Mobility and Context-Aware Pre-caching Strategy Using Spatial-Temporal Informer for Vehicular Service

Chenglong Wang, *Student Member, IEEE,* Jun Peng, *Senior Member, IEEE,* Lin Cai, *Fellow, IEEE,* Weirong Liu, *Member, IEEE,* Ziyu Zhao, Hu He, *Student Member, IEEE,* Zhiwu Huang, *Member, IEEE*

***Abstract*—With the rapid development of vehicle-to-everything technology, vehicular edge caching has emerged as a crucial component for managing frequently accessed content at the network's edge. However, due to vehicles' high mobility, it is challenging to determine where and which content needs to be cached. To address this issue, a mobility and context-aware pre-cache strategy is proposed to proactively pre-fetch and replace content in two steps. Firstly, by integrating the traffic features from vehicles and roads, a spatial-temporal informer-based model is designed to predict long-term vehicle trajectories. Subsequently, a proactive context-aware pre-cache strategy is proposed. By analyzing the context of different cache types, the required content can be further accurately estimated according to the cache type and workload. Extensive simulations based on real-world mobility scenarios are conducted to validate the performance of the proposed method. The results show that the proposed method can improve prediction accuracy and cache hit rate by 34.56% and 18.89%, and reduce mean response time and total energy cost by 6.1% and 2.65% compared to the existing pre-caching methods.**

***Index Terms*—Content pre-cache, Long-term mobility prediction, Context awareness pre-fetch, Vehicular edge network.**

I. Introduction

With the rapid development of vehicle-to-everything (V2X) technology, the emergence of vehicular content networks has ushered to connect vehicles, adjacent roadside units (RSUs), and remote content servers, serving as the backbone for intelligent transportation systems [1], [2]. Empowered by the vehicular content network, connected vehicles can have real-time access to a diverse range of content, facilitating advanced vehicular services such as streaming infotainment [3], driving assistance [4], and autonomous driving [5]. However, the growing demand for these advanced vehicular services requires real-time data retrieval and adjacent storage resources, and the low content access latency cannot be satisfied if the contents need to be retrieved from a remote content server. To ensure low access latency, effective vehicular caching management is crucial.

Developing a feasible vehicular caching strategy for vehicular content networks is challenging due to the vehicle mobility and dynamic content demand [6]. The high-speed movement of vehicles poses challenges to content pre-cache strategy design [7]. To ensure cache hit rate and adjacent accessibility, the content pre-cache strategy needs to determine where and which content should be pre-fetched based on the temporal dependencies of the current requested content and vehicle mobility. On the other hand, the limited storage capacity of RSUs poses difficulties in vehicular content caching replacement. To avoid storage wastage, the designed strategy needs to be aware of upcoming and leaving vehicles and dynamically update stored cache content based on the demands and locations of vehicles. As a result, considering both content pre-fetching and content replacement are needed.

Existing cache management methods primarily focus on content popularity, which leverages statistical analysis [8], request pattern recognition [9], and popularity prediction [10] to obtain content popularity for content pre-caching. However, different types of cached content exhibit distinct request probabilities, which don't always adhere to the popularity relationships dictated by Zipf law [11]. For instance, the next requested cache block will likely be a neighboring block rather than the most popular one for video streaming services. This degrades the performance of popularity-based cache strategies when multiple service types coexist.

Additionally, according to research [12], approximately 70% of caches in the remote content server are one-hit caches, which could be the content exclusive to a specific user. This requires designing a pre-caching strategy that not only considers the dependencies of required content but also is aware of vehicle mobility. Although some works have integrated convolutional neural networks, recurrent neural networks, and attention mechanisms to predict the mobility of vehicles, most of these studies focused on one-step-ahead predictions. The one-step-ahead prediction works are often insufficient to accommodate the computational overhead and the time to update cached content. As a result, the design of a cache strategy that considers long-term mobility prediction and the contextual dependencies and content types is still an open issue.

Chenglong Wang, Weirong Liu, and Hu He are with the School of Computer Science and Engineering, Central South University, Changsha, 410083, China (e-mail: 415389362@qq.com, frat@csu.edu.cn, summerki@csu.edu.cn).

Jun Peng is with the School of Electrical Information, Central South University, Changsha, 410083, China (e-mail: pengj@csu.edu.cn)

Lin Cai is with the Department of Electrical and Computer Engineering, University of Victoria, Victoria BC V8W 3P6, Canada (e-mail:cai@ece.uvic.ca).

Ziyu Zhao is with the School of Law and Criminal Justice, East China University of Political Science and Law, Shanghai, 200042, China, (e-mail: 2221110088@ecupl.edu.cn).

Zhiwu Huang is with the School of Automation, Central South University, Changsha, 410083, China (e-mail: hzw@csu.edu.cn).

(Corresponding author: Weirong Liu.)

Different from the existing state-of-the-art pre-caching schemes that solely rely on content popularity for determining pre-cached content and utilize short-term predictions to decide the pre-caching destination, this paper proposes a long-term mobility and context-aware pre-caching scheme. Specifically, a Spatial-Temporal Informer (ST-Informer) model is proposed to predict long-term vehicular mobility. Despite the increase in the number of model parameters associated with the proposed ST-Informer, the adoption of a probability-sparse self-attention mechanism enables a reduction in computational complexity while significantly enhancing the accuracy of predictions. Leveraging these prediction results, a proactive, context-aware content pre-caching approach is designed. This strategy is designed to not only mitigate prediction inaccuracies but also to dynamically pre-cache and replace various types of cached content in response to the contextual demands of connected vehicles. Compared to the state-of-the-art pre-caching methods, the proposed scheme demonstrates superior performance in adapting to diverse vehicular content.

The main contributions of this paper are summarized as follows:

- A novel long-term vehicle trajectory prediction model, Spatio-Temporal Informer, is proposed for pre-caching destination determination. Compared to the existing prediction models, the proposed ST-Informer can embed both spatial and temporal features into input data, enabling feature extraction of vehicle mobility patterns from both temporal and spatial dimensions and achieving the best prediction accuracy in different prediction lengths.
- To proactive pre-fetching required vehicular content, a mobility and context-aware pre-caching strategy is designed. Compared to the state-of-the-art pre-caching strategy, which is solely based on user mobility or content popularity, the proposed strategy can dynamically pre-cache and replace vehicular content in response to vehicle mobility, content popularity, and contextual demands.
- Extensive experiments are conducted on the realistic urban vehicle mobility scenario in Bologna city to verify the superiority of the proposed pre-caching strategy. The experiment results show that the designed ST-Informer can reduce mean square error by 34.56% compared to the state-of-the-art prediction methods, and the proposed pre-caching strategy can improve the cache hit rate by 18.89% and reduce mean response delay and energy cost by 6.1% and 2.65% compared to the existed caching strategies.

The rest of this paper is organized as follows. Related work is introduced in the next Section. The system model is presented in Section III. The designed ST-Informer is introduced in Section IV. The proactive context-aware pre-caching strategy is presented in Section V. The experiment results are presented in Section VI, followed by the conclusion in Section VII.

II. Related Work

Efficient content caching is crucial for enhancing communication reliability and reducing latency in vehicular networks. Over the past few years, numerous studies have been conducted on caching strategy design, which can be categorized into reactive caching and proactive caching.

Reactive caching involves making content caching decisions based on content popularity. Existing works have employed methods ranging from request content pattern analysis to popularity prediction for formulating reactive caching strategies. In [13], the hidden Markov model was used to predict content popularity based on the statistical analysis of content request characters. Based on the predicted content popularity, a popularity-based content caching method was designed for making caching decisions. Similarly, [14] introduced a QoE-driven caching method that integrates file popularity and user interest into caching update strategies. This approach employs deep reinforcement learning to address the QoE-driven RSU content caching problem effectively. By mining sequential patterns in content retrievals, a predictive edge caching method was proposed in [15] to make caching decisions adaptively according to the real-time content popularity. Some studies go a step further by integrating deep learning into popularity prediction for designing caching strategies. For example, a multi-agent reinforcement learning caching method was introduced in [16], which was capable of adapting to the diversity of RSU content popularity from both spatial and temporal perspectives. In [17], a clustering-based long short-term memory deep learning method was proposed for content popularity prediction. The predicted popularity was then used to inform caching decisions adaptively through the deep deterministic policy gradient method.

The drawback of reactive caching strategies is their focus on caching content without considering the data forwarding path. In practical content networks, nearly 70% of caches are one-hit caches [12], which are content exclusive to a specific vehicle. This means that solely considering popularity and neglecting vehicle mobility can decrease the cache hit rate and storage efficiency of RSUs. Therefore, reactive caching strategies may become ineffective in vehicular edge networks.

The proactive caching strategy considers the data forwarding path, fetching and updating content based on both content characteristics and user mobility. Some works incorporate vehicle mobility to design the mobility-aware caching strategy. For example, a federated learning-based mobility-aware proactive caching strategy was proposed in [18]. The proposed caching strategy integrates a mobility-aware cache replacement policy, which can update contents in response to the mobility patterns of vehicles. In [19], a federated deep reinforcement learning method was proposed for cooperative caching. The proposed cooperative caching strategy took the vehicle positions and velocities as the model input and used an autoencoder to predict popular contents for cooperative caching.

Over the past few years, many researchers have incorporated software-defined network technology and deep learning methods to design content caching methods [20]. To meet augmented reality service delay requirements in edge-assisted software-defined networks, a joint caching and computing resource reservation method was proposed in [21]. By formulating the caching problem as a long-term stochastic op-

timization problem, the proposed method can significantly reduce the overall resource consumption. In [22], spatio-temporal characteristics aware content caching is proposed for emergency content in software-defined vehicular networks. By predicting the content popularity, the proposed method can determine the most suitable content for caching in each RSU. Several works used deep learning methods to predict mobility for content caching. By adopting deep reinforcement learning, a traffic-aware content caching method was proposed in [23], which can optimize the content provider vehicle distribution across the vehicular social network. In [24], a proactive content caching strategy is proposed for urban vehicular networks. The proposed caching strategy uses a deep attention-based learning method to predict the next step vehicle trajectory and content popularity for content caching and dispatching.

Several works further incorporate the transformer into the caching strategy design. To extract the long-term dependencies, an attention-based vision transformer edge caching framework was proposed in [25], which can reduce the computational complexity of the caching strategy. In [26], a transformer-based actor-critic online centralized content caching method was proposed, which can minimize the task execution time subject to resource constraints. In [27], a generative pre-trained transformer model was proposed to cache content for AI-generated content services. By incorporating the in-context learning ability of the pre-trained transformer model, the resources can be allocated to meet the user's demand.

The major limitations of existing content-caching works lie in their focus on short-term mobility prediction and do not account for the context of the required content. Firstly, vehicle mobility prediction is a critical factor in determining the optimal pre-caching locations. Given the dynamic network conditions and the high mobility of vehicles, proactive caching mechanisms must make caching decisions within a limited prediction length. However, a short prediction length not only leaves insufficient time for running the prediction algorithms and content pre-caching but also results in an imbalanced workload due to a lack of long-term preview of future demands. Secondly, current research tends to pre-fetch content based solely on its popularity, overlooking the diversity of cache types and the contextual associations between contents. Consequently, there is an imperative requirement to devise a proactive caching strategy that integrates vehicle mobility with the contextual relevance of content across various cache types, which motivates this work.

III. System Model and Problem Formulation

In this paper, the edge content delivery system includes content servers and roadside units (RSUs). There is a set of vehicular content $[f_1, f_2, ..., f_n] \in F$, and the edge server is co-located with a content server, which stores all content F. Denote by $[r_1, r_2, ..., r_m] \in R$ the RSU set covered by the edge server, which is responsible for providing access to connected vehicles. Assume all the RSUs have the same capacity C to cache the content.

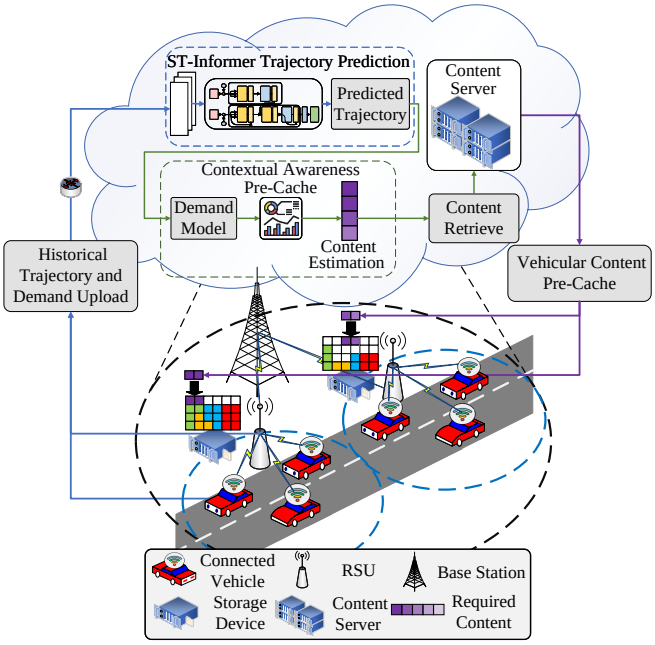


Fig. 1: The system overview of the proposed context-aware pre-caching strategy.

A. Content Request Model

This paper considers a 3-layer content storage model. Requested contents can be retrieved from the access RSU, adjacent RSUs, or the content server. The RSUs and connected vehicles, as well as other RSUs, are interconnected via 5G millimeter-wave wireless links [28]. Refer to the content delivery structure in [29], [30], each RSU is connected to the content server through a backhaul wired network link to retrieve the required vehicular content.

As depicted in Fig 1, if the vehicle requests content f_i and it is stored in the accessed RSU, the content will be directly transmitted to the vehicle, which is defined as the level-1 cache hit (L1 cache hit). Otherwise, if the requested content f_i is not in the accessed RSU, the content request will be sent to the one-hop adjacent RSUs. If the requested content is found in one of the adjacent RSUs, it will be transmitted to the vehicle via the access RSU, which is defined as level-2 cache hit (L2 cache hit). Otherwise, if the requested content cannot be found in the adjacent RSUs, it will be retrieved from the edge content server via the backhaul transmission link.

To bring the content closer to the connected vehicles, a content pre-fetch strategy is proposed in this paper. Once an access RSU receives a content request, a content pre-fetch request packet is sent to the edge content server. The packet consists of the vehicle features (such as speed and acceleration), road features (such as the vehicle density and average travel time), and the ID of the current request content. The content server will use these features to predict the mobility of vehicles and estimate the content that the vehicle may request in the next time slot. After that, the most possible content is pre-fetched to the corresponding RSU according to the mobility prediction and estimation result.

B. Cache Demand Model

In this paper, two different cache types are considered, popular cache (for web and other entertainment applications) and streaming cache (for real-time driving assistant application), respectively. Assume that each vehicle will request a popular cache with probability p_{on}^{pc} and a streaming cache with probability p_{on}^{sc} in each timeslot, and we have $p_{on}^{pc} + p_{on}^{sc} = 1$. Denote by $f_i^{pc} \in F^{pc}$ and $f_i^{sc} \in F^{sc}$ as the set of popular content and streaming content, and we have $F^{pc} + F^{sc} = F$.

$Popular\ Cache$: As many works have presented before, the popular cache follows the Zipf law [11]. Assuming that the total number of content is N, then the cache request probability of content f_i^{pp} can be given as

$$P(f_i^{pc}) = \frac{b}{(i^s \cdot H(M,s))} \tag{1}$$

where b is a normalization constant to ensure that the probabilities sum up to 1. s is the Zipf parameter, and $H(M,s)$ is the harmonic series of the M-th term.

Let Z be the link indicator matrix, whose element $z_{i,j}$ equals 1 when content f_i^{pc} can be directly linked to content f_j^{pc}, otherwise equals 0. Then the next step cache request probability can be represented as

$$P(f_j^{pc}|f_i^{pc}) = \sum_{k=0}^{n} \frac{P(f_j^{pc})}{z_{i,j} \cdot P(f_i^{pc})} \tag{2}$$

$Streaming\ Cache$: Similar to the popular cache content, the cache request probability of content f_i^{sc} is requested can be given as

$$P(f_i^{sc}) = \frac{b}{(i^s \cdot H(M,s))} \tag{3}$$

Unlike the popular cache, when content f_i^{sc} is requested, the next content is most likely to be f_{i+1}^{sc}. For simplicity, it is assumed $P(f_{i+1}^{sc}|f_i^{sc}) = 1$.

C. Communication Cost

As aforementioned, the requested contents can be retrieved from the access RSU, adjacent RSUs, or the edge content server. Denote the distance between the vehicle and access RSU as d_{v,r_i}, then the correspond transmission rate can be represented as

$$R_{v,r_i}(t) = B \cdot \log_2\left(1 + \frac{P_{r_i} h_{v,r_i} d_{v,r_i}^{-\delta}}{I + N}\right). \tag{4}$$

where P_{r_i} and h_{v,r_i} denote the transmit power of RSU and small-scale fading between the vehicle and access RSU, respectively. δ represents the path loss exponent. B denotes the bandwidth. N and I represent the power of noise and the interference between the vehicle and the RSU.

Assumed the content request packet size is $\|f^{\text{req}}\|$, if the requested content is stored at the access RSU, then the response time can be derived according to the (4) as

$$T_{\text{D}}(t) = \frac{\|f^{\text{req}}\|}{R_{v,r_i}^{UL}(t)} + \frac{\|f_k\|}{R_{v,r_i}^{DL}(t)}, \tag{5}$$

where $\|f_k\|$ denotes the content size. $R_{v,r_i}^{UL}(t)$ and $R_{v,r_i}^{DL}(t)$ are the uplink and downlink transmission rates between the connected vehicle and RSU. If the content is stored at a one-hop adjacent RSU r_j, the response time can be given as

$$T_{\text{H}}(t) = \frac{\|f^{\text{req}}\| + \|f_k\|}{R_{r_i,r_j}} + T_{\text{D}}, \tag{6}$$

where R_{r_i,r_j} is the transmission rate between adjacent RSU and access RSU.

It is assumed that each RSU is wired to the content server. If the requested content cannot be found in adjacent RSU, we can only find it from the edge content server, whose response time can be represented as

$$T_{\text{B}}(t) = T_{r_i,e} + \frac{2\|f^{\text{req}}\|}{R_{r_i,r_j}} + T_{\text{D}}, \tag{7}$$

where $T_{r_i,e}$ is the end-to-end propagation delay between access RSU and edge content server.

Since the transmission time over packet-switching networks can overlap with each other, the end-to-end delay from edge content to any RSU $T_{r_i,e}$ is approximated as a constant. In summary, the response time of content request can be rewritten as

$$T_v(t) = \begin{cases} T_{\text{D}}(t), & \text{if L1 Cache Hit,} \\ T_{\text{H}}(t), & \text{if L2 Cache Hit,} \\ T_{\text{B}}(t), & \text{Otherwise,} \end{cases} \tag{8}$$

where the L1 cache hit denotes the required cache is stored in the access RSU, and the L2 cache hit denotes the required cache is stored in a one-hop adjacent RSU.

D. Energy Cost

Similar to the response time, the energy cost is discussed in three different cases. If the cache hits the access RSU, the energy cost can be given as

$$E_{\text{D}}(t) = \frac{P_v \|f^{\text{req}}\|}{R_{v,r_i}^{UL}(t)} + \frac{P_r \|f_k\|}{R_{v,r_i}^{DL}(t)}. \tag{9}$$

where P_v and P_r are the transmission power of the vehicle and RSU. If the cache hits the adjacent RSU, the energy cost can be represented as

$$E_{\text{H}}(t) = \frac{P_r \|f^{\text{req}}\| + \|f_k\|}{R_{r_i,r_j}} + E_{\text{D}}, \tag{10}$$

Similarly, the energy cost to retrieve content from the edge content server can be given as

$$E_{\text{B}}(t) = E_{r_i,e} + \frac{2P_r \|f^{\text{req}}\|}{R_{r_i,r_j}} + E_{\text{D}}, \tag{11}$$

where $E_{r_i,e}$ is the backhaul transmission energy. In summary, the total energy cost can be rewritten as

$$E_v(t) = \begin{cases} E_{\text{D}}(t), & \text{if L1 Cache Hit,} \\ E_{\text{H}}(t), & \text{if L2 Cache Hit,} \\ E_{\text{B}}(t), & \text{Otherwise,} \end{cases} \tag{12}$$

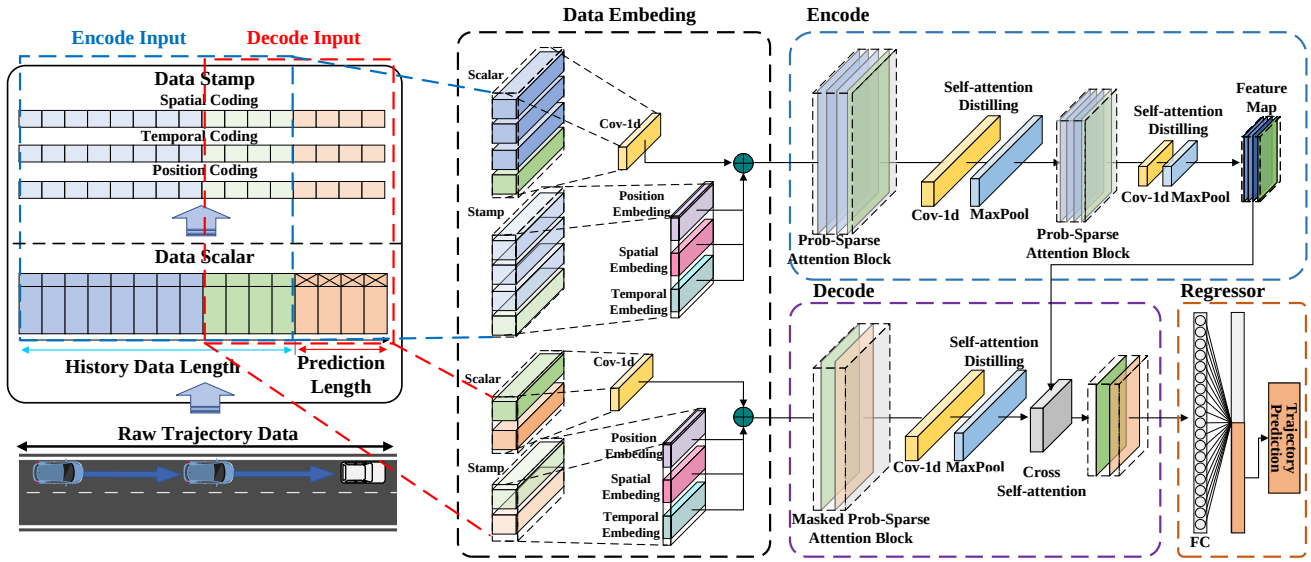


Fig. 2: The model structure of the proposed ST-Informer. The raw features are divided into two parts and pass the embedding layer as the input of the encode-decode layer. The encode-decode layer extracts hidden dependencies by multi-head probability sparse self-attention block to predict vehicle trajectory.

E. Problem Formulation

Denote $\mathbf{\Phi}$ the content storage indicator matrix, whose element $\phi_{i,j}$ equal to 1 if the content f_j storage in the RSU r_i, otherwise equal to 0.

For vehicle content request $q_v \in Q = \{r_i, f_j\}$, the content hit rate can be represented as

$$H(t) = \sum_{v \in V} \sum_{q_v \in Q} \frac{\overbrace{\phi_{r_i,f_j}^{q_v=\{r_i,f_j\}}(t)}^{\text{L1 Hit}} + \overbrace{\phi_{r_k,f_j}^{q_v=\{r_i,f_j\}}(t)}^{\text{L2 Hit}}}{\|Q\|} \tag{13}$$

To reduce the response time and energy cost, the key is to improve the cache hit rate. When the required content can be retrieved in an adjacent RSU, the response time and energy cost can be significantly reduced.

The objective of this paper is to optimize the content pre-cache strategy, which requires determining where and which vehicular content needs to be pre-cached. In the following section, we address this problem in two steps. First, a long-term trajectory predictions model ST-Informer is proposed to identify the pre-cached location. Then, a context-aware pre-caching method is designed to determine the specific content that needs to be pre-cached.

IV. ST-Informer for Trajectory Prediction

In order to design a feasible content pre-caching strategy, we need to know the connected vehicles' trajectories in the future to determine pre-cache location. As most previous works pre-fetch content based on the one-step-ahead forecast, which is not robust against forecast bias, in this work, a long-term mobility prediction method ST-Informer is designed.

In the proposed ST-Informer, we adopt the encode-decode structure of the Informer model with an upgraded data embedding method for the multi-vehicle trajectories prediction problem. As shown in Fig. 2, the proposed ST-Informer consists of data encoding, feature embedding, feature decoding, and output regression. In the following subsection, a detailed introduction to these components of the proposed ST-Informer will be provided.

A. Data Preprocessing

In this paper, we aim to predict all vehicle trajectories in considered areas for determining the pre-caching destination, so unlike other single object time-series prediction tasks directly using vehicle data as model input, data pre-processing is required.

TABLE I: The Input Data Features of ST-Informer

Features Type	Features Name	Description
Time	timestep	The simulation time step
Vehicle	V-ID	The ID of the vehicle
	Azimuth	The azimuth of the vehicle
	Speed	The speed of the vehicle
	Location	The coordinates of the vehicle
Road	R-ID	The ID of the road
	Travel time	The mean travel time of the road
	Max-speed	The maximum speed of the road
	Mean-speed	The mean speed of the road
	Occupancy	The occupancy percentage of the road

As shown in Table I, the raw data contains records of all vehicles during each timeslot, which includes features such as driving speed, azimuth, and current coordinates. Each record also has a unique ID and a lane ID indicating the corresponding vehicle and the road lane where the vehicle is located. To expand the feature dimension, we collect the road

lane features including travel time, maximum speed, mean speed, and occupancy of the road lane.

According to the lane ID and timestep in the vehicle data, the road features are incorporated as one data sample X_t. After that, the incorporated data are grouped by the vehicle ID as vehicle trajectory $\left[X_{t-L_h}, \cdots, X_t, \cdots, X_{t+L_p}\right]$, where $[X_{t-L_h}, \cdots, X_t]$ is the history data and $\left[X_t, \cdots, X_{t+L_p}\right]$ is the prediction data. For simplicity, the history data and prediction data are denoted as $X_{(t-L_h,t)}$ and $X_{(t,t+L_p)}$. L_h and L_p are the history data length and prediction data length.

B. Data Embedding

Due to the no recurrence and limited convolution used in the Informer model, the model lacks the local spatial and temporal dependencies of the vehicle trajectories. However, the spatio-temporal dependencies are beneficial for improving the prediction performance. For example, the traffic condition can be learned from the vehicle that recently traveled the same road.

To capture the global positional dependencies and local spatio-temporal dependencies of the vehicle trajectories, the information about spatial traffic conditions and temporal vehicle mobility trends need to be injected into the trajectory's time sequence. In this paper, three types of data encoding methods are used, positional encoding, spatial encoding, and temporal encoding.

Similar to the classical Transformer, the sine and cosine functions are used as positional encoding, which can be given as

$$\begin{aligned} \mathrm{PE}_{(pos,2i)} &= \sin\left(\mathrm{pos}/10000^{2\mathrm{i}/\mathrm{d}_{\text{model}}}\right) \\ \mathrm{PE}_{(pos,2i+1)} &= \cos\left(\mathrm{pos}/10000^{2\mathrm{i}/\mathrm{d}_{\text{model}}}\right) \end{aligned} \tag{14}$$

where pos and i is the position and dimension index of input X_t, and d_{model} is the size of embedding dimension.

Temporal encoding encompasses two aspects: the data index within the entire trajectory and its specific time index in seconds, minutes, and hours. Each encoding is normalized to the range of $[-0.5, 0.5]$, which be given as

$$\mathrm{TE} = \left[\frac{\mathrm{I(second)}}{59}, \frac{\mathrm{I(minute)}}{59}, \frac{\mathrm{I(hour)}}{23}, \frac{\mathrm{I}(X_t)}{\|X\|-1}\right] - \beta, \tag{15}$$

where $\mathrm{I}(\cdot)$ denotes the index of correspond feature.

Due to the inconsistent scales of different features in equation (15), it is necessary to normalize them to the consistent scales. However, since the $[0, 1]$ interval lacks a negative range, which impairs the performance during gradient computation, we further shift the normalized values by subtracting 0.5, mapping the interval to $[-0.5, 0.5]$. Consequently, β is defined as a normalized vector with elements set to 0.5.

For instance, if a vehicle enters the considered area at 08:01:00 and departs at 08:04:00, the spatial coding for a data sample collected at 08:03:30 would be $\left[\frac{8}{23}, \frac{3}{59}, \frac{30}{59}, \frac{150}{179}\right] - [\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}]$. In this way, each data sample can have a unique representation for temporal feature extraction.

Since we want to learn the hidden spatial dependencies from other vehicles and road lanes, the model needs to be aware of which vehicle the data sample comes from and which road section the vehicle is driving on. Consequently, we encode information regarding the road ID and vehicle ID to obtain a distinctive spatial representation. Similar to (15), the spatial encoding is given as

$$\mathrm{SE} = \left[\frac{\mathrm{I(V\text{-}ID)}}{\|V\|-1}, \frac{\mathrm{I(R\text{-}ID)}}{\|\mathrm{Road}\|-1}\right] - \beta, \tag{16}$$

Then, the data encoding is incorporated together as the input data stamp, which can be given as

$$X_t^{\mathrm{stamp}} = \mathrm{Conv1d(PE)} + \mathrm{Conv1d(TE)} + \mathrm{Conv1d(SE)}, \tag{17}$$

where $\mathrm{Conv1d}(\cdot)$ performs a 1-D convolutional filter with kernel width set as 3. Then data features X_t and data stamp X_t^{stamp} will be the input of the encoder and decoder.

C. Encoder

For the encoder, two multi-head probability sparse self-attention blocks are used to extract hidden spatio-temporal dependencies efficiently.

Given a time sequence $X_{(t-L_h,t+L_p)}$, the history data segment $X_{(t-L_h,t)}$ and its corresponding data stamp $X_{(t-L_h,t)}^{\mathrm{stamp}}$ will be the input of encoder. First, the data features are incorporated with corresponding data stamp as the encoder input by

$$X_{(t-L_h,t)}^{\mathrm{enc\_in}} = \mathrm{Conv1d}\left(X_{(t-L_h,t)}^{\mathrm{scalar}}\right) + X_{(t-L_h,t)}^{\mathrm{stamp}}, \tag{18}$$

where $X_{(t-L_h,t)}^{\mathrm{scalar}}$ is the normalized data from $X_{(t-L_h,t)}$.

Due to the high computational complexity when self-attention mechanisms handle long sequences, scholars have researched to reduce the computational complexity. One such improvement is probability-sparse self-attention. Compared to traditional self-attention, probability-sparse self-attention significantly reduces computational complexity by focusing only on a subset of time steps, rather than considering all steps equally. This makes it more efficient and scalable, especially for long sequences. By concentrating on the most relevant time steps, it filters out noise and less important information, leading to more accurate predictions. Therefore, the proposed ST-Informer adopts probability-sparse self-attention to extract the spatial-temporal dependencies.

Then the multi-head probability sparse self-attention mechanism is used to extract hidden spatio-temporal dependencies. The encoder input $X_{(t-L_h,t)}^{\mathrm{enc\_in}}$ is represented as $\mathbf{Q} = \mathbf{W}^q X_{(t-L_h,t)}^{\mathrm{enc\_in}}$, $\mathbf{K} = \mathbf{W}^k X_{(t-L_h,t)}^{\mathrm{enc\_in}}$, and $\mathbf{V} = \mathbf{W}^v X_{(t-L_h,t)}^{\mathrm{enc\_in}}$ by multiply three trainable weight matrices. After that, the attention can be calculated by

$$\mathcal{A}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}\left(\frac{\bar{\mathbf{Q}}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}, \tag{19}$$

where $\bar{\mathbf{Q}}$ is a sparse matrix only contains the Top-u queries under the sparsity measurement $\bar{M}(\mathbf{q}_i, \mathbf{K})$ proposed in [31]. By selecting sparse Top-u to calculate the dot-product pairs $\bar{\mathbf{Q}}$,

the time and space complexity can be significantly reduced. The sparsity measurement function can be given as

$$\bar{M}\left(\mathbf{q}_i, \mathbf{K}\right) = \max_j \left\{ \frac{\mathbf{q}_i \mathbf{k}_j^\top}{\sqrt{d}} \right\} - \frac{1}{L_K} \sum_{j=1}^{L_K} \frac{\mathbf{q}_i \mathbf{k}_j^\top}{\sqrt{d}}, \tag{20}$$

where L_K denotes the size of $\mathbf{K}$, and $\mathbf{q}_i$, $\mathbf{k}_i$ are the i-th row in $\mathbf{Q}$ and $\mathbf{K}$.

According to the attention value calculated by (19) and (20), the multi-head attention can be obtained as

$$\mathrm{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{Concat}\left(\mathcal{A}_1, ..., \mathcal{A}_i\right) \mathbf{W}_O, \tag{21}$$

where $\mathbf{W}_O$ is the trainable weight of the attention head.

For simplicity, $[\cdot]_{\mathcal{A}}$ is denoted as the calculation of multi-head probability sparse self-attention shown from (19) to (21). Then, the procedure from the k-th multi-head prob-sparse self-attention block to the k+1-th block can be expressed as

$$X^{k+1}_{(t-L_h,t)} = \mathrm{MaxPool}\left(\mathrm{ELU}\left(\mathrm{Cov1d}\left([X^k_{(t-L_h,t)}]_{\mathcal{A}}\right)\right)\right), \tag{22}$$

where $\mathrm{MaxPool}(\cdot)$ is the Max-Pooling function with a stride equal to 2, and $\mathrm{ELU}(\cdot)$ is the ELU activation function. The final multi-head probability sparse self-attention block output will be sent to the Decoder as the feature map.

D. Decoder

For the decoder, one masked multi-head probability sparse self-attention block is used to map the input and extracted feature map.

Considering the generative-style of the transformer, which requires a start token for dynamic decoding [32], a shorter sequence slice of the encoder $X^{\mathrm{dec\_in}}_{(t-L_p,t)}$ will be part of the input of the decoder. The input of the decoder can be represented as

$$X^{\mathrm{dec\_in}}_{(t-L_p,t+L_p)} = \mathrm{Concat}\left(X^{\mathrm{dec\_in}}_{(t-L_p,t)}, X^{\mathbf{0}}_{(t,t+L_p)}\right), \tag{23}$$

where $X^{\mathrm{dec\_in}}_{(t-L_p,t)}$ is performed as a start token for generating prediction results, and $X^{\mathbf{0}}_{(t,t+L_p)}$ is a placeholder for the prediction result, whose elements' values are set to 0. To avoid the future data involved in attention computing, masked multi-head attention is applied in self-attention computing by setting masked dot-products to $-\infty$.

After the calculation of the masked multi-head self-attention block by (22), the output will be processed by the cross multi-head self-attention block for feature mapping. Then, a fully connected layer is used for regression, and only the second half of the output vector will be used in the loss function calculation.

The whole training process is given in Algorithm 1. To train the proposed ST-Informer, the dataset is divided into the training set, validation set, and testing set based on vehicle ID, which consists of 70%, 10%, and 20% of vehicles' trajectory respectively. The MES loss function is used for back-propagation, which can be given as

$$Loss = -\sum_{i=t}^{t+L_p} \frac{(y_i - \hat{y_i})^2}{L_P}, \tag{24}$$

where y_i and $\hat{y_i}$ are the true locations and the predicted locations of the vehicle.

Algorithm 1: The training process of ST-Informer

Data: Training set and Validation set.
Output: Trained ST-Informer model

$epoch \leftarrow 0$, $loss\_indicator \leftarrow 0$
while $epoch \leq 20$ **do**
  $epoch \leftarrow 0$
  while $batch \leq \lceil |Training| / 64 \rceil$ **do**
    $batch \leftarrow 0$.
    Use one batch of training data for model forward propagation to calculate the probability vector $\hat{y}$.
    Calculate the loss based on loss function (24), and back-propagate the loss value for parameter updating.
    $batch \leftarrow batch + 1$.
  while $batch \leq \lceil |Validation| / 64 \rceil$ **do**
    $batch \leftarrow 0$.
    Use one batch of validation data for model forward propagation and calculate the validation loss val_loss based on loss function (24).
    $batch \leftarrow batch + 1$.
  if $val\_loss < loss\_indicator$ **then**
    $loss\_indicator \leftarrow val\_loss$ Save and update the model parameters as a trained model.
  else
    continue.
  $epoch \leftarrow epoch + 1$.

V. Context-Aware Content Pre-Cache Strategy

For vehicular content pre-cache strategy, two issues need to be addressed: (i) When and where to pre-fetch the required cache block based on the prediction results; (ii) How to update the stored cache to avoid wasting storage space and better cater to user content requests. To address these challenges, we will introduce the proposed context-aware pre-caching strategy for cache pre-fetch and cache replacement.

The performance of the pre-caching strategy is highly related to the accuracy of the trajectory prediction. As aforementioned, the long-term vehicle trajectory prediction can be obtained using the proposed ST-Informer method. In each timeslot, the proposed ST-Informer method will predict n-timeslot future data. To fully utilize prediction results, the prediction result can be divided into three parts denoted as pre-fetch initialization part, short-term preview, and long-term preview.

The pre-fetch initialization part corresponds to the pre-fetch initialization phase, which involves time costs associated with cache request transmission, obtaining vehicle trajectory prediction results, and executing pre-fetching. Beyond the initialization phase, the prediction results are further divided

into two parts with equal length: short-term previews and long-term previews. Short-term previews will be utilized to inform pre-fetch decisions, while long-term previews provide insight into future load trend changes and will serve as a reference for cache replacement decisions.

A. Contextual-Aware Content Pre-fetch

As aforementioned, prediction errors are inherent in any prediction method, potentially resulting in caches being pre-fetched to the wrong RSU. To mitigate the performance degradation caused by prediction errors, the iterative error correction method is proposed to improve the prediction accuracy.

The main idea of iterative error correction is that we aim to correct prediction errors by using results from predictions made at different timeslots. For example, when seeking the pre-cache RSU location in the $t+k$ timeslot, we choose the most frequently predicted RSU among predictions made at each timeslot from t to $t+k-1$ as the final pre-fetch RSU.

Denote $\hat{y}_t^{t+k}$ as the $t+k$ timeslot pre-fetch RSU predicted at t timeslot, then the target RSU can be obtained by

$$y_{t+k} = \arg\max \mathrm{Count}([\hat{y}_t^{t+k}, \hat{y}_{t+1}^{t+k}, \ldots, \hat{y}_{t+k-1}^{t+k}]) \quad (25)$$

where $\mathrm{Count}(\cdot)$ represents the number of occurrences of the predicted RSU in the different timeslots. k denotes the pre-fetch delay, which involves time costs associated with request transmission, obtaining vehicle trajectory prediction results, and executing pre-fetching.

Note that iterative error correction is effective when there are fewer dependencies in prediction at different timeslots, and prediction $k-1$ slots earlier has the similar performance as the most recent one. In transformer-based methods, the self-attention mechanism does not inherently prioritize the order of elements in a sequence [33], allowing it to dynamically select features from various timeslots for prediction. Consequently, predictions at different slots exhibit reduced dependency. The results of the experiment in [31] also show that the accuracy of the Informer prediction is hardly affected by an increase in prediction length. This observation strongly supports the adoption of our iterative error correction method.

By adopting iterative error correction, we can determine where the cache needs pre-fetch. Next, we will decide which cache blocks need to be pre-fetched. According to the prediction result using (25) and the pre-fetch delay, the required cache estimation is performed.

In this paper, two types of caches are considered: popular cache and streaming cache. The popular cache corresponds to the non-real-time entertainment cache, which can be repeatedly requested by different vehicles. The streaming cache corresponds to one-time real-time computing caching, which will not be requested repeatedly. Since the streaming cache is accessed sequentially, the cache requested after k steps is f_{i+k}^{sc}.

For popular cache, the pre-fetch cache can be estimated by the k-step Markov chain. Assume the pre-fetch delay is k, the k-step cache request probability can be calculated by

$$v_k = v_0 \cdot \mathbf{D}^k, \quad (26)$$

where v_0 represents the initial state, which indicates the cache requested at the initial state. $\mathbf{D}$ denotes the state transition matrix, whose element can be calculated by (2). The element of v_k indicates the probability of each cache likely to be requested after k steps. The cache with the highest probability in v_k is selected as the pre-fetch cache.

Algorithm 2: Context-Aware Content Pre-fetching

Input: Prediction result $\hat{y}$, Current request cache f^{req}, pre-fetch delay k

Output: pre-fetch cache and destination

```
Perform iterative error correction based on (25) to
  obtain the pre-fetch destination y_{t+k}.
if f^req is popular cache then
    Calculate state transition matrix D based on (2).
    Select the highest probability cache as pre-fetch
      cache according to (26).
else
    Select f_{i+k}^{sc} as pre-fetch cache.
```

In summary, the cache pre-fetch process is given in Algorithm 2. Firstly, the prediction results from each past timeslot are collected for iterative error correction to decide the pre-fetch destination RSU. Subsequently, considering the current requested cache type and the pre-fetch delay, the content requested after the k-step will be estimated.

B. Adaptive Dual-Cache Replacement

Due to the limited storage capacity, the cache pool of RSU needs to be updated dynamically according to the real-time condition. In this paper, an adaptive dual-cache replacement method is designed considering long-term load trends and load balance.

To ensure real-time performance for the streaming cache, we will design the cache replacement strategy in each timeslot based on the pre-fetch decision. The proposed dynamic dual-cache replacement method is shown in Algorithm 3.

In each timeslot, we divide the cache pool into two segments to store the popular cache and streaming cache separately and use n to adjust the number of stored popular caches. The technique to determine the value of n is as follows: when there is available space in the RSU cache pool, increase n based on the pre-fetch decision while ensuring the requirements for streaming cache are satisfied. When the cache space is limited, prioritize streaming cache. In the worst-case scenario, only store the top $n_{\min}$ most popular caches, ensuring that the probability of these caches being requested is greater than threshold ε. In summary, the stored popular cache number n can be given as

$$n = \max\left\{n_{\min}, (C - |f^{\mathrm{sc}}|)\right\}, \quad (27)$$

where $n_{\min}$ is the minimal stored popular cache number, and it follows $\sum_{i=0}^{n_{\min}} P(f_i^{pc}) \leq \varepsilon$. $P(f_i^{pc})$ is the content request possibility, which can be calculated by equation (1).

Algorithm 3: Adaptive Dual-Cache Replacement

Input: Prediction result $\hat{y}$, pre-fetch popular cache, pre-fetch streaming cache
Output: Cache replacement strategy

Determine the stored popular cache number n of RSU based on the short-term pre-fetch cache.
if $(C-n) \geq \|f^{sc}(t)\|$ **then**
    Update streaming caches by using the MRU strategy.
else
    Update $C-n$ streaming caches by using the MRU strategy.
    update the remaining cache to the adjacent RSU if there is space available.
if $n \geq \|f^{pc}(t)\|$ **then**
    Update popular cache by using the LRU strategy.
else
    Update n popular caches by using the LRU strategy.
    update the remaining cache to the adjacent RSU if there is space available.

The cache pool capacity of RSU is denoted by C, and the number of requests of popular cache and streaming cache in time slot t are $\|f^{pc}(t)\|$ and $\|f^{sc}(t)\|$, whose value can be obtained by the proposed pre-fetch algorithm introduced before. Similar to the cache pre-fetch design, cache replacement will have different update strategies based on the type of the cache. Since the streaming cache is a highly real-time, one-time cache, we prioritize updating the pre-fetched streaming cache and use the Most Recently Used (MRU) strategy to replace the already-hit streaming cache in each timeslot. For popular cache, the Least Recently Used (LRU) strategy is used to replace the cache pool. In cases of insufficient cache space, the remaining pre-fetch cache will be stored in the cache pool of adjacent RSUs if they have available space.

VI. Experiment Result

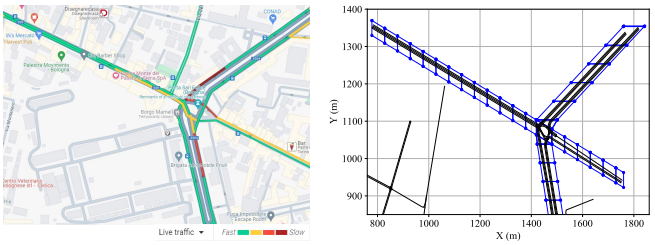
In this part, experiments are conducted to verify the performance of the proposed content pre-cache method. First, the prediction accuracy of the proposed ST-informer is compared with existed prediction models. Then, the performance of the designed pre-cache method is compared with state-of-the-art methods in terms of cache hit rate, response time, and energy cost.

A. Experiment Settings

1) Scenario Settings: The proposed method is evaluated in the city of Bologna by using the Simulation of Urban MObility (SUMO). The mobility dataset is collected from the "Real-World Bologna" [34], which covers a portion of the inner city of Bologna spanning an area of 1500 × 1800 square meters. The data samples of Real-World Bologna were collected from 8,779 vehicles during rush hour from 8 am to 9 am.

TABLE II: The Simulation Setting

Parameter	Value
The number of roads	15
The number of RSUs	31
The coverage of RSUs	50 m
The size of popular cache	5 MB
The size of streaming cache	5 MB
The size of the cache request packet	10 KB
The maximum transmit power of vehicles	23 dbm
The maximum transmit power of RSUs	29 dbm
The background noise power	-104 dbm



(a) Real-world map (b) Road structure in SUMO

Fig. 3: The real-world map and the simulation scenario at the city center of Bologna around the Porta San Felice. The selected area includes 15 roads with a total number of 32 lanes, which is covered by 31 RSUs.

In our experiments, the most congested area in Bologna is selected to verify the performance of the proposed method. The selected area at the city center of Bologna around the Porta San Felice, which includes 15 roads with a total number of 32 lanes as shown in Fig. 3(a). As shown in Fig. 3(b), the selected roads are divided into 31 road segments with a length of 50 meter, each covered by an RSU located in the center of the road segment. In this case, a total of 507,592 data samples are divided into three subsets: 355,314 data samples are used as training data, 50,759 data samples are used as validation data, and the remaining data samples are used as testing data.

2) Simulation Setting: The covered vehicles access the RSU through the NLOS mm-wave channel under the C-V2X wireless communication standard. The maximum transmission power of the vehicle and RSU are set to 23 dbm and 29 dbm respectively. The background noise power refers to the thermal noise power spectral density in [35]. For simplicity, the maximum capacity of the RSU to store cache blocks is set to 30, and the size of both the streaming cache and the popular cache is set to 5 MB. Other detailed settings are given in Table II.

3) Neural Network Setting: The neural network settings for the proposed ST-Informer are outlined in Table III. The encoder of the ST-Informer comprises 2 prob-sparse attention blocks, while the decoder contains one masked prob-sparse attention block. The attention head number of the prob-sparse attention block is set to 8, equivalent to the input feature number. The input lengths for the encoder and decoder are set to 96 and 48, respectively, with a prediction length of 24. The dropout probability for each attention block is set to 0.05. For training the ST-Informer, a step learning rate is employed,

with the initial learning rate set to 1e-4 and reduced by half every 4 epochs. The training epoch is set to 20, and the batch size is 32.

TABLE III: The Hyperparameters Setting

Parameter	Value
The feature number of encoder / decoder	8 / 8
The sequence length of encoder / decoder	96 / 48
Prediction length	24
The number of encoder / decoder layers	2 / 1
The number of attention head	8
Training epoch	20
Batch size	32
Learning rate	1e-4
Optimizer	Adam
StepLR decay	4 epochs
StepLR gamma	0.5
Drop-out probability	0.05

B. Performance Comparison of Trajectory Prediction

In this subsection, the mobility prediction performance of the proposed ST-informer is evaluated. Firstly, the compared methods and performance metrics are introduced. Subsequently, the performance comparison between the proposed ST-informer and other state-of-the-art methods is presented.

1) Performance Metric: To assess the predictive accuracy of our proposed prediction method, the following performance metrics are utilized:

- Mean Absolute Error (MAE):

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$$

- Mean Square Error (MSE):

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

- Root Mean Square Error (RMSE):

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

- Mean Absolute Percentage Error (MAPE):

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

2) Compared Method: We compared the proposed ST-informer with three state-of-the-art deep learning methods.

- ESN-LSTM: an echo state long short-term memory network adopted in [36], incorporating echo state network and LSTM to predict user mobility.
- DLMV: a deep learning-based mobile vehicle trajectory prediction method proposed in [37]. The proposed DLMV method is an autoencoder structure, which extracts hidden mobility patterns from vehicle sensing data for mobility prediction.
- Informer: an efficient generative style transformer-based method proposed in [31], which is known as the most suitable tool for long sequence time-series forecasting problems. The proposed Informer replaces the original canonical self-attention with the prob-sparse self-attention mechanism, which can enhance the prediction capacity with less time complexity.

3) Prediction Accuracy Comparison: In this subsection, we compare the proposed ST-Informer with three state-of-the-art methods in terms of training performance and prediction accuracy.

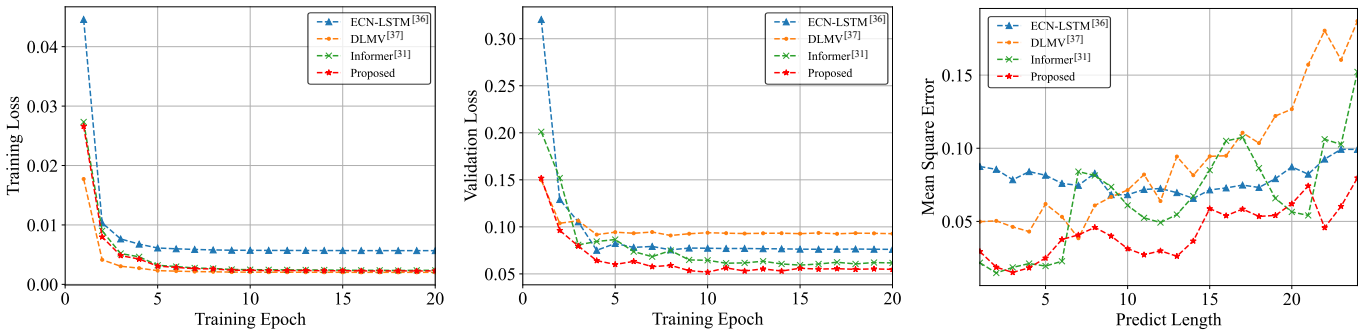
First, the performance during the training process is compared in Fig. 4. The training and validation loss are presented in Fig. 4(a) and Fig. 4(b), respectively, and the corresponding per-slot mean square error in the testing set is shown in Fig. 4(c).

As depicted in Fig. 4(a) and Fig. 4(b), all methods can converge within 5 training epochs. DLMV, Informer, and the proposed ST-Informer achieve almost the same lowest training loss during the 20 training epochs. Considering validation loss, the proposed ST-Informer achieves the lowest validation loss. It means that the proposed ST-Informer achieves the best prediction accuracy among the compared methods.

The corresponding per-slot prediction mean square error in the testing set is provided in Fig. 4(c). For the next 6-slot prediction, ECN-LSTM has the highest mean square error, and DLMV has the second-highest mean square error. Meanwhile, the Informer and the proposed ST-Informer achieve almost the same lowest mean square error during the next 6-slot prediction. Subsequently, all methods display increasing trends from the next 6-slot to the next 18-slot prediction. Among the shown methods, Informer and DLMV experience significant increases as the prediction length extends. In contrast, the proposed ST-Informer maintains the lowest mean square error.

From the next 18-slot to the next 24-slot, the mean squared error of Informer exhibits slight fluctuations and achieves the second-lowest mean square error. This is attributed to the self-attention mechanism not requiring the chronological dependencies of input time series. By employing the self-attention mechanism, Informer and the proposed method can adaptively select important hidden correlations from the input time series, enabling better prediction accuracy in long-term predictions. Similar fluctuations also exist in the proposed ST-Informer. By incorporating the spatio-temporal embedding, the proposed ST-Informer achieves the lowest mean square error during the next 24-slot prediction.

To further investigate prediction accuracy, the average performance metrics for different prediction lengths are provided in Table IV. In the next 6-slot prediction, Informer and the proposed method exhibit similar performance and achieve better prediction accuracy compared to other methods. Informer has the lowest mean square error and root mean square error, while the proposed ST-Informer has the lowest mean absolute error and mean absolute percentage error. Subsequently, the proposed ST-Informer demonstrates the best prediction accuracy in the 12-slot, 18-slot, and 24-

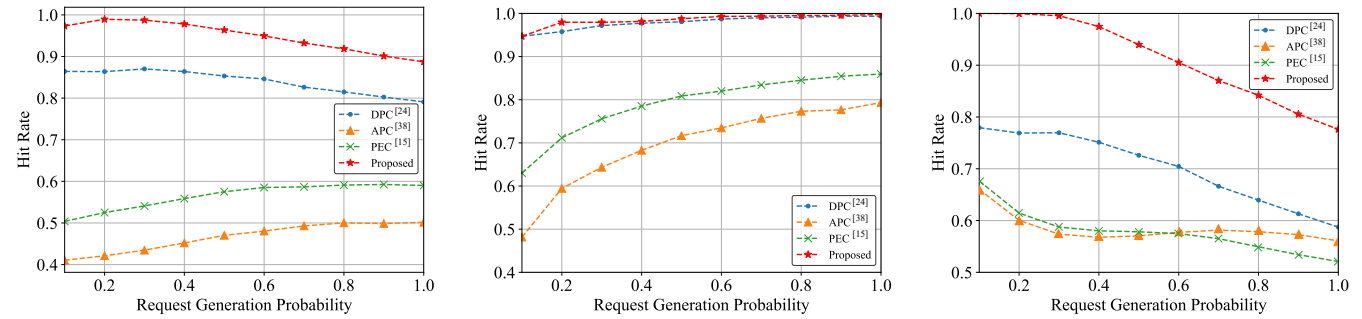


(a) Training loss (b) Validation loss (c) Testing MSE

Fig. 4: The performance comparison with three state-of-the-art methods during the training process. (a) The training loss comparison during 20 training epochs; (b) The validation loss comparison during 20 training epochs; (c) The per-slot testing mean square error comparison for the next 24 slots.

TABLE IV: The Average Performance Metric in Different Prediction Lengths

	Performance Metric for Next 6-slot				Performance Metric for Next 12-slot				Performance Metric for Next 18-slot				Performance Metric for Next 24-slot			
Method	MAE	MSE	RMSE	MAPE	MAE	MSE	RMSE	MAPE	MAE	MSE	RMSE	MAPE	MAE	MSE	RMSE	MAPE
ECN-LSTM [36]	0.1854	0.0821	0.2856	0.4100	0.1822	0.0776	0.2783	0.3992	0.1837	0.0755	0.2745	0.3569	0.1915	0.0791	0.2808	0.3405
DLMV [37]	0.1311	0.0507	0.2249	0.3340	0.1415	0.0573	0.2381	0.3274	0.1606	0.0704	0.2621	0.3103	0.1851	0.0917	0.2949	0.2968
Informer [31]	0.1077	**0.0197**	**0.1401**	0.5807	0.1421	0.0433	0.2034	0.5107	0.1611	0.0569	0.2286	0.4522	0.1716	0.0651	0.2449	0.4079
Proposed	**0.1044**	0.0240	0.1530	**0.2887**	**0.1122**	**0.0299**	**0.1707**	**0.2543**	**0.1226**	**0.0359**	**0.1860**	**0.2466**	**0.1353**	**0.0426**	**0.2018**	**0.2302**



(a) Overall Hit Rate (b) Popular Cache Hit Rate (c) Streaming Cache Hit Rate

Fig. 5: The cache hit rate comparison with three state-of-the-art caching strategies. (a) The overall hit rate comparison includes popular cache and streaming cache; (b) The popular cache hit rate comparison; (c) The streaming cache hit rate comparison.

slot predictions. In the longest prediction length, the proposed ST-Informer reduces by 21.15% for MAE, 34.56% for MSE, 17.59% for RMSE, and 22.43% for MAPE compared to the second-best-performing method.

In summary, the experiment shows that the proposed ST-Informer exhibits similar performance with the Informer in the next 6-slot prediction, and outperforms other state-of-the-art methods in all other longer-term trajectory prediction.

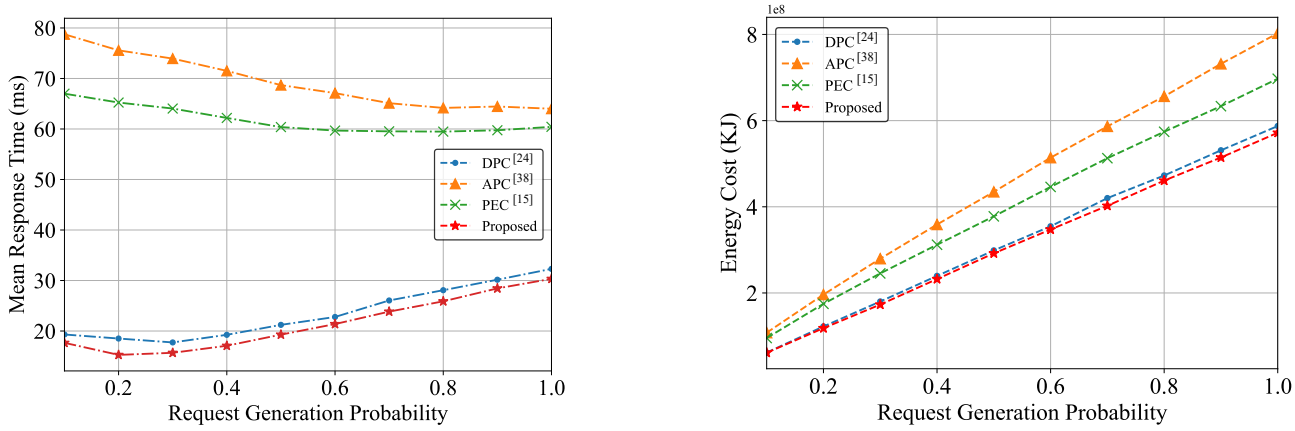
C. Performance Comparison of Pre-Caching Strategy

According to the prediction results, the performance of the pre-caching strategy in terms of cache hit rate, mean response time, and energy cost are evaluated. Similarly, the performance metrics and compared methods are introduced.

1) Performance Metric: To evaluate the proposed pre-caching strategy, the cache hit rates for the popular cache and the streaming cache will be individually presented, along with the aggregate overall cache hit rate for both types. Additionally, the mean response time and energy cost are used for performance evaluation. The performance metric is given as follows:

- Cache Hit Rate: The overall cache hit rate measures the percentage of the requested popular and streaming cache that is successfully retrieved from the RSUs without the need to access the cache server through the backhaul link, calculated by (13).
- Mean Response Time: The Mean Response Time measures the average time taken for a cache request to be fulfilled by retrieving the required cache block, calculated by (8).
- Energy Cost: The Energy Cost metric assesses the energy consumption associated with cache retrieving, calculated by (12).

2) Compared Method: In this part, we evaluate the proposed pre-caching strategy with three state-of-the-art proactive edge caching methods. Two of these methods rely solely on



(a) Mean Response Time

(b) Energy Cost

Fig. 6: The mean response time and energy cost comparison with three state-of-the-art caching strategies.

content popularity, while the remaining one incorporates both popularity and mobility two aspects when making content caching decisions.

- APC: an adaptive priority-based content caching method proposed in [38]. The APC method utilizes a Bayesian network to predict content popularity for cache pre-fetching and employs a priority-based least recently used (LRU) strategy for cache block replacement.
- PEC: a predictive edge content caching method proposed in [15]. The PEC method predicts content popularity for cache pre-fetching. The PEC adopts the LRU2 method for cache replacement, which evicts caches based on the least recently used strategy from the previous two requests.
- DPC: a dynamic proactive content caching method proposed in [24]. The DPC method predicts vehicle trajectory and content popularity for content pre-fetching. The DPC uses the basic LRU strategy for cache replacement.

3) Performance Evaluation: In this subsection, the proposed pre-caching strategy is evaluated in terms of the cache hit rate, mean response time, and energy cost. To investigate the performance on different workloads, the performance comparison is conducted on different request generation probabilities.

The cache hit rate performance comparison is depicted in Fig. 5. The overall cache hit rate, encompassing both streaming cache and popular cache, is presented in Fig. 5(a), while the separated popular cache hit rate and streaming cache hit rate are given in Fig. 5(b) and Fig. 5(c), respectively.

The performance comparison of the overall cache hit rate on different request generation probability is given Fig. 5(a). As shown in Fig. 5(a), there are two trends in the overall cache hit rate with the increase of request generation rate. Popularity-based methods, such as APC and PEC, exhibit an increasing trend, reaching the maximum overall hit rate at the highest workload. In contrast, context and mobility-aware methods, including DPC and the proposed method, show a decrease with the increase in the request generation rate.

The reason for these two opposing trends is due to the opposite trends observed in the hit rates of different cache types. For popular cache, the request probability follows the Zipf law. As the number of requests increases, many content requests are more likely to target the top popular content, further reducing the number of requests for less popular caches. Therefore, as long as RSUs cache these top-popular contents, the content cache hit rate increases. As a result, it is observed that the cache hit rates of all methods increase with the increase in the request generation probability. In this case, the proposed method can achieve the highest cache hit rate at 99.74% on the highest request generation probability, a slight improvement of 0.2% compared to the second-highest method.

The comparison of the streaming cache hit rate is illustrated in Fig. 5(c). The streaming cache hit rate shows a decreasing trend with the rise in request generation probability. In contrast to the hit rate of popular cache, streaming cache is more likely to be a one-time cache, making it less probable for repeated access in a short period. Consequently, the increasing number of requests exacerbates RSU capacity insufficiency, resulting in a decline in the content hit rate. Therefore, handling the streaming cache requires frequent content replacement and precise prediction.

An example can be observed from Fig. 5(c), where the APC method and PEC method have an intersection when the request generation probability equals 0.6. This is because the PEC method employs the LRU2 content replacement method. When the request generation probability is below 0.6, the workload pressure is relatively low, and LRU2 provides a slight improvement over LRU. However, as the workload increases, using LRU2 for updates leads to wasted cache space, resulting in a sharp decline in cache hit rate. In the case of the highest request generation, the APC and PEC methods can only achieve 56.07% and 52.08%, respectively.

Another reason for the decrease in the cache hit rate is that the pre-fetching of streaming cache relies more on the context of preceding requests rather than popularity. Although the DPC method, by combining content popularity and user mobility, can slightly improve the cache hit rate, even under the lowest workload conditions, the cache hit rate can only

reach 77.93%. In contrast, the proposed method can achieve at least 99.54% when the request generation probability is below 0.3 and achieve the highest cache hit rate in the worst case. This shows an 18.89% cache hit rate improvement under the same request generation probability.

In summary, considering the performance on both popular and streaming cache, the proposed method achieves an overall cache hit rate of 88.72%, representing an improvement of almost 10% compared to other methods.

The comparison of mean response time performance is illustrated in Fig. 6(a). Due to the low cache hit rates of the APC and PEC methods, RSUs need to retrieve the required content from the edge content server through the backhaul wireless transmission link to fulfill user content requirements, resulting in a higher mean response time. In contrast, owing to the load balance design in the pre-caching strategy and a higher cache hit rate, the proposed method consistently achieves the lowest mean response time under various workloads. Even in the case of the highest workload, the proposed method achieves a mean response time of 30.33 ms, presenting a performance improvement of approximately 6.10% compared to the DPC method.

The performance comparison of energy cost is depicted in Fig. 6(b). The energy cost of all methods increases with the rise in request generation probability. Similar to the response time analysis, the total energy cost is highly relevant to the cache hit rate. Due to frequent backhaul content transmission, the APC and PEC methods exhibit the highest and second-highest energy consumption, respectively. The PEC method attains the second-lowest energy cost owing to its relatively higher cache hit rate. Among all the compared methods, the proposed method achieves the lowest energy cost. Although the proposed method consumes more energy to pre-fetch content to the adjacent RSU for load balancing, it is still more energy-efficient compared to retrieving content through the content server. In comparison to the second-best energy-saving method, the proposed method can reduce 2.65% energy cost in the highest request generation probability case.

VII. Conclusion

In this paper, we propose a mobility and context-aware pre-caching strategy for vehicular content pre-caching in two steps. Firstly, a long-term vehicle trajectory prediction model ST-Informer is proposed to improve prediction accuracy. The proposed ST-Informer achieves the highest prediction accuracy, demonstrating reductions of 21.15% for MAE, 34.56% for MSE, 17.59% for RMSE, and 22.43% for MAPE compared to the original Informer and other state-of-the-art methods. Based on the prediction results, a context-aware pre-caching strategy is designed. Leveraging short-term and long-term previews from the prediction and the context of the content, the proposed pre-caching strategy can proactively pre-cache and replace different types of content based on the context of the current requested content. Compared to existing methods, the designed pre-caching strategy can improve the cache hit rate by 18.89% and reduce mean response delay and total energy cost by 6.1% and 2.65%, respectively, under the same workload.

References

[1] J. Clancy, D. Mullins, B. Deegan, J. Horgan, E. Ward, C. Eising, P. Denny, E. Jones, and M. Glavin, "Wireless access for v2x communications: Research, challenges and opportunities," *IEEE Communications Surveys & Tutorials*, 2024.

[2] L. Yao, Y. Wang, X. Wang, and G. WU, "Cooperative caching in vehicular content centric network based on social attributes and mobility," *IEEE Transactions on Mobile Computing*, vol. 20, no. 2, pp. 391–402, 2021.

[3] X. Jiang, F. R. Yu, T. Song, and V. C. M. Leung, "Resource allocation of video streaming over vehicular networks: A survey, some research issues and challenges," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 5955–5975, 2022.

[4] W. Zhao, S. Gong, D. Zhao, F. Liu, N. Sze, M. Quddus, and H. Huang, "Developing a new integrated advanced driver assistance system in a connected vehicle environment," *Expert Systems with Applications*, vol. 238, p. 121733, 2024.

[5] L. Liu, S. Lu, R. Zhong, B. Wu, Y. Yao, Q. Zhang, and W. Shi, "Computing systems for autonomous driving: State of the art and challenges," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6469–6486, 2020.

[6] M. A. Javed and S. Zeadally, "Ai-empowered content caching in vehicular edge computing: Opportunities and challenges," *IEEE network*, vol. 35, no. 3, pp. 109–115, 2021.

[7] Y. Zhang, C. Li, T. H. Luan, Y. Fu, W. Shi, and L. Zhu, "A mobility-aware vehicular caching scheme in content centric networks: Model and optimization," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3100–3112, 2019.

[8] B. Bharath, K. G. Nagananda, D. Gündüz, and H. V. Poor, "Caching with time-varying popularity profiles: A learning-theoretic perspective," *IEEE Transactions on Communications*, vol. 66, no. 9, pp. 3837–3847, 2018.

[9] H. S. Goian, O. Y. Al-Jarrah, S. Muhaidat, Y. Al-Hammadi, P. Yoo, and M. Dianati, "Popularity-based video caching techniques for cache-enabled networks: A survey," *IEEE Access*, vol. 7, pp. 27 699–27 719, 2019.

[10] Q. Chen, W. Wang, F. R. Yu, M. Tao, and Z. Zhang, "Content caching oriented popularity prediction: A weighted clustering approach," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 623–636, 2020.

[11] L. A. Adamic and B. A. Huberman, "Zipf's law and the internet." *Glottometrics*, vol. 3, no. 1, pp. 143–150, 2002.

[12] B. M. Maggs and R. K. Sitaraman, "Algorithmic nuggets in content delivery," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 3, pp. 52–66, 2015.

[13] L. Yao, Y. Wang, Q. Xia, and R. Xu, "Popularity prediction caching using hidden markov model for vehicular content centric networks," in *2019 20th IEEE International Conference on Mobile Data Management (MDM)*. IEEE, 2019, pp. 533–538.

[14] C. Song, W. Xu, T. Wu, S. Yu, P. Zeng, and N. Zhang, "Qoe-driven edge caching in vehicle networks based on deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 6, pp. 5286–5295, 2021.

[15] C. Li, X. Wang, T. Zong, H. Cao, and Y. Liu, "Predictive edge caching through deep mining of sequential patterns in user content retrievals," *Computer Networks*, p. 109866, 2023.

[16] P. He, L. Cao, Y. Cui, R. Wang, and D. Wu, "Multi-agent caching strategy for spatial-temporal popularity in iov," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 10, pp. 13 536–13 546, 2023.

[17] Z. Zhang and M. Tao, "Deep learning for wireless coded caching with unknown and time-variant content popularity," *IEEE Transactions on Wireless Communications*, vol. 20, no. 2, pp. 1152–1163, 2021.

[18] Z. Yu, J. Hu, G. Min, Z. Zhao, W. Miao, and M. S. Hossain, "Mobility-aware proactive edge caching for connected vehicles using federated learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 5341–5351, 2020.

[19] Q. Wu, Y. Zhao, Q. Fan, P. Fan, J. Wang, and C. Zhang, "Mobility-aware cooperative caching in vehicular edge computing based on asynchronous federated and deep reinforcement learning," *IEEE Journal of Selected Topics in Signal Processing*, vol. 17, no. 1, pp. 66–81, 2023.

[20] W. Zhuang, Q. Ye, F. Lyu, N. Cheng, and J. Ren, "Sdn/nfv-empowered future iov with enhanced communication, computing, and caching," *Proceedings of the IEEE*, vol. 108, no. 2, pp. 274–291, 2019.

[21] Y. Pei, M. Li, H. Wu, Q. Ye, C. Zhou, S. Hu, and X. Shen, "Joint caching and computing resource reservation for edge-assisted location-

aware augmented reality," in *ICC 2023-IEEE International Conference on Communications*. IEEE, 2023, pp. 2547–2552.

[22] S. Khodaparas, A. Benslimane, and S. Yousefi, "An intelligent caching scheme considering the spatio-temporal characteristics of data in internet of vehicles," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 5, pp. 7019–7033, 2024.

[23] N. Aung, S. Dhelim, L. Chen, A. Lakas, W. Zhang, H. Ning, S. Chaib, and M. T. Kechadi, "Vesonet: Traffic-aware content caching for vehicular social networks using deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 8, pp. 8638–8649, 2023.

[24] B. Feng, C. Feng, D. Feng, Y. Wu, and X.-G. Xia, "Proactive content caching scheme in urban vehicular networks," *IEEE Transactions on Communications*, vol. 71, no. 7, pp. 4165–4180, 2023.

[25] Z. H. Meybodi, A. Mohammadi, E. Rahimian, S. Heidarian, J. Abouei, and K. N. Plataniotis, "Tedge-caching: Transformer-based edge caching towards 6g networks," in *ICC 2022-IEEE International Conference on Communications*. IEEE, 2022, pp. 613–618.

[26] M. Han, X. Sun, X. Wang, W. Zhan, and X. Chen, "Joint caching, communication, computation resource management in mobile-edge computing networks," in *2024 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2024, pp. 1–6.

[27] M. Xu, D. Niyato, H. Zhang, J. Kang, Z. Xiong, S. Mao, and Z. Han, "Sparks of generative pretrained transformers in edge intelligence for the metaverse: Caching and inference for mobile artificial intelligence-generated content services," *IEEE Vehicular Technology Magazine*, vol. 18, no. 4, pp. 35–44, 2023.

[28] Z. Chen, L. X. Cai, and X. Hao, "Near-field and far-field beamforming design for ris-enabled millimeter wave systems," in *2024 IEEE 99th Vehicular Technology Conference (VTC2024-Spring)*, 2024, pp. 1–6.

[29] P. Yadav and S. Kar, "Efficient content distribution in fog-based cdn: A joint optimization algorithm for fog-node placement and content delivery," *IEEE Internet of Things Journal*, vol. 11, no. 9, pp. 16 578–16 590, 2024.

[30] X. Deng, P. Guan, C. Hei, F. Li, J. Liu, and N. Xiong, "An intelligent resource allocation scheme in energy harvesting cognitive wireless sensor networks," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 1900–1912, 2021.

[31] 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," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 12, 2021, pp. 11 106–11 115.

[32] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[33] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," *arXiv preprint arXiv:1803.02155*, 2018.

[34] L. Bieker, D. Krajzewicz, A. Morra, C. Michelacci, and F. Cartolano, "Traffic Simulation for All: A Real World Traffic Scenario from the City of Bologna," in *Modeling Mobility with Open Data*. Springer, 2015, pp. 47–60.

[35] H. Peng, Q. Ye, and X. Shen, "Spectrum management for multi-access edge computing in autonomous vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 7, pp. 3001–3012, 2019.

[36] L. Li, Y. Xu, J. Yin, W. Liang, X. Li, W. Chen, and Z. Han, "Deep reinforcement learning approaches for content caching in cache-enabled d2d networks," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 544–557, 2019.

[37] X. Zhu, Y. Luo, A. Liu, W. Tang, and M. Z. A. Bhuiyan, "A deep learning-based mobile crowdsensing scheme by predicting vehicle mobility," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4648–4659, 2021.

[38] C. Li, M. Song, S. Du, X. Wang, M. Zhang, and Y. Luo, "Adaptive priority-based cache replacement and prediction-based cache prefetching in edge computing environment," *Journal of Network and Computer Applications*, vol. 165, p. 102715, 2020.