–Leibler divergence $D_{KL}(.)$ distance between multivariate distribution heads. As a matter of tractability, given a bounded k-variate t-distribution, the distance function can be derived as:

$$\begin{aligned} &D_{KL}(f_d(\cdot|\mu,\Sigma,\nu)||f_d(\cdot|\mu,\Sigma,\nu')) = \\ &= D_{KL}(f_d(\cdot|0,\mathbf{I},\nu)||f_d(\cdot|0,\mathbf{I},\nu')) \\ &= \int_{\mathbb{R}^n} f_d(\mathbf{x}|0,\mathbf{I},\nu)\log\frac{f_d(\mathbf{x}|0,\mathbf{I},\nu)}{f_d(\mathbf{x}|0,\mathbf{I},\nu')}d\mathbf{x} \end{aligned}$$

Inspired by [30], this Kullback Leibler divergence can be written as:

$$\begin{aligned} &D_{KL}(f_d(\cdot|\mu,\Sigma,\nu)||f_d(\cdot|\mu,\Sigma,\nu')) = \\ &\log\frac{K(d,\nu)}{K(d,\nu')} - \frac{\nu+d}{2}\left[\Psi\left(\frac{\nu+d}{2}\right) - \Psi\left(\frac{\nu}{2}\right)\right] + \\ &\frac{\nu'+d}{2}K(d,\nu)\frac{\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2})}\int_0^\infty\left(1+\frac{t}{\nu}\right)^{-\frac{\nu+d}{2}} t^{\frac{d}{2}-1}\log\left(1+\frac{t}{\nu'}\right)dt \\ &\coloneqq \mathcal{L}_{alignment} \end{aligned} \tag{4}$$

where, $K(d,\nu) = \frac{\Gamma(\frac{\nu+d}{2})}{\Gamma(\frac{\nu}{2})\sqrt{(\pi\nu)^d}}$. The proof unfolds in Appendix A.2.

3.4 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:

$$\text{NLL} = -\sum_{x\in\mathcal{D}} log(P(y|x)),\ \mathcal{L}_{total} = \lambda_1 NLL + (1-\lambda_1)\mathcal{L}_{alignment} \tag{5}$$

where y is the true value, x is the input data, $\mathcal{D}$ is the training set, $P(y|x)$ is the predicted conditional probability of y given x, and λ_1 is the loss component hyperparameter. In multivariate modeling, NLL can be modeled differently. In Strada-LLM, we estimate the joint distribution, but further experiments are discussed in the ablation study 4.4.5.

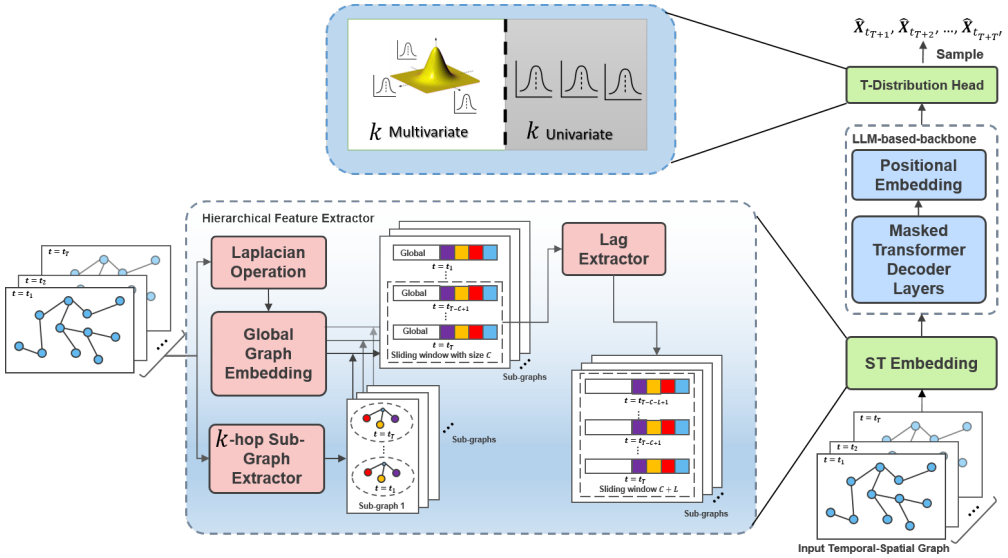
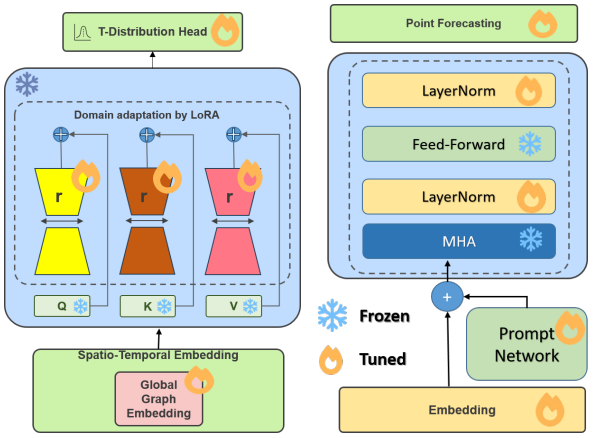


Figure 2: The proposed Strada-LLM architecture. The core novelty lies in the K-hop sub-graph extractor module, which takes lag features in both the spatial and temporal dimensions.



(a) Strada-LLM (our method) (b) UniST[34]

Figure 3: Conceptual LLM adaptation approaches comparison

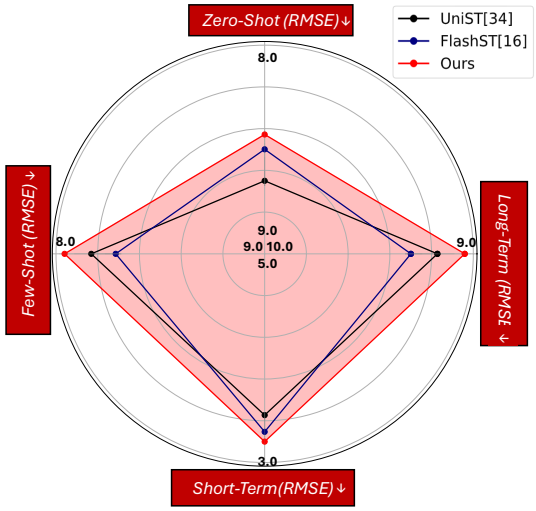


Figure 4: Comparison of unified benchmark for different LLMs.

4 EXPERIMENTS

4.1 Datasets 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 1. Most of these datasets pertain to traffic speed. For the sake of simplicity, traffic speed is utilized as the primary traffic metric in our experimental analysis; thus one can presume $F = 1$. Detailed training strategies, e.g., hyperparameters and experimental setup, are explained in Appendix A.5.

4.2 Baselines

We compare the performance of the Strada-LLM model with **spatio-temporal** methods, GMAN [35], MTGNN [31], GTS [23], STEP [24], and **LLM-based** methods which either are **point-wise forecasters** such as FlashST [16], UniST [34] or **probabilistic forecasters** such as Lag-llama[22]. The details of the baseline methods are explained in Appendix A.1. To compare with the baselines, we adopt the MAE, RMSE, and MAPE prediction error metrics [16]. Due to Strada-LLM being a probabilistic model, we also compare with Lag-Llama [22] using the Continuous Ranked Probability Score

Table 1: Dataset Description

Dataset	Data Type	Number of Nodes	Region	Time Steps
Pretraining Datasets				
PeMS03 [8]	Traffic Speed	170	California	26208
PeMS04 [8]	Traffic Speed	307	California	16992
PeMS07 [8]	Traffic Speed	883	California	26208
PeMS08 [8]	Traffic Flow	170	California	17856
Few-Shot				
METR-LA [8]	Traffic Speed	207	California	34272
PEMS-Bay [8]	Traffic Speed	325	California	52116
PeMS07(M) [8]	Traffic Speed	228	California	12672
Brussels [18]	Traffic Count	207	Brussels	12672
Zero-Shot				
Crowd [34]	Pedestrian Count	16×20	Nanjing	9420
PeMS07(M) [8]	Traffic Count	228	California	12672
Brussels [18]	Traffic Count	207	Brussels	12672

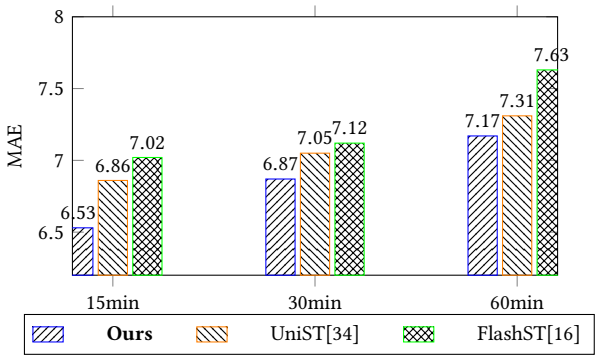


Figure 5: Zero-shot performance comparison on Crowd dataset.

(CRPS) metric, defined as:

$$CRPS(F, y) = \int_{-\infty}^{\infty} (F(x) - H(x - y))^2 dx,$$

where $F(x)$ is the cumulative distribution function (CDF) of the forecasted distribution, y is the observed value, and $H(x - y)$ is the Heaviside step function, which is 0 when $x < y$ and 1 when $x \geq y$.

4.3 LLM Forecasting Results

4.3.1 LLM Short-Term Forecasting. The short-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.

The prediction results for the datasets METR-LA and PEMS-Bay are reported in Tables 2–3, respectively. The ability to perform forecasting and being robust to new domains are important qualities for an LLM. As shown in Tables 2– 3, the proposed LLM has competitive performance compared to STEP [24], which is the corresponding transformer-based fully-supervised method. Readers can refer to Appendix 10 about robustness analysis. On the other hand, compared to the best prompt-tuning method FlashST [16], shows that taking the global structure of the graph into account brings forth a definite advantage. Moreover, although Strada-LLM and STEP [24] are both transformer-based models, the introduction of normalization blocks like RMSNorm [37] and Rotary Positional Encoding (RoPE) differentiates them. Additionally, while STEP [24] is an encoder-decoder model, Strada-LLM is a decoder-only model, which makes it optimized in terms of performance for inference [10]. As we can see in Tables 2–3, the proposed Strada-LLM manages to outperform the baseline models on the aforementioned datasets in the longer prediction horizon more significantly. 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 [16].

4.3.2 LLM Long-Term Forecasting. Followed by the setup of [34], we experiment with long-term forecasting for horizons **90, 120, and 150** minutes. The long-term traffic prediction accuracy was evaluated across two datasets: Crowd and PEMS07(M). We report the performance in Table 4. Consistently, Strada-LLM outperforms the LLM-based baselines. The successful performance can be attributed to the Graphical modeling of spatial covariates. Further visualizations are presented in the Appendix A.3.

4.3.3 LLM Zero-Shot Forecasting. The zero-shot traffic prediction accuracy was evaluated across three datasets: Crowd, Pems07(M), and Brussels. Table 7 presents the results of zero-shot capability of Strada-LLM compared to the other LLMs. Further visualizations are presented in the Appendix A.3. Strada-LLM is able to capture the spatio-temporal dependencies between connected nodes and prediction correlated trends, which is neglected by UniST [34]. To demonstrate the generalizability to the other urban datasets, Strada-LLM's performance on the Crowd dataset [34] is demonstrated in Figure 5, which outperforms other LLM baselines.

4.3.4 LLM Probabilistic Forecasting. Table 6 reports the CRPS metric, likewise Lag-llama[22], it has been implemented as the mean quantile loss. We observe that Strada-LLM consistently achieve strong few-shot and few-shot performance, obtaining either the best or second-best results for the Pems07 and Brussels datasets.

4.4 Ablation Study

4.4.1 Domain Adaptation. We examine the adaptation capabilities of Strada-LLM by evaluating the gap 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 8 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 8 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 Pems-Bay as an example, selecting Pems04 as the pertaining dataset for LLM ends up the 14.07% superior performance in the adaptation phase compared to Metr-LA, because of the inherent closer distribution. Detailed ablated studies about the performance of LoRA in terms of the rank parameter are discussed in Appendix A.6.

Table 2: Traffic short-term forecasting comparison with the state-of-the-art methods on the METR-LA dataset

	Baselines	15 min			30 min			60 min		
		MAE↓	RMSE↓	MAPE(%)↓	MAE↓	RMSE↓	MAPE(%)↓	MAE↓	RMSE↓	MAPE(%)↓
Supervised	GMAN [35]	2.80	5.55	7.41	3.12	6.49	8.73	3.44	7.35	10.07
	MTGNN [31]	2.69	5.18	6.88	3.05	6.17	8.19	3.49	7.23	9.87
	GTS [23]	2.67	5.27	7.21	3.04	6.25	8.41	3.46	7.31	9.98
	STEP[24]	2.61	4.98	6.6	2.99	5.97	7.96	3.37	6.99	9.61
LLM	FlashST [16]	2.84	5.57	7.45	3.19	6.43	9.04	3.60	7.44	10.68
	UniST [34]	2.89	5.63	7.49	3.26	6.51	9.09	3.65	7.52	10.79
	Strada-LLM(ours)	**2.45**	**4.71**	**6.42**	**2.97**	**5.84**	**7.87**	**3.35**	**6.73**	**9.47**

Table 3: Traffic short-term predictions comparison with the state-of-the-art methods on PEMS-Bay dataset

	Baselines	15 min			30 min			60 min		
		MAE↓	RMSE↓	MAPE(%)↓	MAE↓	RMSE↓	MAPE(%)↓	MAE↓	RMSE↓	MAPE(%)↓
Supervised	GMAN [35]	1.34	2.91	2.86	1.63	3.76	3.68	1.86	4.32	4.37
	MTGNN [31]	1.32	2.79	2.77	1.65	3.74	3.69	1.94	4.49	4.53
	GTS [23]	1.34	2.83	2.82	1.66	3.78	3.77	1.95	4.43	4.58
	STEP[24]	**1.26**	**2.73**	**2.59**	1.55	3.58	3.43	1.79	4.20	4.18
LLM	FlashST [16]	1.37	2.96	2.88	1.72	3.98	3.89	2.02	4.69	4.79
	UniST [34]	1.42	3.01	2.91	1.78	3.99	3.91	2.07	4.72	4.81
	Strada-LLM (Ours)	1.28	2.77	2.61	**1.53**	**3.14**	**3.37**	**1.76**	**3.94**	**4.09**

Table 4: Long-Term prediction experiment on the PEMS07 (M) and Brussels dataset (p-value < 0.05)

Baselines	PEMS07(M)									Brussels								
	90 min			120 min			150 min			90 min			120 min			150 min		
	MAE↓	RMSE↓	MAPE(%)↓	MAE↓	RMSE↓	MAPE(%)↓	MAE↓	RMSE↓	MAPE(%)↓	MAE↓	RMSE↓	MAPE(%)↓	MAE↓	RMSE↓	MAPE(%)↓	MAE↓	RMSE↓	MAPE(%)↓
FlashST [16]	4.21	6.98	10.71	4.51	7.20	11.92	4.83	7.54	11.38	6.35	8.53	9.57	6.47	8.89	9.84	6.79	9.31	10.18
UniST [34]	3.98	6.71	10.84	4.09	6.66	11.02	4.34	7.47	11.25	6.27	8.51	9.54	6.42	8.85	9.82	6.82	9.32	10.14
Strada-LLM(ours)	**3.84**	**6.02**	**10.53**	**4.04**	**6.51**	**11.01**	**4.19**	**7.01**	**11.08**	**5.83**	**8.31**	**9.39**	**6.21**	**8.70**	**9.77**	**6.55**	**9.01**	**10.03**

Table 5: 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 [16]	3.37	5.14	5.93	3.59	5.25	6.84	3.83	6.81	7.46	4.73	7.54	8.34	5.24	8.59	9.53	6.06	9.01	10.04
UniST [34]	2.45	4.89	5.06	2.67	5.13	6.56	2.89	6.59	7.50	4.70	7.29	8.09	5.29	8.42	9.34	5.90	8.96	9.94
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**	**8.05**	**5.16**	**8.14**	**9.03**	**5.86**	**8.57**	**9.52**

Table 6: Probabilistic forecasting experiment on the PEMS07(M) and Brussels dataset

Baselines	PEMS07(M)			Brussels		
	CRPS↓			CRPS↓		
	15 min	30 min	60 min	15 min	30 min	60 min
Lag-llama [22]	3.95	4.13	4.92	5.07	5.77	5.91
Strada-LLM(ours)	**3.13**	**3.67**	**4.03**	**4.93**	**5.10**	**5.31**

4.4.2 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 6 which is obtained with T-SNE [5] in 2D dimensions. Our variational posterior distribution $P(\theta \mid X)$ can truly depict the underlying distribution.

4.4.3 Scalability of K-hop structure. Our experimental analysis explores Strada-LLM's sensitivity to different neighborhood radii by examining performance across multiple k-hop values ($k \in \{1, 3, 5\}$).

Table 7: Zero-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 [16]	2.77	5.32	6.01	**2.91**	5.61	**6.75**	3.11	7.11	7.98	4.94	7.92	8.79	5.89	8.86	9.83	6.46	9.42	10.45
UniST [34]	2.92	5.42	6.05	2.99	**5.49**	6.92	3.31	7.02	7.90	4.92	7.69	8.53	5.92	8.88	9.85	6.49	9.59	10.64
Strada-LLM(ours)	**2.76**	**4.77**	**5.84**	2.93	5.80	6.79	**3.09**	**6.81**	**8.10**	**4.90**	**7.48**	**8.30**	**5.86**	**8.73**	**9.69**	**6.36**	**9.39**	**10.42**

Table 8: Domain adaptation comparison for demonstrating selection of effective source dataset

Source → Target		Top^k Fine-tuning		**Ours**		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
PeMS04	Pems-BAY	2.73	5.02	1.96	4.15	1.67	3.67	1.54	3.53

Table 9: The computation cost on the PEMS07(M) with the batchsize=64.

Method	Parameters(M)	Trainable Parameters(M)	Trainable Ratio(%)↓
FlashST[16]	64.02	1.01	1.58
UniST[34]	30.00	0.71	2.39
Strada-LLM (ours)	40.00	0.52	**1.32**

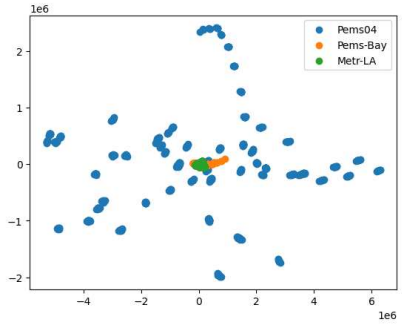


Figure 6: Latent space embedding of the last layer (before the distribution head) visualized in a 2D space.

The results in Table 10 demonstrate a trade-off between prediction accuracy and neighborhood size, with larger radii yielding lower error rates. Comparative analysis presented in Table 6 shows that Strada-LLM achieves superior performance over Lag-Llama [22], which did not take neighbors into account, primarily due to our model's incorporation of spatial dependencies through the K-hop block structure, a feature absent in the baseline approach.

4.4.4 Computation Cost. We evaluate the performance of our proposed LoRA-based approach compared to existing LLMs. Results are demonstrated in Table 9. As can be seen, the lower error comes with the cost of involving more parameters. However, in contrast to UniST [34], Strada-LLM's adaptation outperforms the compression rate, from 1.58(%) to 1.32(%), which demonstrates its efficiency.

Table 10: Feature Scalability Analysis Benchmark

K	MAE↓	RMSE↓	MAPE(%)↓
1	2.96	5.34	7.35
3	**2.60**	**4.86**	**6.51**
5	2.63	4.89	6.54

For real-time applications, more lightweight models are being demanded, which remains the future direction. Performance over parameter analysis is illustrated in Appendix A.8.

4.4.5 Distribution modeling. We evaluate our model's flexibility by deploying it across multiple LLM architectures. So, we report the performance in Table 11. Consistently, Strada-LLM outperforms the LLM-based baselines. The successful performance can be attributed to the Graphical modeling of spatial covariates.

Table 11: Ablation study on distribution modeling for PEMS07(M)

Backbone	MAE↓	RMSE↓	MAPE(%)↓
Univariate	2.81	5.63	6.81
Independent	2.69	5.37	6.73
Joint	**2.60**	**4.86**	**6.51**

4.4.6 LLM Cross-Compatibility. We evaluate our model's flexibility by deploying it across multiple LLM architectures. Specifically, we utilized Mistral and Llama 2 as base models to assess the adaptability of our proposed method. The results are shown in Table 12.

5 Conclusion

This research proposes a probabilistic-based LLM for traffic forecasting called Strada-LLM, which injects the local 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

Table 12: Ablation study on LLM backbone adaptability for PEMS07(M)

Backbone	MAE↓	RMSE↓	MAPE(%)↓
Llama2 [22]	2.83	5.66	6.89
Mistral [12]	**2.60**	**4.86**	**6.51**

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, Strada-LLM is compliant to a variety of known LLM backbones. 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.

References

[1] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. 2019. MixHop: Higher-Order Graph Convolutional Architectures via Sparsified Neighborhood Mixing. arXiv:1905.00067 [cs.LG]

[2] Simone Alghisi, Massimo Rizzoli, Gabriel Roccabruna, Seyed Mahed Mousavi, and Giuseppe Riccardi. 2024. Should We Fine-Tune or RAG? Evaluating Different Techniques to Adapt LLMs for Dialogue. In *Proceedings of the 17th International Natural Language Generation Conference*, Saad Mahamood, Nguyen Le Minh, and Daphne Ippolito (Eds.). Association for Computational Linguistics, Tokyo, Japan, 180–197. https://aclanthology.org/2024.inlg-main.15/

[3] Federico Barbero, Ameya Velingker, Amin Saberi, Michael Bronstein, and Francesco Di Giovanni. 2024. Locality-Aware Graph-Rewiring in GNNs. arXiv:2310.01668 [cs.LG] https://arxiv.org/abs/2310.01668

[4] J M Bernardo, M J Bayarri, J O Berger, A P Dawid, D Heckerman, A F M Smith, and M West. 2007. *Bayesian Statistics 8: Proceedings of the Eighth Valencia International Meeting June 2–6, 2006.* Oxford University Press. doi:10.1093/oso/9780199214655.001.0001

[5] T. Tony Cai and Rong Ma. 2022. Theoretical Foundations of t-SNE for Visualizing High-Dimensional Clustered Data. arXiv:2105.07536 [stat.ML] https://arxiv.org/abs/2105.07536

[6] Muxi Chen, Zhijian Xu, Ailing Zeng, and Qiang Xu. 2023. FrAug: Frequency Domain Augmentation for Time Series Forecasting. https://openreview.net/forum?id=j83rZLZgYBv

[7] Tri Dao. 2023. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. arXiv:2307.08691 [cs.LG] https://arxiv.org/abs/2307.08691

[8] Zheng Fang, Qingqing Long, Guojie Song, and Kunqing Xie. 2021. Spatial-Temporal Graph ODE Networks for Traffic Flow Forecasting. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (Virtual Event, Singapore) *(KDD '21)*. Association for Computing Machinery, New York, NY, USA, 364–373. doi:10.1145/3447548.3467430

[9] N Filatov and M Kindulov. 2023. Low Rank Adaptation for Stable Domain Adaptation of Vision Transformers. *Optical Memory and Neural Networks* 32, Suppl 2 (2023), S277–S283.

[10] Zihao Fu, Wai Lam, Qian Yu, Anthony Man-Cho So, Shengding Hu, Zhiyuan Liu, and Nigel Collier. 2023. Decoder-Only or Encoder-Decoder? Interpreting Language Model as a Regularized Encoder-Decoder. arXiv:2304.04052 [cs.CL] https://arxiv.org/abs/2304.04052

[11] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685 [cs.CL]

[12] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. arXiv:2310.06825 [cs.CL] https://arxiv.org/abs/2310.06825

[13] Diederik P Kingma. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[14] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *International Conference on Learning Representations*. https://openreview.net/forum?id=SJiHXGWAZ

[15] Zhonghang Li, Lianghao Xia, Jiabin Tang, Yong Xu, Lei Shi, Long Xia, Dawei Yin, and Chao Huang. 2024. UrbanGPT: Spatio-Temporal Large Language Models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Barcelona, Spain) *(KDD '24)*. Association for Computing Machinery, New York, NY, USA, 5351–5362. doi:10.1145/3637528.3671578

[16] Zhonghang Li, Lianghao Xia, Yong Xu, and Chao Huang. 2024. FlashST: A Simple and Universal Prompt-Tuning Framework for Traffic Prediction. arXiv:2405.17898 [cs.LG] https://arxiv.org/abs/2405.17898

[17] Yuxuan Liang, Haomin Wen, Yuqi Nie, Yushan Jiang, Ming Jin, Dongjin Song, Shirui Pan, and Qingsong Wen. 2024. Foundation Models for Time Series Analysis: A Tutorial and Survey. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Barcelona, Spain) *(KDD '24)*. Association for Computing Machinery, New York, NY, USA, 6555–6565. doi:10.1145/3637528.3671451

[18] S. Moghadas, Y. Lyu, B. Cornelis, and A. Munteanu. 2024. STRADA: Spatial-Temporal Dashboard for traffic forecasting. In *2024 25th IEEE International Conference on Mobile Data Management (MDM)*. IEEE Computer Society, Los Alamitos, CA, USA, 251–254. doi:10.1109/MDM61037.2024.00052

[19] Giannis Nikolentzos, George Dasoulas, and Michalis Vazirgiannis. 2019. k-hop Graph Neural Networks. doi:10.48550/arXiv.1907.06051

[20] Yushan Qian, Weinan Zhang, and Ting Liu. 2023. Harnessing the Power of Large Language Models for Empathetic Response Generation: Empirical Investigations and Improvements. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 6516–6528. doi:10.18653/v1/2023.findings-emnlp.433

[21] Shyam Sundhar Ramesh, Yifan Hu, Iason Chaimalas, Viraj Mehta, Pier Giuseppe Sessa, Haitham Bou Ammar, and Ilija Bogunovic. 2024. Group Robust Preference Optimization in Reward-free RLHF. arXiv:2405.20304 [cs.CL] https://arxiv.org/abs/2405.20304

[22] Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Hena Ghonia, Rishika Bhagwatkar, Arian Khorasani, Mohammad Javad Darvishi Bayazi, George Adamopoulos, Roland Riachi, Nadhir Hassen, Marin Biloš, Sahil Garg, Anderson Schneider, Nicolas Chapados, Alexandre Drouin, Valentina Zantedeschi, Yuriy Nevmyvaka, and Irina Rish. 2024. Lag-Llama: Towards Foundation Models for Probabilistic Time Series Forecasting. arXiv:2310.08278 [cs.LG] https://arxiv.org/abs/2310.08278

[23] Chao Shang, Jie Chen, and Jinbo Bi. 2021. Discrete Graph Structure Learning for Forecasting Multiple Time Series. In *International Conference on Learning Representations*. https://openreview.net/forum?id=WEHSlH5mOk

[24] Zezhi Shao, Zhao Zhang, Fei Wang, and Yongjun Xu. 2022. 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* (Washington DC, USA) *(KDD '22)*. Association for Computing Machinery, New York, NY, USA, 1567–1577. doi:10.1145/3534678.3539396

[25] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2023. RoFormer: Enhanced Transformer with Rotary Position Embedding. arXiv:2104.09864 [cs.CL]

[26] Sainbayar Sukhbaatar, arthur szlam, Jason Weston, and Rob Fergus. 2015. End-To-End Memory Networks. In *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (Eds.), Vol. 28. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2015/file/8fb21ee7a2207526da55a679f0332de2-Paper.pdf

[27] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971 [cs.CL]

[28] NicolÃ¡s GarcÃa Trillos, Franca Hoffmann, and Bamdad Hosseini. 2021. Geometric structure of graph Laplacian embeddings. *Journal of Machine Learning Research* 22, 63 (2021), 1–55. http://jmlr.org/papers/v22/19-683.html

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention Is All You Need. arXiv:1706.03762 [cs.CL]

[30] Cristiano Villa and Francisco J. Rubio. 2018. Objective priors for the number of degrees of freedom of a multivariate t distribution and the t-copula. arXiv:1701.05638 [stat.ME] https://arxiv.org/abs/1701.05638

[31] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. 2020. 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* (Virtual Event, CA, USA) *(KDD '20)*. Association for Computing Machinery, New York, NY, USA, 753–763. doi:10.1145/3394486.3403118

[32] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph WaveNet for Deep Spatial-Temporal Graph Modeling. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 1907–1913. doi:10.24963/ijcai.2019/264

[33] Huaxiu Yao, Yiding Liu, Ying Wei, Xianfeng Tang, and Zhenhui Li. 2020. Learning from Multiple Cities: A Meta-Learning Approach for Spatial-Temporal Prediction. arXiv:1901.08518 [cs.LG] https://arxiv.org/abs/1901.08518

[34] Yuan Yuan, Jingtao Ding, Jie Feng, Depeng Jin, and Yong Li. 2024. 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* (Barcelona, Spain) *(KDD '24)*. Association for Computing Machinery, New York, NY, USA, 4095–4106. doi:10.1145/3637528.3671662

[35] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. 2020. GMAN: A Graph Multi-Attention Network for Traffic Prediction. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 01 (Apr. 2020), 1234–1241. doi:10.1609/aaai.v34i01.5477

[36] Cai Zhou, Rose Yu, and Yusu Wang. 2024. 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 (Proceedings of Machine Learning Research, Vol. 238)*, Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li (Eds.). PMLR, 2179–2187. https://proceedings.mlr.press/v238/zhou24a.html

[37] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 12 (May 2021), 11106–11115. doi:10.1609/aaai.v35i12.17325

[38] K. Zografos. 1999. On maximum entropy characterization of Pearson's type II and VII multivariate distributions. *J. Multivar. Anal.* 71, 1 (Oct. 1999), 67–75. doi:10.1006/jmva.1999.1824

A Appendix

A.1 Baselines

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

- GMAN [35]: The gating fusion mechanism redefines the spatio-temporal attention block.
- MTGNN [31]: Alternating use of graph convolution and temporal convolution modules.
- GTS [23]: The underlying graph structure is learned among multiple time series, and the corresponding time series is simultaneously predicted with DCRNN [14].
- STEP [24]: Adopts the graph wavenet model [32], integrated into the transformer backbone and pre-training scheme.
- FlashST [16], UniST [34]: The data distribution shift is learned by leveraging prompt-tuning.
- Lag-Llama [22]: They used known LLM backbones for unified probabilistic time-series forecasting.

To compare with the baselines, we adopt MAE, RMSE, and MAPE prediction error [16].

A.2 Proof of Equation 4

Let $f_d(\mathbf{x} \mid \boldsymbol{\mu}, \Sigma, \nu)$ be a multivariate t, of dimension d, with location vector $\boldsymbol{\mu}$, scale matrix Σ and ν degrees of freedom. The aim is to define an objective prior for the parameter ν. For simplicity in the notation, we will write $f_{d,\nu} = f_d(\mathbf{x} \mid \boldsymbol{\mu}, \Sigma, \nu)$, for $\nu = 1, \ldots, \nu_{\max -1}$, and $f_{d,\nu_{\max}} = N_d(\mathbf{x} \mid \boldsymbol{\mu}, \Sigma)$, with

$$N_d(\mathbf{x} \mid \boldsymbol{\mu}, \Sigma) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\},$$

where in this case $\boldsymbol{\mu}$ is the vector of means and Σ is the covariance matrix. The prior for ν here discussed depends on the Kullback–Leibler divergence between two multivariate densities. In particular, for $\nu = 1, \ldots, \nu_{\max -1}$, the prior is based on the Kullback–Leibler divergence between two multivariate t densities, which differ only in the number of degrees of freedom. The divergence between two d-variate t densities, $f_{d,\nu}$ and $f_{d,\nu'}$, is given by

$$\begin{aligned}
& D_{KL}(f_d(\cdot \mid \boldsymbol{\mu}, \Sigma, \nu) \mid\mid f_{d,}(\cdot \mid \boldsymbol{\mu}, \Sigma, \nu')) && (6)\\
= \;& D_{KL}(f_d(\cdot \mid \mathbf{0}, \mathbf{I}, \nu) \mid\mid f_d(\cdot \mid \mathbf{0}, \mathbf{I}, \nu')) \\
= \;& \int_{\mathbb{R}^n} f_d(\mathbf{x} \mid \mathbf{0}, \mathbf{I}, \nu) \log \frac{f_d(\mathbf{x} \mid \mathbf{0}, \mathbf{I}, \nu)}{f_d(\mathbf{x} \mid \mathbf{0}, \mathbf{I}, \nu')}\, d\mathbf{x} \\
= \;& \int_{\mathbb{R}^n} K(d,\nu)\left(1+\frac{\mathbf{x}^\top\mathbf{x}}{\nu}\right)^{-\frac{\nu+d}{2}} \log \frac{K(d,\nu)\left(1+\frac{\mathbf{x}^\top\mathbf{x}}{\nu}\right)^{-\frac{\nu+d}{2}}}{K(d,\nu')\left(1+\frac{\mathbf{x}^\top\mathbf{x}}{\nu'}\right)^{-\frac{\nu'+d}{2}}}\, d\mathbf{x} \\
= \;& \log\frac{K(d,\nu)}{K(d,\nu')} - \frac{\nu+d}{2}\mathbb{E}_{d,\nu}\left[\log\left(1+\frac{\mathbf{x}^\top\mathbf{x}}{\nu}\right)\right] + && (7)\\
& \frac{\nu'+d}{2}\mathbb{E}_{d,\nu}\left[\log\left(1+\frac{\mathbf{x}^\top\mathbf{x}}{\nu'}\right)\right], && (8)
\end{aligned}$$

where

$$K(d,\nu) = \frac{\Gamma\left(\frac{\nu+d}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)\sqrt{(\pi\nu)^d}},$$

and $\mathbb{E}_{d,\nu}$ represents the expected value with respect to $f_d(\cdot \mid \mathbf{0}, \mathbf{I}, \nu)$. More specifically,

$$\mathbb{E}_{d,\nu}\left[\log\left(1+\frac{\mathbf{x}^\top\mathbf{x}}{\nu}\right)\right] = \Psi\left(\frac{\nu+d}{2}\right) - \Psi\left(\frac{\nu}{2}\right),$$

where Ψ is the digamma function. For the second expectation in (6), we use Lemma 2 in [38] to obtain the following expression after a change of variables in terms of spherical coordinates

$$\begin{aligned}
\mathbb{E}_{d,\nu}\left[\log\left(1+\frac{\mathbf{x}^\top\mathbf{x}}{\nu'}\right)\right] \;=\; & K(d,\nu)\frac{\pi^{\frac{d}{2}}}{\Gamma\left(\frac{d}{2}\right)} \times \\
& \int_0^\infty \left(1+\frac{t}{\nu}\right)^{-\frac{\nu+d}{2}} t^{\frac{d}{2}-1} \log\left(1+\frac{t}{\nu'}\right) dt
\end{aligned}$$

which only requires one-dimensional numerical integration, regardless of the dimension d.

A.3 Visualizations

To better understand and assess our model visually, we present additional illustrations in this section. The visualization of long-term forecasting in zero-shot setting for Pems07(M) dataset is shown in Figure 7. Figure 7 is an instance of multivariate probabilistic forecasting in which the dark blue nodes are part of the k-hop extracted subgraph. A critical aspect of time-series prediction algorithms is their ability to capture volatility importance - the varying degrees of uncertainty and rapid changes in the data over time. By incorporating volatility measures, these algorithms can better understand periods of high traffic turbulence versus relative stability, leading to more nuanced and accurate predictions. Strada-LLM is also able to capture volatility existing in Pems07(M) dataset, which is shown in Figure 8.

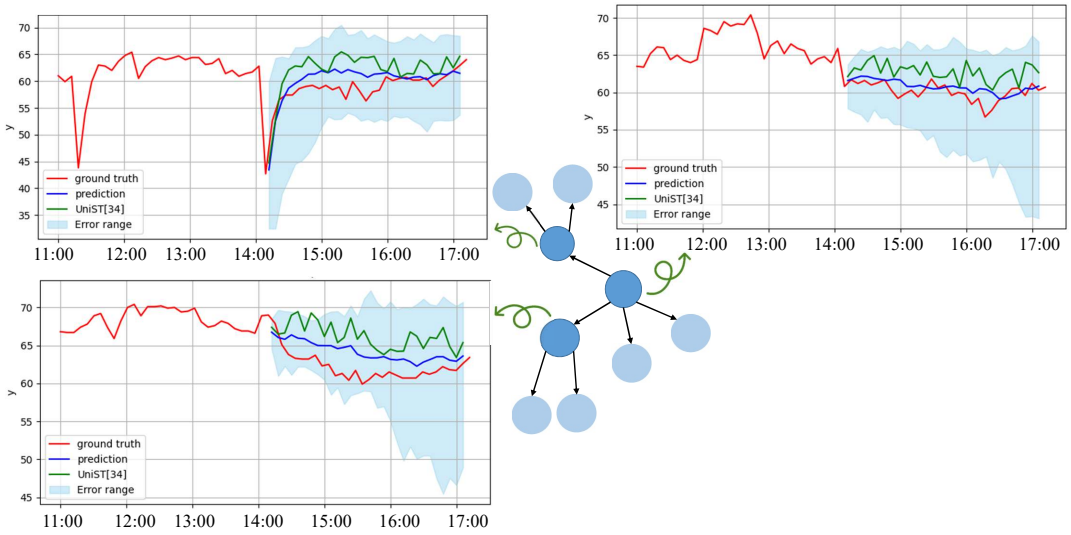
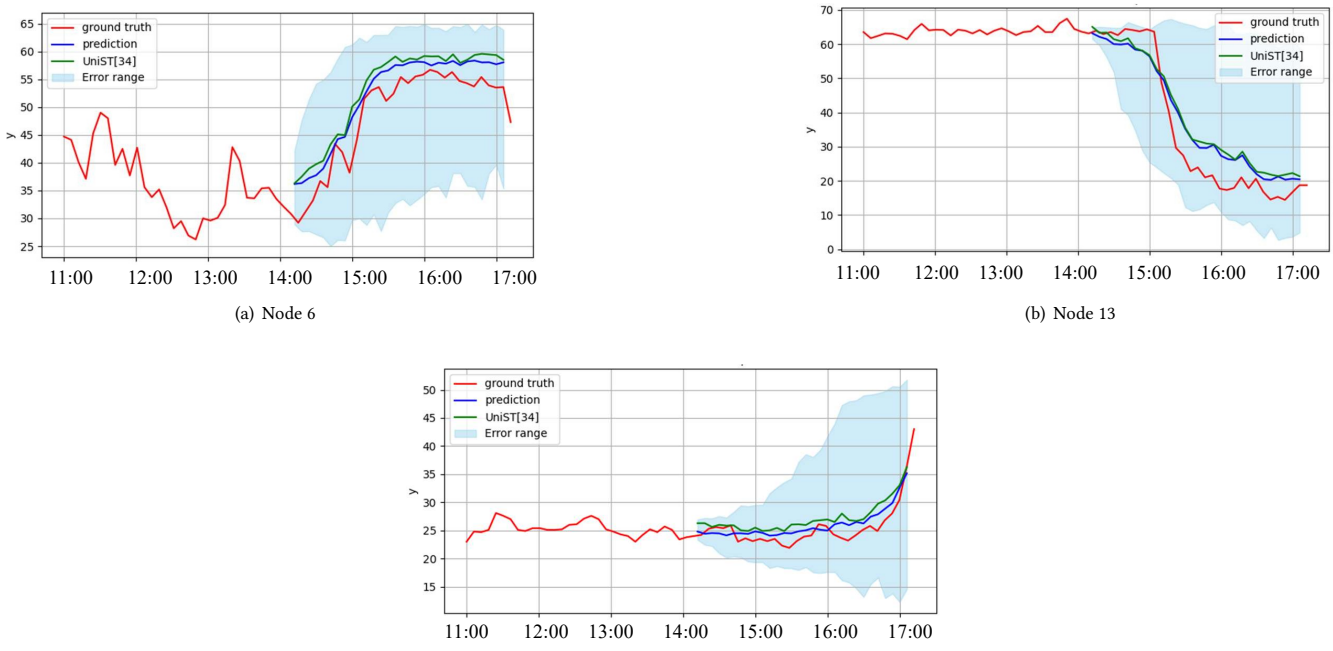


Figure 7: The visualization of k-hop multivariate zero-shot prediction for Pems07(M) dataset for long-term horizon



(a) Node 6

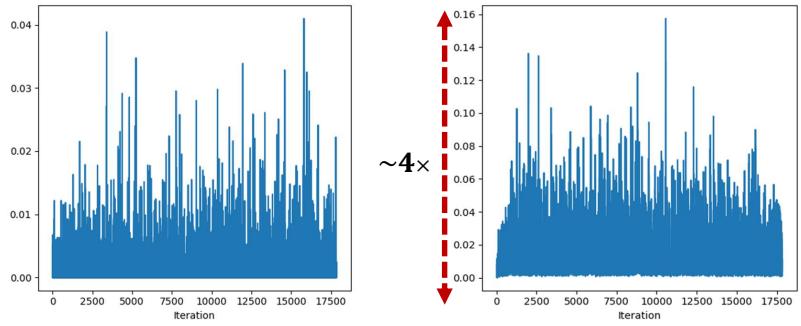
(b) Node 13

(c) Node 48

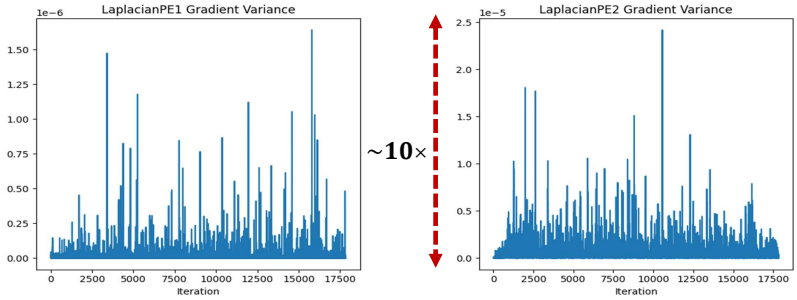
Figure 8: Strada-LLM traffic prediction in the capturing volatility scenarios for three (not-connected) different nodes exist in Pems07(M) dataset.

A.4 Gradient Analysis

As mentioned in section 3.3, the low-rank adaptation has not been utilised in the Graph Embedding module, which causes model performance degradation. To explain the reason, the norm and variance magnitude of the gradients of two MLP layers inside the module are plotted in Figure 9. As shown in Figure 9(b), according to the heterogeneous gradient backpropagation of these layers, e.g., one MLP's variance is 10 times the other MLP layer, and the adoption of the same low-rank adaptation causes performance degradation. Proposing an efficient low-rank adaptation in this use case will remain an ongoing research direction.



(a) Comparison of l_2 gradient norms between the first layer (left) and the second layer (right) across training iterations.



(b) Gradient σ^2 variance comparison between LaplacianPE1 (left) and LaplacianPE2 (right) layers during training.

Figure 9: Training dynamics comparison between two graph Laplacian positional encoding layers: gradient norms and variances over iterations.

A.5 Training Strategy

A.5.1 Augmentation Techniques. We adopted the time series augmentation techniques *FreqMix*[6] and *FreqMask* [6] to prevent over-fitting. Our training framework comprises two distinct stages: pre-training and fine-tuning. During pre-training, we leverage a diverse combination of datasets: D_{base} for general language understanding, D_{domain} for domain-specific knowledge, and D_{task} for task-oriented capabilities. These datasets are processed in a curriculum learning manner, starting with D_{base} to establish foundational knowledge, followed by D_{domain} to inject domain expertise, and finally D_{task} to enhance task-specific performance. To maintain balanced exposure across datasets while optimizing computational efficiency, we employ a dynamic batch construction strategy where each training batch B_t consists of samples drawn from multiple datasets with proportional representation: $\alpha_1\%$ from D_{base}, $\alpha_2\%$ from D_{domain}, and $\alpha_3\%$ from D_{task}, where $\sum_i \alpha_i = 1$. This balanced approach ensures the model develops robust general capabilities while maintaining strong performance on domain-specific tasks. In the fine-tuning stage, we exclusively focus on D_{task} with smaller learning rates and specialized optimization techniques to prevent catastrophic forgetting of pre-trained knowledge while adapting to target objectives.

A.5.2 Experimental Setup. 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 [13]. In the experiment, we follow the weight decay $1e-8$ and set the learning rate to 0.001 [13], 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.

Table 13: 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**

A.6 Rank Analysis

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 13. 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.

A.7 Perturbation Analysis and Robustness

To evaluate the robustness 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 10, 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.

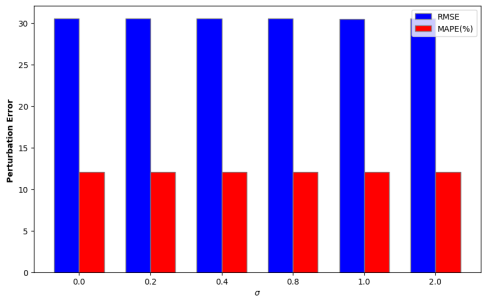


Figure 10: Evaluation metrics for the perturbation analysis. Different σ values were examined to test the model's robustness.

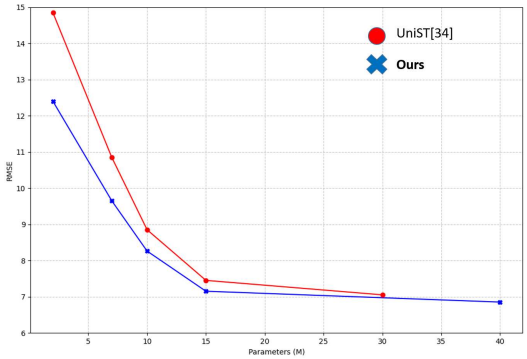


Figure 11: Strada-LLM scalability analysis compared to existing LLMs.

A.8 Scalability Comparison

Our empirical analysis reveals a nuanced relationship between model size and performance, characterized by diminishing returns beyond certain parameter thresholds. As illustrated in Figure 11, our model performs superior on the similar parameter capacity on the Crowd dataset.