Spatio-temporal Multivariate Time Series Forecast with Chosen Variables

Zibo Liu
CISE
University of Florida
Gainesville, FL, USA
ziboliu@ufl.edu

Zhe Jiang
CISE
University of Florida
Gainesville, FL, USA
zhe.jiang@ufl.edu

Zelin Xu
CISE
University of Florida
Gainesville, FL, USA
zelin.xu@ufl.edu

Tingsong Xiao
CISE
University of Florida
Gainesville, FL, USA
xiaotingsong@ufl.edu

Yupu Zhang
CISE
University of Florida
Gainesville, FL, USA
y.zhang1@ufl.edu

Zhengkun Xiao
CISE
University of Florida
Gainesville, FL, USA
xiaoz@ufl.edu

Haibo Wang
Department of Computer Science
University of Kentucky
Lexington, KY, USA
haibo@ieee.org

Shigang Chen
CISE
University of Florida
Gainesville, FL, USA
sgchen@ufl.edu

***Abstract*—Spatio-Temporal Multivariate time series Forecast (STMF) uses the time series of n spatially distributed variables in a period of recent past to forecast their values in a period of near future. It has important applications in spatio-temporal sensing forecast such as road traffic prediction and air pollution prediction. Recent papers have addressed a practical problem of missing variables in the model input, which arises in the sensing applications where the number m of sensors is far less than the number n of locations to be monitored, due to budget constraints. We observe that the state of the art assumes that the m variables (i.e., locations with sensors) in the model input are pre-determined and the important problem of how to choose the m variables in the input has never been studied. This paper fills the gap by studying a new problem of STMF with chosen variables, which optimally selects m-out-of-n variables for the model input in order to maximize the forecast accuracy. We propose a unified framework that jointly performs variable selection and model optimization for both forecast accuracy and model efficiency. It consists of three novel technical components: (1) *masked variable-parameter pruning*, which progressively prunes less informative variables and attention parameters through quantile-based masking; (2) *prioritized variable-parameter replay*, which replays low-loss past samples to preserve learned knowledge for model stability; (3) *dynamic extrapolation mechanism*, which propagates information from variables selected for the input to all other variables via learnable spatial embeddings and adjacency information. Experiments on five real-world datasets show that our work significantly outperforms the state-of-the-art baselines in both accuracy and efficiency, demonstrating the effectiveness of joint variable selection and model optimization.**

***Index Terms*—Multivariate time series, spatio-temporal forecast, machine learning, budget constraints**

I. Introduction

New Problem: The problem of *Spatio-Temporal Multivariate time series Forecast* (STMF) is to use the time series of n spatially distributed variables in a period l of recent past to forecast their values in a period l' of near future. It has been extensively studied [1], [2], [3], [4], which learn their forecast models from the training data of the n time series recorded for adequately long time. Recent studies have explored two variants of STMF. The first one is *STMF with missing data* [5], [6], [7], [8], which deals with sporadically missing values in the $n \times l$ input. The second variant is *STMF with missing variables* [9], [10], [11], which deals with permanently missing a subset of variables in the input — the model input dimensions become $m \times l$ with $n - m$ variables missing, where $m < n$. These variants are motivated from practical applications, and they assume that while the model input for forecasting may miss variables, the training data — serving as the ground truth for modeling — have all n time series, which will be justified shortly. The difference between STMF and its variants is illustrated in Figure 1(a)–(c).

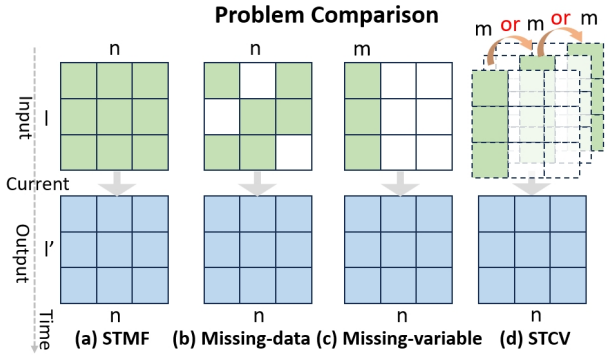

Fig. 1. Problem comparison: (a) STMF, using $n \times l$ input of the past to forecast $n \times l'$ output of the future, (b) STMF with missing data, where some values in the input are missing, (c) STMF with missing variables, where some variables are missing from the input due to budget constraint, using $m \times l$ input to forecast $n \times l'$ output, (d) STCV, allowing optimal selection of the m variables in the input to maximize forecast accuracy.

The above problem of STMF with missing variables assumes that the m variables in the model input are given, without considering how to select them from the n variables. This paper studies a new, complementary problem, *STMF with Chosen Variables*, denoted as STCV, that addresses the issue of optimally choosing the m variables in the input to maximize

the model forecast accuracy. Its difference from the prior studies is illustrated in Figure 1(d). Its practical motivation is given below.

Application Motivation: Consider a city with n locations of interest for traffic forecast, which may help monitor potential congestion and provide data for optimizing traffic signing. In this STMF application, sensors can be deployed at the n locations (e.g., street intersections or highway exits) to record traffic volumes in 5-min intervals. A model is learned from the training data, i.e., n time series of sensor data over a long period of recording, e.g., 6 months. When the model is deployed, at each time interval, it may take one past hour of sensor data as input (size $n \times 12$) to forecast one future hour of traffic as output (size $n \times 12$), where one hour has 12 intervals of 5 minutes.

The case of missing data mirrors sporadic sensor failures. The case of missing variables (i.e., missing sensors) is due to cost saving. The equipment and installation cost of permanent traffic sensors is in thousands of dollars per intersection [12], [13], [14], for example, \$10,000–\$30,000 for an inductive loop and \$20,000–\$40,000 for a video-based system, whereas cheap, temporary sensor installation only costs hundreds of dollars a piece [15]. Considering hundreds or thousands of street intersections in a city, if the budget is limited, one may consider deploying expensive permanent sensors at only a subset of m intersections to save cost, while using cheap, temporary sensors to collect the training data at all locations —- these temporary sensors will be removed and reused elsewhere after model training, while m permanent sensors will be deployed to provide ongoing input to the model for forecasting. The cost saving can be substantial if one can drive m down to a small fraction of n such as 10% as our evaluation will achieve. The existing work on STMF with missing variables [9], [10], [11] assumes the locations of m permanent sensors are predetermined and investigates how to build a forecast model to forecast traffic at the $n - m$ locations without permanent sensors. This paper is complementary to the existing work by choosing the m locations for optimal deployment of permanent sensors.

Another application that is also used in our evaluation is air quality monitoring, where permanent sensors are deployed in some chosen locations to provide input for a model to forecast the air quality at numerous other locations without permanent sensors.

Challenges: The new problem of STMF with chosen variables (STCV) presents several fundamental challenges. First, the underlying distributions of the variables can vary significantly. As shown in Fig. 2, traffic patterns at locations 121, 44, and 33 from the PEMS08 dataset differ notably, which is also observed for numerous other locations, making it difficult to choose where sensors should be replaced so that readings from the chosen locations can accurately forecast other locations. Second, the space of possible variable selections is combinatorially large. Given n variables and a budget of m in the model input, there are $\binom{n}{m}$ possible selections, which makes exhaustive search infeasible, especially for very

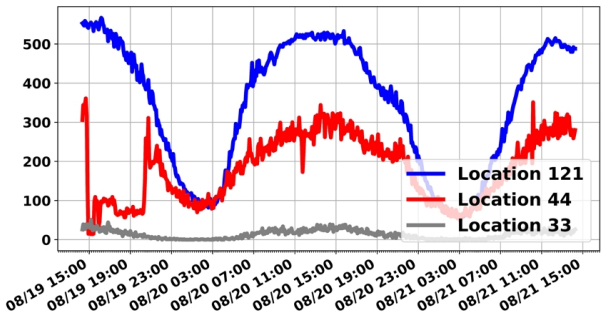

Fig. 2. visualization of traffic volumes at three locations for two days from the PEMS08 dataset.

large scale deployments. Third, the data scale can be very large. Consider the traffic forecast application. The model input/output will include a spatio-temporal traffic matrix of tens to hundreds intersections (town), to thousands (city), to tens of thousands (e.g., over 50,000 intersections in NYC) in the space dimension and numerous time intervals in the time dimension, for potentially millions variables, together with additional topological input. This is time series forecast with a very large number of variables. It poses two challenges. One is to accurately forecast the future values of a large number of variables from the past values of a relatively small number of variables. The other is to do so efficiently, both in memory and in inference time, in order to scale to large problem size. Finally, as variable selection and model structure evolve during training, the model faces the risk of *catastrophic forgetting*, where previously learned patterns may be lost as variables are removed from the model input. Ensuring stable learning during variable selection presents an additional challenge.

Contributions: To address the STCV problem, we propose a new model-learning framework called *Variable-Parameter Iterative Pruning (VIP)*, which consists of three components: masked variable-parameter pruning, dynamic extrapolation, and prioritized variable-parameter replay. The main contributions of this paper are as follows:

• **New problem Introduction.** We introduce and formally define the problem of spatio-temporal multivariate time series forecast with chosen variables, which integrates variable selection and spatio-temporal time series forecast into a unified optimization task under a given budget of variables in the input. This problem setting reflects deployment constraints in real-world applications and has not been systematically studied before.

• **Masked variable-parameter pruning**. We design a unified masked pruning mechanism that simultaneously selects variables and prunes redundant attention parameters. This design enables the model to learn compact yet accurate representations, resulting in both improved forecast accuracy and reduced model complexity. Our variable-parameter pruning technique can not only solve the problem of optimal variable selection for forecast accuracy, but also address scalability of large problem size. The model's memory footprint is drastically reduced with variable-parameter pruning and its inference time

is also reduced with a smaller model.

• **Dynamic extrapolation.** We introduce a dynamic extrapolation module that leverages graph structure and node embeddings (that are iteratively learned) to infer values of missing variables, i.e., those that are not selected for the input. This enables accurate forecasting even with highly sparse variable selection.

• **Prioritized variable-parameter replay.** We propose a prioritized replay strategy that buffers and reuses informative samples during training. This mitigates catastrophic forgetting during variable-parameter pruning, ensuring stable optimization and model convergence.

II. Preliminaries

A. Problem Definition

Definition 1: Consider a set N of n time series variables. For the road traffic application in the introduction, each variable represents a location of interest, and its time series consists of the traffic volumes at the location in successive time intervals; for the air quality application, each variable represents a location in a city, and its time series may consist of the PM2.5 measurements after successive time intervals. For simplicity, we normalize each time interval as one unit of time. Let M be a subset of variables, i.e., $M \subseteq N$, where $m = |M|$. Let T be a series of time intervals — for example, T may be $\{t-l+1, ..., t-1, t\}$ of l intervals, where t is the current time. The *value matrix* over variables M and time T is defined as $\mathcal{X}_{M,T} = [\mathcal{X}_{i,j}, i \in M, j \in T]$, where $\mathcal{X}_{i,j}$ is the value of variable i at time j. Further denote the time series of variable i over T as $\mathcal{X}_{i,T} = [\mathcal{X}_{i,j}, j \in T]$. Hence, $\mathcal{X}_{M,T} = [\mathcal{X}_{i,T}, i \in M]$, consisting of m time series for m variables over time T. Let $A = [A_{i,j}, i, j \in N] \in \mathbb{R}^{n \times n}$ be a *spatial adjacency matrix* of the variables in N. It is needed for spatio-temperal forecast, where the distributed variables are spatially connected — for example, the intersections in the road traffic application are interconnected by a road system. The adjacency matrix A defines a graph where each node represents a variable. The ith row (column) in A represents variable i (or node i). $A_{i,j} \neq 0$ indicates there is a link between node i and node j, and its value represents a static connectivity feature of the link. For example, $A_{i,j} = 1$ (or 0) may indicate that there is (or not) a road connecting location i and location j [16], and its value may be set proportional to the road distance. In the air quality application, A can be a learnable adjacency matrix [11]. In summary, A represents the static feature of the multivariate time series system, whereas $\mathcal{X}_{N,T}$ represents the dynamic feature of the system.

Definition 2: *Problem of Spatio-Temporal Multivariate Time Series Forecast with Chosen Variables (STCV).* The problem of STCV is to build a model that selects a subset M of m variables as the model input and uses their value matrix $\mathcal{X}_{M,T}$ of the recent past to forecast the value matrix $\mathcal{X}_{N,T'}$ in the near future for all variables, where $n > m$, $T' = \{t+1, ..., t+l'\}$, $T = \{t-l+1, ..., t-1, t\}$, l' is the length of the recent past in the input, l is the length of the near future in the output, and time t is the current time. M is the subset of m chosen variables in the model input, and $M' = N - M$ is the subset of m' remaining variables missing from the input.

There are two special cases: If $m = n$, the problem becomes the traditional STMF, which was extensively studied [17], [16], [1], [2]. If $m < n$ and M is given, it is STMF with missing variables, which was studied recently [11], [9].

Similar to [11], [9], we assume that the training data contains the time series of all variables in both M and M' as the ground truth for model training, which has been justified in the introduction.

B. Temporal Embeddings and Node Embeddings

Learnable temporal embeddings, denoted as E_N^{temp}, are used to capture temporal patterns. They are application-dependent. For example, for the road traffic application, they may include time-of-day (tod) embeddings and day-of-week (dow) embeddings [18], [1]: Discretize a day into $D = 288$ time steps of 5-minute each, and a week into $W = 7$ days. Each time step in $[0, D)$ is mapped to a trainable time-of-day embedding vector in $\mathbb{R}^{d_{tod}}$, and each day in $[0, W)$ is mapped to a trainable day-of-week embedding vector in $\mathbb{R}^{d_{dow}}$. Given the value matrix $\mathcal{X}_{N,T} \in \mathbb{R}^{n \times l}$, where n is the number of variables and l is the length of time T, we assign temporal embeddings to each entry $\mathcal{X}_{i,j}$. This yields a time-of-day embedding matrix $E_N^{tod} \in \mathbb{R}^{n \times l \times d_{tod}}$ and a day-of-week embedding matrix $E_N^{dow} \in \mathbb{R}^{n \times l \times d_{dow}}$. We aggregate the above learnable temporal embeddings as $E_N^{temp} = E_N^{tod} || E_N^{dow}$. In addition, each node v in the graph has a learnable node embedding vector in $\mathbb{R}^{d_v}$; the node embedding matrix is denoted as $E_N^{node} \in \mathbb{R}^{n \times l \times d_v}$, which repeats the node embeddings across time T of length l.

III. Framework of Variable-Parameter Iterative Pruning

A. Pre-training A Base STMF Model

We begin with a base STMF model to address the spatio-temporal multivariate time series forecast problem. The model is denoted as $F(\mathcal{X}_{N,T}, A, \Theta)$, which takes a past value matrix $\mathcal{X}_{N,T}$ from all variables N together with A as input to forecast a future value matrix $\mathcal{X}_{N,T'}$ as output, with the learned model parameters Θ. The model begins with a Multi-Layer Perceptron (MLP) that processes the input data $\mathcal{X}_{N,T}$ and produce a feature embedding $E_N^f \in \mathbb{R}^{n \times l \times d}$. It then concatenates this with the learnable temporal embeddings E_N^{temp} and the learnable node embeddings E_N^{node} to form aggregated features $E_N^{all} \in \mathbb{R}^{n \times l \times q}$, where q is the number of dimensions in the attention layers. E_N^{all} is then passed through a number of temporal attention layers, each denoted as ATT_t, and a number of spatial attention layers, each denoted as ATT_s, to produce a high-dimensional representation $H_{N,T} \in \mathbb{R}^{n \times l \times q}$.

$$E_N^f = MLP(\mathcal{X}_{N,T}) \tag{1}$$

$$E_N^{all} = E_N^f || E_N^{node} || E_N^{temp} \tag{2}$$

$$H_{N,T} = ATT_s(...ATT_s(ATT_t(...ATT_t(E_N^{all})...))...) \tag{3}$$

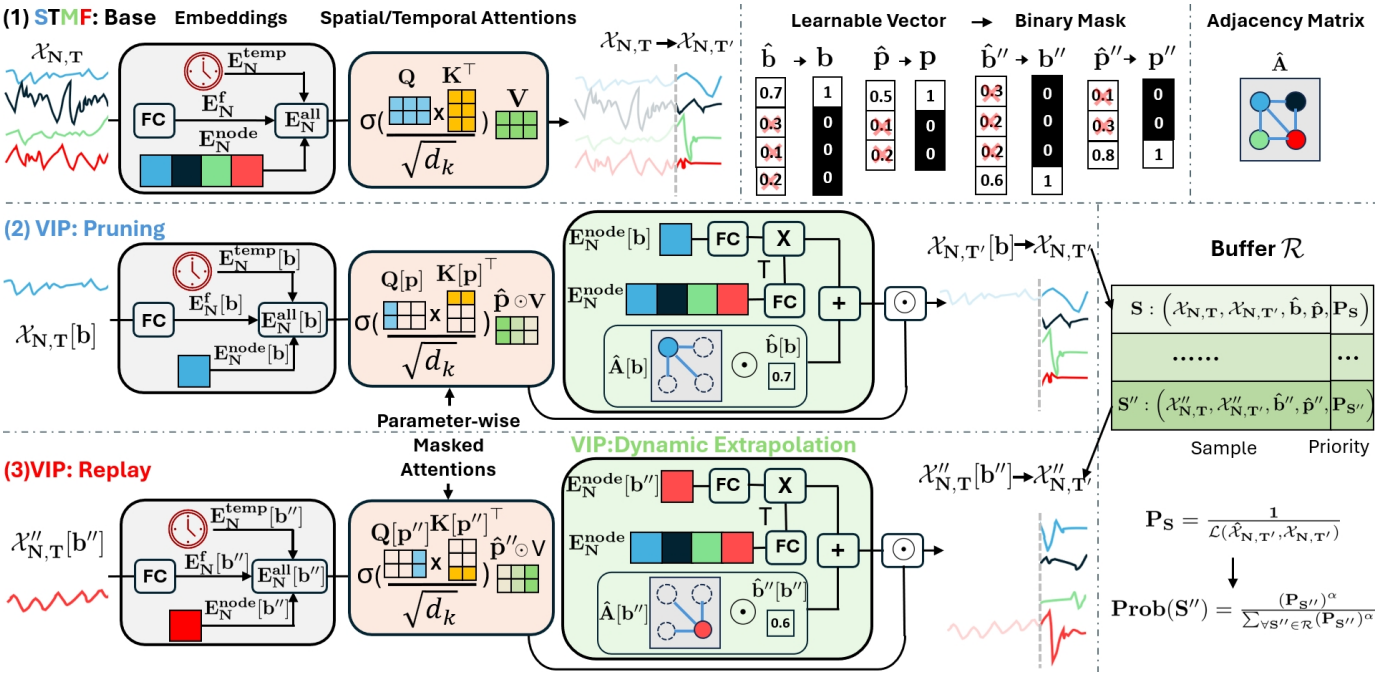

Fig. 3. (1) the base STMF model, shown at the top left of the figure. (2) VIP: pruning, shown at the middle left. It learns from the current sample, $S:\left(\mathcal{X}_{N,T},\mathcal{X}_{N,T'},\hat{b},\hat{p},P_S\right)$, and then stores the sample in a buffer, shown at the middle right. (3) VIP: replay, shown at the bottom left. It learns from a replayed sample that is retrieved from the buffer, denoted as $S'':\left(\mathcal{X}''_{N,T},\mathcal{X}''_{N,T'},\hat{b}'',\hat{p}'',P_{S''}\right)$, shown at the bottom right. (4) learnable variable and parameter vectors $\hat{b}$, $\hat{p}$, along with their corresponding binary masks b, p, shown at the top right. (5) adjacency matrix $\hat{A}$, shown at the top right. In (1), (2) and (3) of the figure, we illustrate the temporal embeddings E_N^{temp}, the feature embedding E_N^f, and the node embeddings E_N^{node} (gray blocks); spatial or temporal attentions (orange block in (1)); parameter-wise masked attentions (orange blocks in (2) and (3)); and dynamic extrapolation (green blocks).

where ATT_t and ATT_s are designed to capture the spatio-temporal correlations. Let $H^t_{N,T}\in\mathbb{R}^{n\times l\times q}$ be the output of the first temporal attention, $ATT_t(E^{all}_N)$. It is calculated as follows: We obtain query Q^t, key K^t and value V^t matrices through the temporal attention layer, where W^t_Q, W^t_K, $W^t_V\in\mathbb{R}^{q\times q}$ are their learnable parameters, respectively:

$$Q^t=E^{all}_N W^t_Q,\ K^t=E^{all}_N W^t_K,\ V^t=E^{all}_N W^t_V \quad (4)$$

$$H^t_{N,T}=Softmax\left(\frac{Q^t(K^t)^\top}{\sqrt{d_t}}\right)V^t \quad (5)$$

where d_k is the number of heads in the multi-head mechanism of the attention. We repeat for additional temporal attention layers, each with its input being the output of the previous layer, using that input to substitute E^{all}_N in Eq. (4)-(5). Let $H^s_{N,T}\in\mathbb{R}^{n\times l\times q}$ be the output of the spatial attention, $ATT_s((H^t_{N,T})^\top)$, where $\top$ is for transpose. Each spatial attention is implemented similarly, with $ATT_s(H)=ATT_t(H^\top))$ for any input H. Let $H^s_{N,T}\in\mathbb{R}^{n\times l\times q}$ be the output after all spatial attentions. Finally, we use an MLP to map $\hat{H}^s_{N,T}\in\mathbb{R}^{n\times l\times q}$ to the forecast output $\hat{\mathcal{X}}_{N,T'}\in\mathbb{R}^{n\times l'}$:

$$\hat{\mathcal{X}}_{N,T}=MLP(H^s_{N,T}) \quad (6)$$

B. Variable-Parameter Iterative Pruning

From STMF to STCV: The proposed STMF model is designed to forecast future values of all time series in N based on their past values. It relies on dense neural networks, i.e., embeddings and attention mechanisms, and requires all variables in the model input.

We regard STCV as a variable reduction problem and produce its model from the pretrained STMF model by iteratively pruning insignificant variables to select a final desired set M of variables. With a reduced number of variables, we also prune insignificant dimensions of model parameters to reduce the model size and thus improve its inference efficiency. Furthermore, reducing input variables and parameter dimensions not only lowers computational cost but also effectively shrinks the combinatorial search space for optimal model configurations, thus enabling more efficient learning [19].

Below we propose a new learning framework for STCV, called *Variable-Parameter Iterative Pruning (VIP)*, where each iteration jointly prunes less-important variables and parameters and the process ends after a budgeted number m of variables are reached.

Overview: VIP jointly tackles the challenges of variable selection and model compression in the STCV setting. It progressively reduces the number of input variables and model parameters through an iterative masking strategy. VIP is composed of the three tightly integrated components:

(1) *Masked Variable-Parameter Pruning.* We use learnable masks over variables and attention parameters to identify and eliminate less-informative inputs and model dimensions by quantile-based thresholding. Such a pruning mechanism is

illustrated on the left side of Fig. 3.

(2) *Dynamic Extrapolation.* We define a unified training objective that integrates forecast loss, replay loss, and stochastic regularization over mask vectors to balance convergence and exploration. We enable dynamic extrapolation by propagating the features learned from chosen variables in M to missing variables in M' via the node embeddings, as shown in the green blocks of Fig. 3.

(3) *Prioritized Variable-Parameter Replay.* To alleviate catastrophic forgetting induced by dynamic pruning, we employ a replay buffer that retains previously encountered low-loss samples, prioritizing those deemed more informative. This mechanism is depicted in the bottom-right corner of Fig. 3.

Together, these components form a cohesive learning framework that jointly optimizes variable selection and parameter sparsity, enabling scalable and accurate spatio-temporal forecast.

***Masked Variable-Parameter Pruning**:* We first introduce variable mask and parameter mask. We then explain how to perform variable-parameter pruning.

Variable Mask. We use a binary variable mask $b \in \{0,1\}^n$ to specify variable selection: variable i is selected iff $b_i = 1$, where b_i is the ith masking bit of b, $\forall i \in [0, n)$. Given a value matrix $\mathcal{X}_{N,T}$ and a variable mask b, we define $\mathcal{X}_{N,T}[b]$ as a sub-matrix with only those time series $\mathcal{X}_{i,T}$ whose corresponding masking bits in b are 1's, i.e., $b_i = 1$, where $[b]$ is a *masking operator*: When it is applied to a matrix of size $n \times ...$, it selects the rows whose masking bits are ones; when it is applied to a vector of size n, it selects the elements whose masking bits are ones. For another example, given an adjacency matrix A and a variable mask b, $A[b]$ is a sub-adjacency matrix with only those rows (variables) whose mask bits in b are 1's.

Parameter Mask. Given the set Θ of parameters in an STMF model, with each of its attentions having q dimensions of parameters, and a binary parameter mask $p \in \{0,1\}^q$ that is applied to all attentions, we define $\Theta[p]$ as a subset of parameters with only those dimensions (in each attention) whose mask bits in p are 1's, i.e., $p_j = 1$, where p_j is the jth masking bit in p. In the sequel, this parameter masking operation $[p]$ will also be applied to other parameter sets of q dimensions.

Iterative Pruning. Given a pre-trained STMF model $F(\mathcal{X}_{N,T}, A, \Theta)$, we will adopt $F(\mathcal{X}_{N,T}[b], A[b], \Theta[p])$ as an STCV model that forecasts the future values of all variables, but with fewer input variables $\mathcal{X}_{N,T}[b]$ and fewer model parameters $\Theta[p]$, where b and p are iteratively optimized.

Starting from the initial masks of b^0 and p^0 whose bits are all ones, our task is to learn b^k and p^k together iteratively, where k is the number of iterations, such that the final masks, denoted as b^* and p^*, achieve the best performance for STCV, where $\|b^*\|_0 = m$, and $\|p^*\|_0 = q'$ for a desired target number q' of parameter dimensions in each attention, with $q' < q$.

Let $\hat{b} \in \mathbb{R}^n$ be a learnable variable importance vector, and $\hat{p} \in \mathbb{R}^q$ be a learnable parameter importance vector. These vectors are optimized jointly with the model parameters during training. The binary variable mask b^k and parameter mask p^k at the k-th iteration are derived from $\hat{b}$ and $\hat{p}$ via quantile-based thresholding.

The vector $\hat{b}$ is initialized based on the normalized graph structure: $\hat{b} = \left[\sum_{j=1}^n \hat{A}_{0j}, \ldots, \sum_{j=1}^n \hat{A}_{nj}\right]^\top$, where $\hat{A} = \tilde{D}^{-\frac{1}{2}}(A+I)\tilde{D}^{-\frac{1}{2}}$ is the symmetrically normalized adjacency matrix with self-loops. The i-th value of $\hat{b}$ is denoted $\hat{b}_i$, for $i \in [0, n)$. The vector $\hat{p}$ is initialized from a standard normal distribution, with $\hat{p}_j$ denoting the j-th element, for $j \in [0, q)$.

The binary masks b^k and p^k are updated as follows:

$$b_i^k = \begin{cases} 0, & \text{if } |\hat{b}_i| \leq \text{Quantile}(|\hat{b}[b^{k-1}]|, r_b) \\ 1, & \text{otherwise} \end{cases} \tag{7}$$

$$p_j^k = \begin{cases} 0, & \text{if } |\hat{p}_j| \leq \text{Quantile}(|\hat{p}[p^{k-1}]|, r_p) \\ 1, & \text{otherwise} \end{cases} \tag{8}$$

Here, $[\,\cdot\,]$ denotes the masking operator that selects elements whose masking bits are 1s. The operator $|\cdot|$ takes element-wise absolute values, and $\text{Quantile}(x, \beta)$ denotes the β-quantile of vector x. The scalars r_b and r_p are the pruning rates for variable and parameter masks, respectively.

From Eq. (7)-(8), after the k-th iteration, we will retain $\|b^k\|_0$ variables, where $\|b^k\|_0 = max\{\lfloor n \times (1-r_b)^k \rfloor, 1\}$, and retain $\|p^k\|_0$ dimensions of parameters in each attention, where $\|p^k\|_0 = max\{\lfloor q \times (1-r_p)^k \rfloor, 1\}$.

Consider a value matrix of all variables, $\mathcal{X}_{N,T} \in \mathbb{R}^{n \times l}$. With the retained variable mask b^k, the masked value matrix is $\mathcal{X}_{N,T}[b^k] \in \mathbb{R}^{\|b^k\|_0 \times l}$. From Eq. (1)-(2) in the STMF model, we can calculate its aggregated feature $E_N^{all}[b^k] \in \mathbb{R}^{\|b^k\|_0 \times l \times q}$.

To effectively retain the model's learning power while reducing the model parameters, we design a parameter-wise masked attention, which is shown in the orange blocks of Fig. 3. It applies the learnable parameter vector $\hat{p}$ and the binary parameter mask p^k to each attention in Eq. (4) and (5) as follow:

$$\hat{Q}^t = \left(E_N^{all}[b^k]\right)\left(W_Q^t[p^k]\right) \tag{9}$$

$$\hat{K}^t = \left(E_N^{all}[b^k]\right)\left(W_K^t[p^k]\right) \tag{10}$$

$$\hat{V}^t = \hat{p} \odot \left(E_N^{all}[b^k] W_V^t\right) \tag{11}$$

$$H_{N,T}^t[b^k] = Softmax\left(\frac{\hat{Q}^t(\hat{K}^t)^\top}{\sqrt{d_t}}\right)\hat{V}^t \tag{12}$$

where $W_Q^t[p^k], W_K^t[p^k] \in \mathbb{R}^{q \times \|p^k\|_0}$ and $\odot$ is the element-wise product. Spatial attentions are handled similarly except that transpose is performed on the input, i.e., $\hat{K}^t$.

***Dynamic Extrapolation**:* To ensure accurate forecast, we must propagate learned representations from the variables that are selected to the variables that are not selected. We design a dynamic extrapolation mechanism, which is shown in the green blocks of Fig. 3, that constructs a global information bridge between variables in M and those in M'. It operates in two stages: (1) computing a soft similarity-based attention matrix over variable embeddings, and (2) fusing it with the original masked adjacency matrix to allow knowledge transfer across all variables.

a) Step 1: Similarity-based Extrapolation Attention.: We first compute a global attention matrix $\hat{B} \in \mathbb{R}^{m\times n}$ based on node embedding similarity:

$$\hat{B} = \sigma\left(FC(E_N^{node}[b^k]) \cdot FC(E_N^{node})^\top\right) \tag{13}$$

where $FC(\cdot)$ is a shared learnable linear projection. The activation $\sigma(\cdot)$ is the GeLU function [20]. This attention matrix $\hat{B}$ serves as a dense bridge from the variables selected by mask b^k to all variables.

b) Step 2: Fusion with Masked Adjacency.: We then combine $\hat{B}$ with the masked adjacency matrix to form a final extrapolated adjacency:

$$\hat{A}' = \hat{b}[b^k] \odot \hat{A}[b^k] + \hat{B} \tag{14}$$

Here, $\hat{A}[b^k]$ is the symmetrically normalized adjacency matrix for the selected variables, and $\hat{b}[b^k]$ acts as a learned variable-level weight. This fused matrix $\hat{A}' \in \mathbb{R}^{m\times n}$ enables spatial message passing from the selected variables to all variables.

c) Step 3: Feature Propagation.: The extrapolated representation for all variables is computed as:

$$H_{N,T} = \hat{A}'^\top \cdot H_{N,T}[b^k] \tag{15}$$

This mechanism allows latent features from the selected variables to influence other variables via learned attention, enhancing forecast quality under sparse input.

Finally, an MLP is used to map the high-dimensional representation $H_{N,T}$ to the forecast output $\hat{\mathcal{X}}_{N,T'}$ as in Eq. (6), and the loss function is given below

$$\mathcal{L}_{main}(\cdot,\cdot) = \mathcal{L}\left(F(\mathcal{X}_{N,T}[b^k], A[b^k], \Theta[p^k]), \mathcal{X}_{N,T'}\right). \tag{16}$$

***Prioritized Variable-Parameter Replay**:* Within the kth iteration of VIP, mask b^k for variable selection and mask p^k for parameter selection will be updated together with the learnable vectors $\hat{b}$ and $\hat{p}$ during training. If the two masks differ significantly after each update, the model might not retain the significant patterns from the previous training, which is a phenomenon known as catastrophic forgetting [21]. The experience replay (ER) methods [22], which store some historical samples in a buffer and replay them to the current model using a first-in-first-out strategy, can mitigate the problem of catastrophic forgetting. Furthermore, the prioritized experience replay method [23] assigns priority to the samples for replay and achieves better performance.

Motivated from the prior ER work, we introduce the prioritized variable-parameter replay method (PVR) to mitigate catastrophic forgetting in our proposed VIP. After a new sample is applied in training, PVR stores this sample $S : \left(\mathcal{X}_{N,T}, \mathcal{X}_{N,T'}, \hat{b}, \hat{p}, P_S\right)$ in a buffer $\mathcal{R}$, which can keep up to $\|\mathcal{R}\|$ samples, where the priority P_S of the current sample S is given as

$$P_S = \frac{1}{\mathcal{L}_{main}(\cdot,\cdot)}. \tag{17}$$

Algorithm 1 k-th iteration of variable-parameter iterative pruning in model learning

Input: Value matrix $\mathcal{X}_{N,T}$, adjacency matrix A, STMF model $F(\cdot, A, \Theta)$, trainable variable mask $\hat{b}$ and parameter mask $\hat{p}$, buffer $\mathcal{R}$, learning rate λ
Output: Sparsified variable mask b^k, parameter mask p^k, model parameters Θ
$\mathcal{R} \leftarrow \{\}$
repeat
  // Compute binary masks for current sample
  Compute variable mask b^k and parameter mask p^k from $\hat{b}$ and $\hat{p}$ by Eq. (7)-(8)
  // Forward pass on current sample
  Forward $F(\mathcal{X}_{N,T}[b^k], A[b^k], \Theta[p^k])$ to compute the main loss $\mathcal{L}_{\text{main}}(\cdot,\cdot)$ by Eq. (16)
  // Replay sample selection and loss computation
  Select a replay sample $(\mathcal{X}''_{N,T}, \mathcal{X}''_{N,T'}, \hat{b}'', \hat{p}'')$ from $\mathcal{R}$ using Eq. (17) and Eq. (18)
  Compute variable mask $b^{''k}$ and parameter mask $p^{''k}$ from $\hat{b}''$ and $\hat{p}''$ by Eq. (7)-(8)
  Forward $F(\mathcal{X}''_{N,T}[b^{''k}], A[b^{''k}], \Theta[p^{''k}])$ to compute the replay loss $\mathcal{L}_{\text{replay}}(\cdot,\cdot)$ by Eq. (19)
  // Total loss and gradient update
  Obtain total loss $\mathcal{L}_{\text{sum}}(\cdot,\cdot)$ by Eq. (20)
  Backpropagate to update $\Theta \leftarrow \Theta - \lambda\nabla_\Theta \mathcal{L}_{\text{sum}}(\cdot,\cdot)$
  Update $\hat{b} \leftarrow \hat{b} - \lambda\nabla_{\hat{b}} \mathcal{L}_{\text{sum}}(\cdot,\cdot)$
  Update $\hat{p} \leftarrow \hat{p} - \lambda\nabla_{\hat{p}} \mathcal{L}_{\text{sum}}(\cdot,\cdot)$
  // Store current sample into replay buffer
  Insert current sample $(\mathcal{X}_{N,T}, \mathcal{X}_{N,T'}, \hat{b}, \hat{p}, P_S)$ into $\mathcal{R}$
until all batches in epoch are processed

Intuitively, in the context of VIP where variable-parameter pruning can cause learned patterns to be pruned from the model, low-loss samples should be given priorities to combat catastrophic forgetting. This assumption is supported by experimental evaluation in Section IV-H, confirming that samples yielding lower $\mathcal{L}_{main}$ help retain significant patterns.

When inserting a new sample to $\mathcal{R}$, if $\mathcal{R}$ is full, PVR will probabilistically select an existing sample from $\mathcal{R}$, replay it, and then remove it. The probability for a sample $S' \in \mathcal{R}$ to be selected is

$$Prob(S') = \frac{(P_{S'})^\alpha}{\sum_{\forall S'\in\mathcal{R}}(P_{S'})^\alpha} \tag{18}$$

where α is the hyperparameter. Let $S'' : \left(\mathcal{X}''_{N,T}, \mathcal{X}''_{N,T'}, \hat{b}'', \hat{p}'', P_{S''}\right)$ denote the sample selected for replay. Its loss function is

$$\mathcal{L}_{replay}(\cdot,\cdot) = \mathcal{L}\left(F(\mathcal{X}''_{N,T}[b^{''k}], A[b^{''k}], \Theta[p^{''k}]), \mathcal{X}''_{N,T'}\right) \tag{19}$$

where the values of $b^{''k}$ and $p^{''k}$ are obtained from $\hat{b}''$ and $\hat{p}''$ through Eq. (7)-(8) by replacing $\hat{b}''$ for $\hat{b}$ and $\hat{p}''$ for $\hat{p}$.

***Co-optimized Variable-Parameter Loss**:* In practice, there is a balance between adjusting and stabilizing the selected

variables and model parameters. On the one hand, adjusting more aggressively introduces more combinations of selection. On the other hand, it makes the selection harder to converge. Stabilizing the masks aggressively accelerates convergence, but it may lead to suboptimal forecast performance.

To address this issue, we introduce a new *co-optimized variable-parameter loss* function $\mathcal{L}_{sum}$. This loss consists of three components: the main loss $\mathcal{L}_{main}$, which aims to find the optimal selection of of variables and model parameters; the replay loss $\mathcal{L}_{replay}$, which helps the model retain knowledge of previous samples; the random regularization of $\hat{b}$ and $\hat{p}$, which encourages sparse and diverse pruning decisions during training, acting as a stochastic exploration mechanism to avoid premature convergence to suboptimal masks. The co-optimized variable-parameter loss is defined as

$$\begin{aligned}\mathcal{L}_{sum}(\cdot,\cdot) =& \mathcal{L}_{main}(\cdot,\cdot) + \gamma_1 \mathcal{L}_{replay}(\cdot,\cdot) + \\ & \gamma_2 \|\hat{b}[r_1]\|_1 + \gamma_3 \|\hat{p}[r_2]\|_1 \end{aligned} \tag{20}$$

where the trainable variable vector $\hat{b}$ is optimized on a subset of variables chosen by a random mask $r_1 \in \{0,1\}^n$ with $\|r_1\|_0$ being a small number, the trainable parameter vector $\hat{p}$ is optimized on a subset of parameter dimensions chosed by another random mask $r_2 \in \{0,1\}^q$ with $\|r_2\|_0$ being a small number, and γ_1, γ_2, and γ_3 are the hyperparameters.

***Complexity Analysis**:* We analyze the inference complexity of our VIP model and compare it to the base STMF model. Let m be the number of chosen variables, n the total number of variables, l the length of the time period in the model input (output), L the number of attension layers, q the original number of attention dimensions, and q' the after-prune number of attention dimensions.

d) Time Complexity.: The total inference time complexity of the VIP model is:

$$\mathcal{O}\left(m \cdot l \cdot q + L \cdot m \cdot l \cdot q' \cdot q + m \cdot n \cdot l\right)$$

The three terms correspond to: (1) variable-wise feature embedding, (2) attention computation, and (3) extrapolation from the chosen variables to all variables. In contrast, the original STMF model has a time complexity of:

$$\mathcal{O}\left(n \cdot l \cdot q + L \cdot n \cdot l \cdot q^2\right)$$

By reducing the number of input variables from n to m, and attention dimensions from q to q', our model achieves significant computational savings, especially when $m \ll n$ and $q' \ll q$.

e) Memory Complexity.: The total memory complexity of the STCV model is:

$$\mathcal{O}\left(m \cdot l \cdot q + L \cdot q' \cdot q + n + L \cdot q\right)$$

where the terms corresponds to the embeddings, the pruned attention layers, and the trainable masks, $\hat{b}$ and $\hat{p}$. In comparison, the memory complexity of the STMF model is:

$$\mathcal{O}\left(n \cdot l \cdot q + L \cdot q^2\right)$$

The STCV model saves both space and time by working with a sparse set of input variables and a sparse set of parameters during inference, while still forecasting for all n variables.

IV. Trace-based Experimental Evaluation

A. Settings and Datasets

Datasets: We use five widely used public datasets, including four road traffic datasets (METR-LA, PEMSBAY, PEMS04 and PEMS08) and an air quality dataset (AQI) [1], for the road traffic forecast application and the air quality forecast application in the introduction, respectively. The details of data statistics are shown in Table II. We split METR-LA, PEMSBAY dataset into three subsets in 7:1.5:1.5 ratio for training, validation, and test and 3:1:1 for the other datasets.

These datasets are mapped to the STCV problem as follows: Each of the n locations in a dataset corresponds to a variable whose values are the time series of traffic (air quality) measurements at the location. After collecting the training/validation data subsets with cheap, temporary sensors from n locations (i.e., variables), we select m locations for permanent sensor deployment in the test data subsets for evaluation. These m *chosen locations* correspond to the chosen variables in the model input; the other $(m' = n - m)$ *unchosen locations* correspond to the missing variables not in the model input.

We define the sensor *deployment ratio* as $\frac{m}{n}$ and the deployment *sparcity ratio* as $1 - \frac{m}{n}$. In the application context of our evaluation description, we will use the term *location* for *variable* in the general STCV definition.

Hyperparameters: $m = 0.1n$ by default; hence, the deployment ratio is 10% and 90% by default — note that different settings of m and deployment (sparcity) ratios will be studied. $l = l' = 12$ for one hour forecast in the datasets. We use the Mean Absolute Error (MAE) as the loss function. All experiments are conducted on a server equipped with an AMD EPYC 7742 64-Core Processor @ 2.25 GHz, 500 GB RAM, and an NVIDIA A100 GPU with 80 GB memory.

We perform a systematic grid search over other hyperparameters, as summarized in Table III. These hyperparameters are categorized into two groups: primary and secondary. Each hyperparameter is tested across four candidate values, and we explore all combinations within each group independently.

The primary group includes the node pruning rate r_b, parameter pruning rate r_p, size $|\mathcal{R}|$ of the replay buffer, priority coefficient α, learning rate, and batch size. The secondary group contains the loss-related coefficients $\gamma_1, \gamma_2, \gamma_3$, the number $|r_1|_0$ of selected elements in the node vector, and the number $|r_2|_0$ of selected elements in the parameter vector, as defined in Eq. 20.

For the remaining STMF hyperparameters, we follow the default settings established in prior work [1], namely: $q = 152$, $d = 24$, $d_{\text{tod}} = 24$, $d_{\text{dow}} = 24$, $d_v = 80$, and $L = 6$.

Baselines: The baselines for comparison include spatio-temporal forecast models, such as STGODE [24], ST-SSL

[1]The datasets are provided in the GinAR github repository at https://github.com/GestaltCogTeam/GinAR/

TABLE I

PERFORMANCE COMPARISON OVER FIVE DATASETS, WHERE n IS THE TOTAL NUMBER OF LOCATIONS AND m IS THE NUMBER OF CHOSEN LOCATIONS.

Models	PEMS04			PEMS08			PEMSBAY			METRLA			AQI		
	deployment ratio $m/n = 10\%$**, sparsity ratio** $1 - m/n = 90\%$														
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
Matrix Factorization	92.13	141.41	124.29	75.80	131.17	82.16	6.51	14.29	21.09	15.61	33.39	48.51	67.63	115.43	130.18
STID	94.72	123.30	149.68	81.64	113.20	86.54	3.78	8.65	10.07	6.24	12.06	23.17	25.97	40.28	62.41
DSformer	40.31	59.18	25.62	38.79	55.34	32.61	3.72	9.06	10.23	5.02	10.15	17.69	23.17	35.72	57.31
STGODE	30.91	48.25	25.59	26.73	41.20	23.20	3.29	8.22	9.76	4.48	8.49	14.79	23.78	38.69	47.10
STSGCN	34.42	50.51	22.09	28.51	46.31	17.94	3.23	7.73	8.41	4.39	8.45	12.61	25.61	40.74	59.96
MegaCRN	36.14	52.17	23.42	34.52	52.75	24.64	3.54	8.25	9.23	4.58	8.72	13.54	22.61	35.74	53.28
ST-SSL	28.89	44.11	19.07	24.87	40.87	17.46	3.05	6.91	7.86	3.96	7.98	11.82	18.98	32.97	38.59
TESTAM	29.11	44.13	19.11	24.81	38.43	15.43	2.85	6.91	7.80	3.85	7.79	11.25	16.98	32.12	37.62
STAEFormer	28.11	43.32	18.35	24.31	39.25	15.63	2.72	6.58	6.70	4.07	8.43	12.05	17.47	32.60	39.09
STAEFormer + IGNNK	27.89	42.97	18.21	24.99	39.15	15.32	2.80	6.90	6.85	4.32	8.78	12.21	17.32	33.32	40.12
STAEFormer + STGNP	28.10	42.40	18.39	24.37	39.39	15.91	2.73	6.60	6.79	4.19	8.58	12.29	17.59	34.15	41.61
DCRNN+GPT4TS	31.42	46.17	21.27	31.78	46.22	22.96	3.31	7.69	8.82	4.29	8.31	12.18	21.78	34.28	50.64
DFDGCN+TimesNet	30.98	45.93	21.43	30.46	45.83	21.59	3.27	7.15	8.34	4.34	8.33	12.24	21.06	33.19	51.28
MTGNN+GRIN	30.61	45.88	21.04	29.15	42.79	21.33	3.22	7.26	8.45	4.25	8.29	12.14	20.02	34.15	50.23
FourierGNN+GATGPT	32.16	48.83	23.65	29.11	41.44	20.61	3.28	7.44	9.07	4.33	8.37	12.28	20.18	34.17	50.94
LGnet	36.29	55.12	23.74	35.48	52.06	25.71	3.67	8.92	9.83	4.79	9.15	14.38	23.24	35.54	57.18
GC-VRNN	30.06	45.12	20.43	28.46	41.98	20.54	3.12	7.32	7.94	4.32	8.35	12.29	19.24	32.91	48.73
TriD-MAE	29.54	44.23	20.05	26.18	39.25	16.43	3.02	7.02	7.63	4.11	8.22	11.92	18.04	32.68	45.76
BiTGraph	28.71	43.37	19.08	25.01	39.06	16.18	2.83	6.75	7.42	3.91	8.03	11.78	17.06	31.85	38.17
GinAR	28.20	42.82	18.31	24.83	38.87	15.82	2.77	6.43	6.94	3.87	7.84	11.25	16.83	30.97	36.30
MaxConnectivity+GinAR	28.11	42.65	18.14	24.89	39.07	15.99	2.84	6.80	7.23	3.93	8.09	12.21	16.97	31.35	37.89
MaxValue+GinAR	28.04	42.43	18.67	24.59	38.27	15.95	2.67	6.30	6.73	3.78	7.83	11.20	16.71	31.54	37.75
Grid+GinAR	28.87	42.31	18.06	24.70	38.84	15.74	2.78	6.59	6.93	3.90	7.99	11.36	16.89	31.14	37.52
Metaheuristics+GinAR	30.54	46.83	21.12	26.98	41.62	17.81	3.10	7.19	8.22	4.51	8.93	13.45	19.37	34.97	39.33
NIP+Pre	22.99	38.51	16.23	18.62	31.98	13.01	2.43	4.89	5.79	3.80	7.91	11.05	16.23	30.03	35.92
VIP wo. pretrain	23.99	39.27	16.05	19.87	32.73	13.19	2.44	4.99	5.80	3.52	7.14	10.20	16.98	31.34	35.21
VIP w. pretrain	22.93	38.26	16.06	18.59	31.51	12.63	2.38	4.76	5.64	3.64	7.68	10.94	15.89	29.22	35.01

TABLE II

BASIC STATISTICS OF THE DATASETS USED IN OUR EXPERIMENTS.

Dataset	AQI	PEMS04	PEMS08	PEMS-BAY	METR-LA
Area	China	CA, USA			
Time Span	1/1/2015 - 12/31/2022	1/1/2018 - 2/28/2018	7/1/2016 - 8/31/2016	1/1/2017 - 5/31/2017	3/1/2012 - 6/30/2012
Time Interval	1 hour	5 min			
Number of Locations, n	350	307	170	325	207
Number of Time Intervals	59,710	16,992	17,856	52,116	34,272

TABLE III

HYPERPARAMETER SETTINGS WITH THE DEPLOYMENT RATE OF 10% ON ALL DATASETS, WHERE THE BOLD VALUES REPRESENT THE BEST SETTING

Group	Hyperparameter	Value 1	Value 2	Value 3	Value 4
primary	r_b	0.05	**0.10**	0.20	0.40
	r_p	0.01	0.02	**0.05**	0.10
	$\|\mathcal{R}\|$	288	288*3	**288*7**	288*30
	α	0.2	0.4	**0.6**	0.8
	batch size	8	16	**64**	128
	learning rate	10^{-2}	$\mathbf{10^{-3}}$	5×10^{-4}	10^{-4}
secondary	γ_1	0.1	0.5	**1.0**	2.0
	γ_2	0.1	0.5	**1.0**	2.0
	γ_3	0.1	0.5	**1.0**	2.0
	$\|r_1\|_0$	0	1	**2**	5
	$\|r_2\|_0$	0	**1**	2	5

[25], STID [18], STSGCN [16], DSformer [26], MegaCRN [17], STAEFormer [1], and TESTAM [2]; two-stage spatial extrapolation models, which first perform spatio-temporal forecast (using past measurements from the m chosen locations to predict the future measurements at the m locations) and then perform spatial extrapolation (using the predicted future measurements at the m locations to extrapolate the future measurements at all n locations), such as DCRNN[27]+GPT4TS[28], DFDGCN[29]+TimesNet[30], MTGNN[31]+GRIN[6], FourierGNN[32]+GATGPT[33], STAEFormer+IGNNK [34] and STAEFormer+STGNP [35]; and the spatio-temporal imputation models, such as LGnet [8], GC-VRNN [36], TriD-MAE [7], BiTGraph [5] and GinAR [11] on STMF with missing variables, which is most related to our work; and a non-deep-learning Matrix Factorization method [37]. Adapting the baselines to work for the problem of STCV, following GinAR [11], we set the values of unchosen variables in their input value matrix to 0s.

In addition, we extend comparison to cover sensor placement strategies from the wireless sensor network domain and other heuristic approaches by including four two-phase baselines, with the first phase selecting sensor locations [38], [39], [11] and the second phase using GinAR [11] to perform fore-

cast. These baselines are referred to as Grid[38]+GinAR[11], Metaheuristics[39]+GinAR[11], MaxConnectivity+GinAR[11], and MaxValue+GinAR[11], where Grid segments the region (as shown in Fig. 4) into a set of rectangular grids and chooses the one with the highest connectivity within each grid for sensor deployment, Metaheuristics optimize sensor placement using a genetic algorithm (GA) [40], MaxConnectivity chooses the locations with the highest connections (node degrees), and MaxValue chooses the locations with the highest values (e.g., traffic rates) on average. Finally, we present a hybrid model, VIP+Pre, where half of the sensor locations are selected by MaxValue (which could be due to a government policy) and then the other half are selected by our proposed VIP. It covers the case where some chosen variables are pre-determined and others can be optimally selected.

For all baselines, we use the recommended hyperparameters from their GitHub repository. Before feeding a value matrix as input to the model, we preprocess the value matrix using z-score normalization, standardizing each value by subtracting the mean flow rate and dividing by the standard deviation.

Accuracy Metrics: We evaluate the forecast accuracy of the proposed work and the baseline models by Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE). Given the ground truth $\mathcal{X}_{i,j}$ and the forecast result $\hat{\mathcal{X}}_{i,j}$, $i \in N$, $j \in T'$, MAE at time $j = \frac{1}{n}\sum_{i=1}^{n}\left|\mathcal{X}_{i,j} - \hat{\mathcal{X}}_{i,j}\right|$, MAPE at time $j = \left(\frac{1}{n}\sum_{i=1}^{n}\left|\frac{\mathcal{X}_{i,j}-\hat{\mathcal{X}}_{i,j}}{\mathcal{X}_{i,j}}\right|\right) \times 100\%$, and RMSE at time $j = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\mathcal{X}_{i,j} - \hat{\mathcal{X}}_{i,j}\right)^2}$.

B. Forecast Accuracy

Our VIP model includes two variants: VIP without pretraining, denoted as VIP wo. pretrain, and VIP with pretraining, denoted as VIP w. pretrain. The VIP w. pretrain variant builds upon a pretrained STMF model, as described in Section III-A, where the STMF model is first trained to convergence with the best validation. In contrast, VIP wo. pretrain uses the same base STMF architecture without its parameters being pretrained before VIP training (which will train those parameters). Table I shows that our VIP w. pretrain consistently outperforms all baseline models in forecast accuracy, for example, achieving 30% improvement by VIP w. pretrain in RMSE against the best baseline result over the PEMSBAY dataset. VIP wo. pretrain performs second best on REMS04, PEMS08 and PEMSBAY, clearly better than all baselines, while the improvement on METRLA and AQI is less pronounced. Yet it takes less time for our VIP wo. pretrain to be trained than most baselines as we will demonstrate in the next subsection. The non-deep-learning matrix factorization method performs poorly probably because it is unable to capture complex spatial-temporal dependencies. Among all the spatial-temporal models, STAEFormer [1] achieves the best overall performance, benefiting from its embedding techniques (e.g., node embedding, time-of-day and day-of-week embedding).

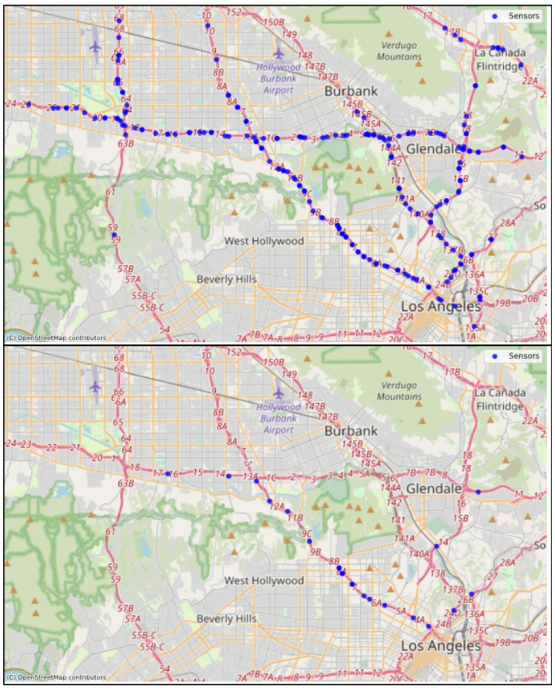

Fig. 4. Visualization of sensor locations with METR-LA dataset.

TESTAM [2] performs close to STAEFormer, thanks to its informative mixture-of-experts (MOE) [41] architecture. The two-stage spatial-temporal extrapolation models suffer from the error accumulation problem after the first stage of spatial-temporal multivariate forecast. The spatial-temporal imputation methods substitute zeros for the missing values, which could hurt the forecast accuracy as the substituted zeros and the true zeros (or low values) are mixed in the input. The two-phase baselines, which select sensor locations heuristically and then learn their forecast models based on the selection, underperform when compared to our VIP model, which integrates location selection and forecast training in the same iterative learning process.

C. A Case Demonstration of VIP's Location Selection

To help understand the problem of STMF with chosen variables (locations), we use a concrete example to demonstrate qualitatively and visually VIP's location selection in Fig. 4, where we show the full set N of locations of interest in the METRLA dataset on a map in the top plot and the subset M of locations that are selected by VIP in one execution with 10% deployment ratio in the bottom plot. The figure shows conceptually that while the chosen locations effectively covers (or are close to) major junctions, their density thins out disproportionally along different routes. For example, we observe more locations are chosen along the US-101 corridor (from 24B to 13A to 24) than along the I-5 highway (from 26B to 5 to 152). Such selection bias appears to arise from higher topological connectivity and greater traffic variability along the US-101 route, which connects central Los Angeles to Burbank. These characteristics make the US-101 corridor

TABLE IV
DIFFERENT ABLATION METHODS ON PEMS08 DATASET OVER THE AVERAGE HORIZON WHEN THE SENSOR DEPLOYMENT RATE $m/n = 10\%$.

Models	wo. pretrain			w. pretrain		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE
VIP	19.87	32.73	13.19	18.59	31.33	12.63
VIP wo. extra	22.10	38.42	14.39	21.19	37.78	13.47
VIP wo. b	20.63	33.47	13.91	19.22	31.79	12.85
VIP wo. p	20.77	33.22	15.90	18.93	31.48	13.88
VIP wo. replay	20.61	33.50	15.61	19.20	32.02	13.55

more informative under our pruning objective. This spatially diverse selection strategy enables more accurate forecasting even under a constrained budget of just 10% deployment as shown in Table I.

D. Efficiency Study

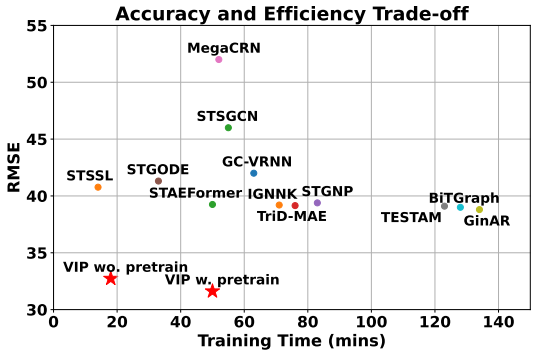

Fig. 5. Comparing our VIP models with the baselines in accuracy-efficiency tradeoff, over dataset PEMS08, with a deployment ratio of $m/n = 10\%$.

Fig. 5 compares our VIP models with the baseline models in efficiency-accuracy tradeoff. The batch size is uniformly set to 16. Our models achieve the best RMSE among all models with relatively low computational cost. This dual advantage of high accuracy and efficiency sets our models apart in practical forecast. Note that the training time of VIP w. pretrain includes the training time of its base STMF model. The reason for VIP's efficiency is that we prune locations and parameters with the variable mask and the parameter mask, which reduces both the input size and the network size during training.

E. Ablation Study

To investigate the effect of different components in VIP, we conduct ablation experiments over PEMS08 by taking one component away each time: **VIP wo. extra**: It does not use the dynamic extrapolation in Eq. (15). Instead, we use an MLP to map the representation $H_{N,T}[b]$ to $H_{N,T}$. **VIP wo. b**: It removes random regularization of the learnable mask $\hat{b}$. **VIP wo. p**: It removes random regularization of the learnable mask $\hat{p}$. **VIP wo. replay**: It removes the prioritized variable-parameter replay.

Table IV compares VIP (wo. or w. pretrain) with all the above variants, each missing a component, over PEMS08 dataset when the deployment ratio is $m/n = 10\%$. The degraded result of VIP wo. extra demonstrates the importance to capture the spatial correlations among the locations with the help of adjacency matrix and location embedding. The degraded result of VIP wo. b (VIP wo. p) demonstrates the importance of randomly-initiated sparse learnable node mask $\hat{b}$ (parameter mask $\hat{p}$), which generates different selections for optimization, instead of stable selections that lead to sub-optimal results. Finally VIP wo. replay also performs worse than VIP because it suffers from the catastrophic forgetting problem, where the model cannot remember the previous knowledge during pruning as new knowledge is learned.

F. Diversity Study

TABLE V
LOCATION SELECTION DIVERSITY AND FORECAST ACCURACY UNDER DIFFERENT REGULARIZATION METHODS OVER ALL BATCHES FROM THE PEMS08 DATASET WITH 10% DEPLOYMENT RATIO.

Regularization	Jaccard Distance	VIP w. pretrain		
		MAE	RMSE	MAPE
VIP w. L1	2.8%	18.59	31.33	12.63
VIP w. L2	2.3%	19.19	32.59	13.31
VIP w. ElasticNet	2.4%	19.07	32.11	13.02
VIP wo. Reg	1.5%	20.25	33.90	14.53

In order to justify how L1 norm in Eq. (20) promotes diversity in location selection, we present experiment results in Table V that compares different regularization methods, including our choice of VIP with L1 regularization (VIP w. L1) and alternative choices of VIP with L2 regularization (VIP w. L2), VIP with Elastic Net regularization [42] (VIP w. ElasticNet), and VIP without any regularization (VIP wo. reg). We focus on two metrics: the forecast accuracy and the Jaccard Distance [43], which is defined to measure the diversity of a dataset, in our case, the diversity of the location selections (i.e., variable masks) across all batches in all iterations. Its formula is

$$\text{Jaccard Distance} = 1 - \frac{2}{K \cdot B(B-1)} \sum_{k=1}^{K} \sum_{1 \le t < t' \le B} \frac{|b_t^k \cap b_{t'}^k|}{|b_t^k \cup b_{t'}^k|}$$

where K is the total number of iterations to reach the 10% deployment ratio, B is the number of batches for each iteration, b_t^k and $b_{t'}^k$ are variable masks for the t-th batch and the t'-th batch in the k-th iteration.

The experiment results show that VIP with L1 regularization generates variable masks with the greatest Jaccard Distance, indicating better location selection diversity, and thus achieves the best accuracy when compared to other regularization methods.

G. Scalability Study

We present the results of the scalability studies in Table VI and Table VII, comparing the memory footprint and inference latency of our VIP models against the most related GinAR model, respectively. All experiments were conducted on an NVIDIA B200 GPU with 180 GB memory, an Intel(R) Xeon(R) Platinum 8570 CPU, and 200 GB RAM. The batch size is set to 1, $l = l' = 12$, and the deployment ratio is 10% across all scales.

TABLE VI
MODEL PARAMETER SIZE (IN MB) UNDER VARYING NUMBERS OF VARIABLES.

Model	100	500	1,000	5,000	10,000	50,000	100,000
GinAR	0.67	2.02	4.61	61.29	222.14	5108.94	20217.44
VIP	0.68	1.09	1.64	7.88	20.18	298.59	1096.59

TABLE VII
INFERENCE TIME PER FORWARD PASS (IN MILLISECONDS) UNDER VARYING NUMBERS OF VARIABLES.

Model	100	500	1,000	5,000	10,000	50,000	100,000
GinAR	75.1	141.4	328.9	10086.76	OOM	OOM	OOM
VIP	3.47	3.53	4.35	4.35	6.08	53.86	188.6

The results show that as the total number n of variables increases from 100 to 100,000, VIP exhibits significantly better scalability in both memory efficiency and inference speed. In Table VI, GinAR's model size grows rapidly from 0.67 MB at 100 variables to 20,217.44 MB at 100,000 variables, while VIP's model size increases much more slowly—from 0.68 MB to only 1,096.59 MB under the same conditions. In Table VII, VIP's inference latency is 188.6 ms per forward pass even at 100,000 variables, whereas GinAR's inference latency is 10086.76ms at 5,000 variables and it experiences out-of-memory (OOM) errors beyond 10,000 variables due to excessive memory consumption.

These results clearly demonstrate that VIP scales more favorably than GinAR in both model size and inference time, making it highly suitable for real-time spatio-temporal forecasting in large scale systems.

H. Different Experience Replay Methods

To further evaluate the design of our prioritized variable-parameter replay (PVR), we substitute it with two other experience replay designs: random experience replay (RAND-ER) and inverse prioritized variable-parameter replay (IN-PVR). RAND-ER randomly replays the samples from the buffer. IN-PVR gives the high-loss samples a higher probability of being replayed. Specifically, we flip the priority P_S in Eq. (17) to $P_S = \mathcal{L}_{main}(\cdot,\cdot)$, and the probability for a sample to be selected is proportional to P_S.

Table VIII compares PVR and RAND-ER and IN-PVR in terms of average MAE, MAPE and RMSE on dataset PEMS08 with a deployment ratio of $m/n = 10\%$. It shows that PVR achieves the best performance because the dynamic change of variable/parameter masks causes model instability, the pruning of variables and parameters can result in forgetting the established patterns, and replaying low-loss samples helps mitigate that. That is also the reason that IN-PVR performs worse than RAND-ER because replaying high-loss samples will not do better in restoring model stability during pruning than replaying random samples.

I. RMSE under Different Sparsity Ratios

Fig. 6 presents the forecast accuracy results (in terms of RMSE) of VIP w. pretrain under different sparsity ratios, with

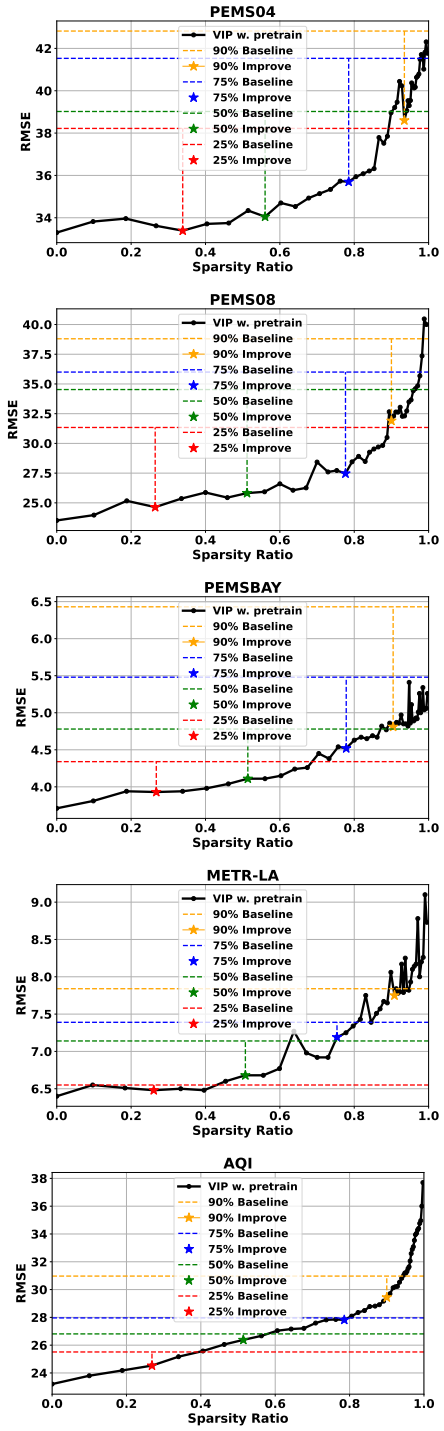

Fig. 6. Visualization of node sparsity in different datasets.

TABLE VIII
DIFFERENT EXPERIENCE REPLAY METHODS ON PEMS08 DATASET WITH A DEPLOYMENT RATIO OF $m/n = 10\%$.

Models	VIP wo. pretrain			VIP w. pretrain		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE
PVR	19.87	32.73	13.19	18.59	31.33	12.63
RAND-ER	20.47	33.62	13.53	19.47	31.92	13.08
IN-PVR	21.68	34.74	18.35	20.56	33.81	15.38

the pruning variable rate $r_b = 0.1$ and the pruning parameter rate $r_p = 0.05$. The baseline for comparison is GinAR, which is the most related work. Each plot in the figure is for a different dataset. The horizontal dashed lines show the RMSE values achieved by the baseline GinAR under sparsity ratios of 25%, 50%, 75%, and 90%, respectively. The stars on the curve show the RMSE values achieved by VIP w. pretrain under these ratios, respectively. Each vertical line segment from a star shows the RMSE improvement of VIP over the baseline. The results demonstrate that VIP is more accurate than GinAR under different sparsity ratios over different datasets. Note that a higher sparsity ratio means a lower deployment ratio.

V. RELATED WORKS

Most existing work focuses on the spatial-temporal forecast problem shown in Figure 1(a), including recurrent neural networks (RNNs) [44], convolutional neural networks (CNNs) or multiple layer perceptrons (MLPs) with graph neural networks (GNNs) [45], [27], [46], [3], [4], [32], neural differential equations (NDEs) [24], [47], transformers [1], [48], and most recent mixture-of-experts (MOE) [41] based model [2].

Some recent papers [49], [6], [5], [8], [7] investigate missing-data forecast as shown in Figure 1(b), and they attempt to recover the missing measurements at locations where sensors may sometimes fail. These methods still assume that sensors are deployed at all locations. When data from certain locations remain unavailable for long periods, the existing approaches would introduce incorrect temporal dependencies, leading to suboptimal data recovery performance. Spatial extrapolation methods, such as non-negative matrix factorization methods [37] and GNN methods [34], [50], [35], [51], could be used to estimate missing values of some variables based on available values from other variables within the same time period, but they do not model complex spatio-temporal dynamics for accuracy forecast into future time periods.

A few recent papers study STMF with missing variables, as shown in Figure 1(c), including Frigate [9], STSM [10], and GinAR [11]. In the context of traffic forecast, they assume that only some locations are deployed with permanent sensors. Among them, the most related work is GinAR, which performs much better than Frigate, and unlike STSM it does not require meta-knowledge about location environment (such as nearby malls). However, GinAR assumes the sensor locations (i.e., chosen variables) are given, without handling the issue of optimal senor placement, which is addressed by this paper.

The problem of sensor deployment has been extensively studied in the domain of wireless sensor networks (WSNs) [52], where the objective is to determine the optimal sensor placement to ensure sensing coverage and communication connectivity in a given spatial area, with each sensor having a sensing radius and a communication radius. Techniques in this domain include force-based models [53] (e.g., virtual force algorithm [54]), grid-based methods [55], and metaheuristics [39]. This line of research differs fundamentally from ours: (1) Their goal is to ensure that all interested areas/points/lines are wirelessly covered by the deployed sensors, whereas we do not address the coverage problem and our sensors produce data at the locations of their deployment, without a radius of wireless coverage. (2) Our goal is to use sensed information in the past from a selected subset of locations to predict the information about other locations in the future, whereas their work is not related to that. Therefore, their optimized locations for coverage will not align with our optimal locations for forecast.

Another line of research is the subset selection, which has emerged as a promising direction for improving the efficiency of machine learning by selecting representative data samples for training, thus reducing computational cost while maintaining performance. Notable approaches [56], [57], [58] focus on sample-based selection for text or images tasks, however, they barely study the spatio-temporal forecasting problem.

Finally, a component of our VIP framework adopts the experience replay (ER) method [22], which stores the historical samples in the buffer and replays the samples to the current model, which is a common technique in the continual learning domain [59] to mitigate catastrophic forgetting [21]. Recently, the prioritized experience replay (PER) [23] follows the ER method and prioritizes the samples to be replayed, achieving excellent performance primarily in the classification task in computer vision and natural language processing. However, its effectiveness in our spatial-temporal forecast context remains unclear.

In summary, this is the first work on STMF with chosen variables that performs variable selection and parameter reduction to optimize forecast performance in accuracy and efficiency.

VI. CONCLUSION

This paper studies a new problem of spatio-temporal multivariate time series forecast with chosen variables (STCV). Our variable-parameter iterative pruning model (VIP) jointly optimizes variable selection and parameter sparsification via a co-optimized training scheme. By integrating a dynamic extrapolation strategy and a prioritized replay mechanism, VIP maintains stable learning under iterative pruning. Experiments across five real-world datasets validate the effectiveness and efficiency of our proposed model.

REFERENCES

[1] H. Liu, Z. Dong, R. Jiang, J. Deng, J. Deng, Q. Chen, and X. Song, "Spatio-temporal adaptive embedding makes vanilla transformer sota for traffic forecasting," in *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2023, pp. 4125–4129.

[2] H. Lee and S. Ko, "TESTAM: A time-enhanced spatio-temporal attention model with mixture of experts," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=N0nTk5BSvO

[3] M. Li and Z. Zhu, "Spatial-temporal fusion graph neural networks for traffic flow forecasting," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 5, pp. 4189–4196, May 2021. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/16542

[4] R. Jiang, Z. Wang, J. Yong, P. Jeph, Q. Chen, Y. Kobayashi, X. Song, S. Fukushima, and T. Suzumura, "Spatio-temporal meta-graph learning for traffic forecasting," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 7, pp. 8078–8086, Jun. 2023. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/25976

[5] X. Chen, X. Li, B. Liu, and Z. Li, "Biased temporal convolution graph network for time series forecasting with missing values," in *The Twelfth International Conference on Learning Representations*, 2023.

[6] A. Cini, I. Marisca, and C. Alippi, "Filling the g_ap_s: Multivariate time series imputation by graph neural networks," *arXiv preprint arXiv:2108.00298*, 2021.

[7] K. Zhang, C. Li, and Q. Yang, "Trid-mae: A generic pre-trained model for multivariate time series with missing values," in *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2023, pp. 3164–3173.

[8] X. Tang, H. Yao, Y. Sun, C. Aggarwal, P. Mitra, and S. Wang, "Joint modeling of local and global temporal dynamics for multivariate time series forecasting with missing values," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 04, 2020, pp. 5956–5963.

[9] M. Gupta, H. Kodamana, and S. Ranu, "Frigate: Frugal spatio-temporal forecasting on road networks," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ser. KDD '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 649–660. [Online]. Available: https://doi.org/10.1145/3580305.3599357

[10] X. Su, J. Qi, E. Tanin, Y. Chang, and M. Sarvi, "Spatial-temporal forecasting for regions without observations," *Advances in Database Technology - EDBT*, vol. 27, no. 3, pp. 488–500, 2024.

[11] C. Yu, F. Wang, Z. Shao, T. Qian, Z. Zhang, W. Wei, and Y. Xu, "Ginar: An end-to-end multivariate time series forecasting model suitable for variable missing," *arXiv preprint arXiv:2405.11333*, 2024.

[12] Retail Sensing. (2024, Aug.) Vehicle detection: Ten ways compared. Accessed: 2025-06-24. [Online]. Available: https://www.retailsensing.com/people-counting/vehicle-detection-count-traffic/

[13] ChatGPT. (2025, Jun.) Chatgpt search: cost of installing equipment at a street intersection to detect and count vehicles. Accessed: 2025-06-29. [Online]. Available: https://www.chatgpt.com/

[14] Google. (2025, Jun.) Google search: cost of installing equipment at a street intersection to detect and count vehicles. Accessed: 2025-06-29. [Online]. Available: https://www.google.com/search?q=cost+of+installing+equipment+at+a+street+intersection+to+detect+and+count+vehicles

[15] ChatGPT. (2025, Jun.) Chatgpt search: cost of installing cheap temporary equipment at a street intersection to detect and count vehicles. Accessed: 2025-06-29. [Online]. Available: https://www.chatgpt.com/

[16] C. Song, Y. Lin, S. Guo, and H. Wan, "Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, 2020, pp. 914–921.

[17] R. Jiang, Z. Wang, J. Yong, P. Jeph, Q. Chen, Y. Kobayashi, X. Song, S. Fukushima, and T. Suzumura, "Spatio-temporal meta-graph learning for traffic forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 7, 2023, pp. 8078–8086.

[18] Z. Shao, Z. Zhang, F. Wang, W. Wei, and Y. Xu, "Spatial-temporal identity: A simple yet effective baseline for multivariate time series forecasting," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 4454–4458.

[19] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *arXiv preprint arXiv:1803.03635*, 2018.

[20] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[21] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[22] S. Zhang and R. S. Sutton, "A deeper look at experience replay," *arXiv preprint arXiv:1712.01275*, 2017.

[23] T. Schaul, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[24] Z. Fang, Q. Long, G. Song, and K. Xie, "Spatial-temporal graph ode networks for traffic flow forecasting," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 364–373.

[25] J. Ji, J. Wang, C. Huang, J. Wu, B. Xu, Z. Wu, J. Zhang, and Y. Zheng, "Spatio-temporal self-supervised learning for traffic flow prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 4, 2023, pp. 4356–4364.

[26] C. Yu, F. Wang, Z. Shao, T. Sun, L. Wu, and Y. Xu, "Dsformer: A double sampling transformer for multivariate time series long-term prediction," in *Proceedings of the 32nd ACM international conference on information and knowledge management*, 2023, pp. 3062–3072.

[27] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=SJiHXGWAZ

[28] T. Zhou, P. Niu, X. Wang, L. Sun, and R. Jin, "One fits all: Universal time series analysis by pretrained lm and specially designed adaptors," *arXiv preprint arXiv:2311.14782*, 2023.

[29] Y. Li, Z. Shao, Y. Xu, Q. Qiu, Z. Cao, and F. Wang, "Dynamic frequency domain graph convolutional network for traffic forecasting," in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 5245–5249.

[30] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long, "Timesnet: Temporal 2d-variation modeling for general time series analysis," in *The eleventh international conference on learning representations*, 2022.

[31] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "Connecting the dots: Multivariate time series forecasting with graph neural networks," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020, pp. 753–763.

[32] K. Yi, Q. Zhang, W. Fan, H. He, L. Hu, P. Wang, N. An, L. Cao, and Z. Niu, "Fouriergnn: Rethinking multivariate time series forecasting from a pure graph perspective," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[33] Y. Chen, X. Wang, and G. Xu, "Gatgpt: A pre-trained large language model with graph attention network for spatiotemporal imputation," *arXiv preprint arXiv:2311.14332*, 2023.

[34] Y. Wu, D. Zhuang, A. Labbe, and L. Sun, "Inductive graph neural networks for spatiotemporal kriging," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 5, 2021, pp. 4478–4485.

[35] J. Hu, Y. Liang, Z. Fan, H. Chen, Y. Zheng, and R. Zimmermann, "Graph neural processes for spatio-temporal extrapolation," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 752–763.

[36] Y. Xu, A. Bazarjani, H.-g. Chi, C. Choi, and Y. Fu, "Uncovering the missing pattern: Unified framework towards trajectory imputation and prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 9632–9643.

[37] D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," *Advances in neural information processing systems*, vol. 13, 2000.

[38] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava, "Coverage problems in wireless ad-hoc sensor networks," in *Proceedings IEEE INFOCOM 2001. Conference on computer communications. Twentieth annual joint conference of the IEEE computer and communications society (Cat. No. 01CH37213)*, vol. 3. IEEE, 2001, pp. 1380–1387.

[39] S. K. Gupta, P. Kuila, and P. K. Jana, "Genetic algorithm for k-connected relay node placement in wireless sensor networks," in *Proceedings of the Second International Conference on Computer and Communication Technologies: IC3T 2015, Volume 1*. Springer, 2016, pp. 721–729.

[40] J. H. Holland, "Genetic algorithms," *Scientific american*, vol. 267, no. 1, pp. 66–73, 1992.

[41] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," *arXiv preprint arXiv:1701.06538*, 2017.

[42] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 67, no. 2, pp. 301–320, 2005.

[43] R. Real and J. M. Vargas, "The probabilistic basis of jaccard's index of similarity," *Systematic biology*, vol. 45, no. 3, pp. 380–385, 1996.

[44] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, Feb. 2017. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/10735

[45] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," *arXiv preprint arXiv:1709.04875*, 2017.

[46] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, ser. IJCAI'19. AAAI Press, 2021, p. 1907–1913.

[47] Z. Liu, P. Shojaee, and C. K. Reddy, "Graph-based multi-ODE neural networks for spatio-temporal traffic forecasting," *Transactions on Machine Learning Research*, 2023. [Online]. Available: https://openreview.net/forum?id=Oq5XKRVYpQ

[48] J. Jiang, C. Han, W. X. Zhao, and J. Wang, "Pdformer: propagation delay-aware dynamic long-range transformer for traffic flow prediction," in *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, ser. AAAI'23/IAAI'23/EAAI'23. AAAI Press, 2023. [Online]. Available: https://doi.org/10.1609/aaai.v37i4.25556

[49] X. Li, H. Li, H. Lu, C. S. Jensen, V. Pandey, and V. Markl, "Missing value imputation for multi-attribute sensor data streams via message propagation," *Proceedings of the VLDB Endowment*, vol. 17, no. 3, pp. 345–358, 2023.

[50] C. Zheng, X. Fan, C. Wang, J. Qi, C. Chen, and L. Chen, "Increase: Inductive graph representation learning for spatio-temporal kriging," in *Proceedings of the ACM Web Conference 2023*, 2023, pp. 673–683.

[51] G. Appleby, L. Liu, and L.-P. Liu, "Kriging convolutional networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 3187–3194.

[52] R. Priyadarshi, B. Gupta, and A. Anurag, "Deployment techniques in wireless sensor networks: a survey, classification, challenges, and future research issues," *The Journal of Supercomputing*, vol. 76, pp. 7333–7373, 2020.

[53] H. Z. Abidin, N. M. Din, N. M. Radzi, and Z. I. Rizman, "A review on sensor node placement techniques in wireless sensor networks," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 7, no. 1, pp. 190–197, 2017.

[54] X. Yu, N. Liu, W. Huang, X. Qian, and T. Zhang, "A node deployment algorithm based on van der waals force in wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 9, no. 10, p. 505710, 2013.

[55] F. M. Al-Turjman, H. S. Hassanein, and M. Ibnkahla, "Quantifying connectivity in wireless sensor networks with grid-based deployments," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 368–377, 2013.

[56] O. Pooladzandi, D. Davini, and B. Mirzasoleiman, "Adaptive second order coresets for data-efficient machine learning," in *International Conference on Machine Learning*. PMLR, 2022, pp. 17 848–17 869.

[57] Y. Yang, H. Kang, and B. Mirzasoleiman, "Towards sustainable learning: Coresets for data-efficient deep learning," in *International Conference on Machine Learning*. PMLR, 2023, pp. 39 314–39 330.

[58] X. Xia, J. Liu, J. Yu, X. Shen, B. Han, and T. Liu, "Moderate coreset: A universal method of data selection for real-world data-efficient deep learning," in *The Eleventh International Conference on Learning Representations*, 2022.

[59] R. Aljundi, K. Kelchtermans, and T. Tuytelaars, "Task-free continual learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 11 254–11 263.