Treating Brain-inspired Memories as Priors for Diffusion Model to Forecast Multivariate Time Series

Muyao Wang   Wenchao Chen   Zhibin Duan   Bo Chen
National Laboratory of Radar Signal Processing Xidian University

Abstract

Forecasting Multivariate Time Series (MTS) involves significant challenges in various application domains. One immediate challenge is modeling temporal patterns with the finite length of the input. These temporal patterns usually involve periodic and sudden events that recur across different channels. To better capture temporal patterns, we get inspiration from humans' memory mechanisms and propose a channel-shared, brain-inspired memory module for MTS. Specifically, brain-inspired memory comprises semantic and episodic memory, where the former is used to capture general patterns, such as periodic events, and the latter is employed to capture special patterns, such as sudden events, respectively. Meanwhile, we design corresponding recall and update mechanisms to better utilize these patterns. Furthermore, acknowledging the capacity of diffusion models to leverage memory as a prior, we present a brain-inspired memory-augmented diffusion model. This innovative model retrieves relevant memories for different channels, utilizing them as distinct priors for MTS predictions. This incorporation significantly enhances the accuracy and robustness of predictions. Experimental results on eight datasets consistently validate the superiority of our approach in capturing and leveraging diverse recurrent temporal patterns across different channels.

1   Introduction

Multivariate time series (MTS) plays an important role in a wide variety of domains, including internet services [Dai et al., 2021] industrial devices [Finn et al., 2016, Oh et al., 2015] , health care [Choi et al., 2016a,b], finance [Maeda et al., 2019, Gu et al., 2020] and so on. With the development of neural networks, deep learning models, such as CNN-based models [Wu et al., 2022], Transformer-based models [Liu et al., 2023, Zhou et al., 2022, 2021] and Linear-based models [Zeng et al., 2023] have been applied to MTS and achieved better performance compared with traditional statistical methods [Box et al., 2015, Ariyo et al., 2014] in MTS forecasting tasks in terms of their higher capacity. Besides the advanced design of model architectures, some researchers have focused on diffusion theory for MTS forecasting and achieved promising results recently [Shen et al., 2024, Shen and Kwok, 2023, Rasul et al., 2021, Alcaraz and Strodthoff, 2022, Tashiro et al., 2021].

Despite the remarkable progress achieved by employing advanced methods [Nie et al., 2022, Zeng et al., 2023, Shen et al., 2024] to explore long-range temporal dependencies in MTS, predicting future observations over long time steps from limited past examples remains a challenging problem. One contributing factor to this challenge is the temporal information is extracted solely from the lookback window and not from the entire time series. Similar challenges are encountered in language modeling, where model performance is constrained by context length [Yang et al., 2020, Wang et al., 2020]. Substantial efforts have been invested in developing external knowledge modules [Bulatov et al., 2022, Guu et al., 2020, Gao et al., 2023], such as memory module, for language modeling to extend context information.

While memory modules have shown effective progress in Natural Language Processing (NLP), to the best of our knowledge, few studies have explored their adaptation to MTS forecasting tasks. Adapting a memory module to MTS is not straightforward, mainly because multivariate time series typically encompass multiple channels with specific connections among them [Nie et al., 2022, Liu et al., 2023]. As exemplified in Fig. 1 through the green and blue boxes, it is evident that the same temporal patterns can recur in different channels. To effectively capture the potential recurrent characteristics of temporal patterns across different channels, the designed MTS memory module should be shared among different channels. Simultaneously, temporal patterns in MTS can often be categorized into general patterns and special patterns, as illustrated in Fig.1. The former, such as cyclic variations, tends to occur more frequently, while the latter, such as sudden events, occurs less frequently.

To simultaneously capture both general patterns and special patterns in time series, we draw inspiration from brain memory [Tulving et al., 1972, Renoult et al., 2012] to formulate a brain-inspired memory module. In line with the structure of brain memory, we construct two components within our memory module. The first component is semantic memory storing the knowledge information in the human brain, which learns the general temporal patterns between different channels, enabling the network to exhibit human-like generalization abilities across different channels. The second component is episodic memory storing the representative events in the human brain, which stores the special temporal patterns across different channels that are diffi-
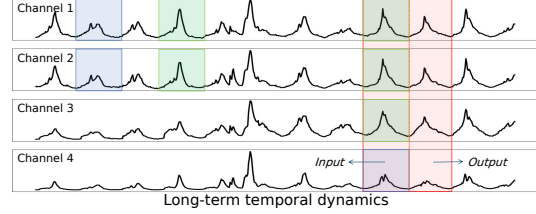

Figure 1: The red box represents the current MTS forecasting task. The blue box and green box represent diverse temporal patterns across different channels which can help channel 3 and 4 to forecast better. Because those temporal patterns have happened in other channels in history.

cult to predict, allowing the network to remember important exceptions. Subsequently, we introduce two pivotal processes in the brain-inspired memory module: 1) memory recall, which retrieves the relevant temporal patterns from brain-inspired memory to apply to the MTS forecasting task; 2) memory update, which continuously summarizes general patterns and selects special patterns during the training process and gradually consolidates the memories stored in brain-inspired memory.

In addition, the Bayesian generative model can not only naturally incorporate these memories as prior knowledge into the generation process, but also model the uncertainty in temporal patterns caused by noise [Han et al., 2022, Salinas et al., 2020, Wang et al., 2024]. Given the promising results achieved by diffusion models in MTS forecasting [Shen et al., 2024], we advocate using a diffusion model to encompass this non-deterministic temporal information, thereby enhancing predictive performance across various channels. Once our network possesses brain-inspired memory akin to human cognition, we can leverage them as priors for a diffusion model. By doing so, we can construct a generative model named **B**rain-**i**nspired **m**emory-augmented **Diff**usion Model (**Bim-Diff**), effectively modeling general and special temporal patterns. The main contributions of our work are summarized as follows:

- We propose a brain-inspired memory consisting of semantic and episodic memory, which can capture general and special temporal patterns across different channels within the MTS.
- We design efficient update and recall mechanisms for the proposed memory module, enabling it to capture and utilize two distinct types of temporal patterns effectively.
- We develop Bim-Diff, an efficient probabilistic model that integrates the two memories into the diffusion framework as the conditional prior, for MTS forecasting.
- Experiments on eight real-world datasets illustrate the efficiency of our model on the MTS forecasting task. Specifically, Bim-Diff ranks top-1 under the average ranking setting among the advanced models for comparison in the main experiment.

2 Related Work

In recent decades, the field of MTS forecasting has evolved significantly. It has transitioned from conventional statistical approaches such as ARIMA [Ariyo et al., 2014] and machine learning techniques like GBRT [Friedman, 2001] towards more advanced deep learning-based solutions, including Recurrent Neural Networks [Lai et al., 2018], Convolutional Neural Networks [Bai et al.,

2018], [Liu et al., 2021] and Multi-layer Perceptron based models [Zeng et al., 2023, Oreshkin et al., 2019, Challu et al., 2023]. More recently, several Transformer-based models have demonstrated remarkable efficacy in capturing temporal dependencies for MTS forecasting. Notable examples include Autoformer [Wu et al., 2021], which proposes a decomposition architecture with an Auto-Correlation mechanism to capture long-range temporal dependencies, and FEDformer [Zhou et al., 2022], which utilizes a seasonal-trend decomposition with frequency-enhanced blocks for the same purpose. PatchTST [Nie et al., 2022] performs time series prediction by patching the time series and making different channels to extract local semantic information independently. iTransformer [Liu et al., 2023] has recently achieved promising performance by simply inverting the vanilla Transformer to capture the channel relationships.

Very recently, diffusion models have also been developed for time series data. TimeGrad [Rasul et al., 2021] uses a recurrent neural network as a condition to predict in an autoregressive manner. Due to its prediction manner suffers from slow inference on long time series, CSDI [Tashiro et al., 2021] proposes a non-regressive method to solve the problem. Besides, SSSD [Alcaraz and Strodthoff, 2022] adjusts the mode structure and further accelerates the inference speed of the diffusion model. As SSSD and CSDI are mask-based condition methods, which suffer from the problem of boundary disharmony. To solve the problem, the non-autoregressive TimeDiff method [Shen and Kwok, 2023] uses future mixup and autoregressive initialization for conditioning. To make the diffusion model leverage the time series characteristic better, mrDiff [Shen et al., 2024] proposes a multi-resolution temporal structure and achieves state-of-the-art. However, due to the uncertainty in MTS forecasting, as well as challenges related to limited input length and long prediction horizons, overall, MTS forecasting remains a significant challenge.

3 Memory Augmented Diffusion Model

3.1 Problem Definition

Defining the MTS as $\boldsymbol{x} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_L\}$, where L is the duration of $\boldsymbol{x}$ and the observation at time t, $\boldsymbol{x}_t \in \mathbb{R}^N$, where N denotes the number of channels, thus $\boldsymbol{x} \in \mathbb{R}^{L \times N}$. multivariate time series forecasting aims to predict the value of $\boldsymbol{y}_{1:H}$, where H is the number of time steps in the future.

3.2 Denoising Diffusion Probabilistic Models

The Denoising Diffusion Probabilistic Model (DDPM)[Ho et al., 2020] is a kind of well-known probabilistic model, which usually consists of a fixed forward process that can map a complex data distribution to an easy prior distribution, such as a normal standard distribution, and a learnable backward denoising process that can generate a sample from the complex data distribution. We specify the forward process conditional distributions as:

$$q\left(\boldsymbol{x}^k \mid \boldsymbol{x}^{k-1}\right) = \mathcal{N}\left(\boldsymbol{x}^k; \sqrt{1-\beta_k}\boldsymbol{x}^{k-1}, \beta_k \boldsymbol{I}\right), \quad k = 1, \ldots, K \quad (1)$$

where at the kth step, x^k is generated by corrupting the previous iterate x^{k-1}(scaled by $\sqrt{1-\beta_k}$) with standard normal distribution(with variance $\beta_k \in [0,1]$). The formula (1) admits a closed-form sampling distribution with an arbitrary step k: $q\left(\boldsymbol{x}^k \mid \boldsymbol{x}^0\right) = \mathcal{N}\left(\boldsymbol{x}^k; \sqrt{\bar{\alpha}_k}\boldsymbol{x}^0, (1-\bar{\alpha}_k)\boldsymbol{I}\right)$, where $\bar{\alpha}_k = \prod_{s=1}^k \alpha_s$ and $\alpha_k = 1 - \beta_k$. Such formula corresponds to a tractable forward process posterior:

$$q\left(\boldsymbol{x}^{k-1} \mid \boldsymbol{x}^k, \boldsymbol{x}^0\right) = \mathcal{N}\left(\boldsymbol{x}^{k-1}; \tilde{\mu}(\boldsymbol{x}^k, \boldsymbol{x}^0), \tilde{\beta}_k \boldsymbol{I}\right) \quad (2)$$

where $\tilde{\mu} := \frac{\sqrt{\alpha_k}(1-\bar{\alpha}_{k-1})}{1-\bar{\alpha}_k}\boldsymbol{x}^k + \frac{\sqrt{\bar{\alpha}_{k-1}}\beta_k}{1-\bar{\alpha}_k}\boldsymbol{x}^0, \tilde{\beta}_k := \frac{(1-\bar{\alpha}_{k-1})}{(1-\bar{\alpha}_k)}\beta_k$.

In DDPM[Ho et al., 2020], backward denoising is defined as a Markovian process. Specifically, at the kth denoising step, x^{k-1} can be sampled from the following conditional distribution:

$$p_\theta\left(\boldsymbol{x}^{k-1} \mid \boldsymbol{x}^k\right) = \mathcal{N}\left(\boldsymbol{x}^{k-1}; \mu_\theta(\boldsymbol{x}^k, k), \sum_\theta(\boldsymbol{x}^k, k)\right), \quad k = 1, \ldots, K \quad (3)$$

where the variance $\sum_\theta(\boldsymbol{x}^k, k)$ is usually fixed the same as $\tilde{\beta}_k$ in Eq.(2). Thus, to approximate the tractable forward process posterior, parameter θ is learned by minimizing the loss $\mathcal{L} = \left\|\mu_\theta(\boldsymbol{x}^k, k) - \tilde{\mu}\right\|^2$. According to the [Ho et al., 2020], the final loss function can be defined as:

$$\mathcal{L} = \mathrm{E}_{\boldsymbol{x}^0, \epsilon, k} \left\|\boldsymbol{x}^0 - \boldsymbol{x}_\theta(\boldsymbol{x}^k, k)\right\|^2 \quad (4)$$

3

which means uses a network $\boldsymbol{x}_\theta$ to obtain an estimate $\boldsymbol{x}_\theta(\boldsymbol{x}^k, k)$ of the clean data $\boldsymbol{x}^0$ given $\boldsymbol{x}^k$, the ϵ represents a random noise sampled from the normal standard distribution.

3.3 Memory-augmented Conditional Diffusion Models for Time Series Prediction

Theory framework: When using conditional diffusion models for time series prediction, the following distribution is considered [Shen et al., 2024, Rasul et al., 2021].

$$p_\theta\left(\boldsymbol{y}_{1:H}^{0:K} \mid \boldsymbol{c}\right) = p_\theta\left(\boldsymbol{y}_{1:H}^K\right) \prod_{k=1}^K p_\theta\left(\boldsymbol{y}_{1:H}^{k-1} \mid \boldsymbol{y}_{1:H}^k, \boldsymbol{c}\right), \boldsymbol{c} = \mathcal{F}\left(\boldsymbol{x}_{1:L}^0\right) \tag{5}$$

where $\boldsymbol{y}_{1:H}^K \sim \mathcal{N}(0,1)$, c is the condition, and $\mathcal{F}$ is a conditioning network that takes the past observations $\boldsymbol{x}_{1:L}^0$ as input. Different from the deterministic prior $\boldsymbol{c}$ used in the previous method [Shen et al., 2024], we propose a variational brain-inspired memory $\boldsymbol{m}$ as diffusion model priors to enhance the model performance. Thus, the prediction process of $\boldsymbol{y}_{1:H}^0$ can be defined as follow:

$$p_\theta\left(\boldsymbol{y}_{1:H}^0\right) = \iiint p_\theta\left(\boldsymbol{y}_{1:H}^{0:K} \mid \boldsymbol{c}\right) p_\theta(\boldsymbol{c} \mid \boldsymbol{m}) p(\boldsymbol{m} \mid \boldsymbol{x}_{1:L}^0, \boldsymbol{M}) d\boldsymbol{c} d\boldsymbol{m} d\boldsymbol{y}_{1:H}^{1:K} \tag{6}$$

The condition vector $\boldsymbol{c}$ can be sampled from a condition distribution $p_\theta(\boldsymbol{c} \mid \boldsymbol{m}) = \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{m}), \operatorname{diag}(\boldsymbol{\sigma}(\cdot)))$ and the $\boldsymbol{m}$ can be sampled from the conditional distribution: $q_\phi(\boldsymbol{m} \mid \boldsymbol{x}_{1:L}^0, \boldsymbol{M}) = \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{m}^e, \boldsymbol{m}^s), \operatorname{diag}(\boldsymbol{\sigma}(\cdot)))$. Specifically, $\boldsymbol{\mu}(\boldsymbol{m}) = \boldsymbol{W}_1(\boldsymbol{m}), \boldsymbol{\mu}(\boldsymbol{m}^e, \boldsymbol{m}^s) = \boldsymbol{W}_2(\boldsymbol{m}^e + \boldsymbol{m}^s)$, where $\boldsymbol{W}_1, \boldsymbol{W}_2 \in R^{L \times L}$ are learnable matrixes, and $\boldsymbol{m}^e, \boldsymbol{m}^s$ are recalled from memory blocks $\boldsymbol{M}$ by Eq.(15), Eq.(12) respectively, which will be introduced in the next section in detail, and $\operatorname{diag}(\boldsymbol{\sigma}(\cdot))$ is a learnable diagonal matrix $\boldsymbol{I}$.

Conditional denoising network: Inspired by [Shen and Kwok, 2023], we use the same future mixup strategy to produce the conditioning signal to guide the training as follows:

$$\boldsymbol{c}_{\mathrm{mix}} = \boldsymbol{m}^{mask} \odot \boldsymbol{c} + \left(1 - \boldsymbol{m}^{mask}\right) \odot \boldsymbol{y}_{1:H}^0 \tag{7}$$

where $\boldsymbol{m}^{mask} \in [0,1)^{H \times N}$ is a mixing matrix with each element randomly sampled from the uniform distribution on $[0,1)$, and $\odot$ is the Hadamard product.

Analogous to Eq.(3), the conditional denoising process at step k is denied by:

$$p_\theta\left(\boldsymbol{y}_{1:H}^{k-1} \mid \boldsymbol{y}_{1:H}^k\right) = \mathcal{N}\left(\boldsymbol{y}_{1:H}^{k-1}; \mu_\theta(\boldsymbol{y}_{1:H}^k, k, \boldsymbol{c}_{\mathrm{mix}}), \tilde{\beta}_k \boldsymbol{I}\right), \quad k = 1, \ldots, K \tag{8}$$

where $\boldsymbol{\mu}_\theta := \frac{\sqrt{\alpha_k}(1-\bar{\alpha}_{k-1})}{1-\bar{\alpha}_k}\boldsymbol{y}_{1:H}^k + \frac{\sqrt{\bar{\alpha}_{k-1}}\beta_k}{1-\bar{\alpha}_k}\boldsymbol{y}_\theta(\boldsymbol{y}_{1:H}^k, k, c_{\mathrm{mix}})$, $\boldsymbol{y}_\theta(\boldsymbol{y}_{1:H}^k, k, c_{\mathrm{mix}})$ is an estimation of input $\boldsymbol{y}_{1:H}^0$, and θ includes all parameters in the conditional denoising network. Finally, analogous to Eq.(4), θ is obtained by minimizing the following denoising objective:

$$\mathcal{L}_{\mathrm{condition}} = \mathrm{E}_{\boldsymbol{y}_{1:H}^0, \epsilon, k} \left\| \boldsymbol{y}_{1:H}^0 - \boldsymbol{y}_\theta(\boldsymbol{y}_{1:H}^k, k, \boldsymbol{c}_{\mathrm{mix}}) \right\|^2 \tag{9}$$

On inference, the ground truth $\boldsymbol{y}_{1:H}^0$ is no longer available. Hence, we simply set $c_{\mathrm{mix}} = c$ in Eq.(7). Based on the denoise process Eq.(8), the final prediction $\hat{\boldsymbol{y}}_{1:H}$ can be obtained from random noise following the below formula step by step:

$$\hat{\boldsymbol{y}}_{1:H}^{k-1} = \frac{\sqrt{\alpha_k}(1-\bar{\alpha}_{k-1})}{1-\bar{\alpha}_k}\hat{\boldsymbol{y}}_{1:H}^k + \frac{\sqrt{\bar{\alpha}_{k-1}}\beta_k}{1-\bar{\alpha}_k}\boldsymbol{y}_\theta(\hat{\boldsymbol{y}}_{1:H}^k, k, \boldsymbol{c}_{\mathrm{mix}}) + \tilde{\beta}_k \epsilon, \quad k = 1, \ldots, K \tag{10}$$

where the ϵ represents a random noise sampled from the normal standard distribution.

4 Capturing Temporal Patterns via Brain-inspired Memory

In this section, we will introduce Brain-inspired memory (Bim), equipped with the mechanism of memory recall and update strategy. The fundamental concept behind Bim draws inspiration from the memory mechanisms observed in the human brain [Tulving et al., 1972, Renoult et al., 2012]: 1) there are two advanced forms of memory in the human brain, specifically, semantic memory allows the summary of general conceptual information, and episodic memory allows the collection of detailed events; 2) In the task of forecasting MTS, the semantic memory can summary the general temporal patterns of the MTS, while the episodic memory can store the special temporal patterns of the MTS as we mentioned before.
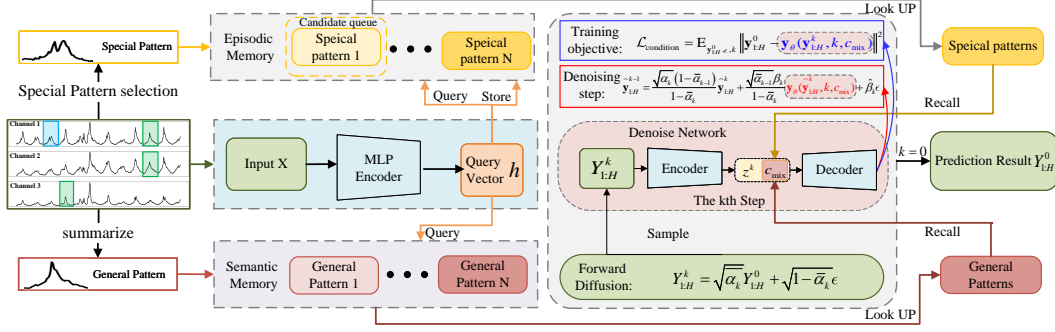
Figure 2: The framework of brain-inspired memory-augmented diffusion model.

4.1 Semantic Memory

As shown in the Fig 2, there are N_1 learnable semantic memory blocks in semantic memory module denoted as $\boldsymbol{M}^s = \{\boldsymbol{M}_i^s\}_{i=1}^{N_1}$, where i denotes the ith block and s means the semantic memory. The purpose of designing semantic memory is to encapsulate and summarize the recurrent general patterns across different channels in the historical series.

Storage strategy: The initialization of the semantic memory is random, and the memory is learnable during the training process. To summarize general patterns across different channels, each semantic memory block is shared among different channels and is specifically designed to store univariate temporal patterns referred to as general patterns. In line with our earlier discussion, we initialize the i-th block (pattern) of semantic memory with a learnable vector $\boldsymbol{M}_i^s \in T \times 1$, sampled from a normal Gaussian distribution.

Recall strategy: For the MTS forecasting task, the semantic memory initially computes the attention score between each channel and each general pattern stored in the semantic memory. Subsequently, an attention mechanism is employed to aggregate these patterns into a semantic memory $\boldsymbol{m}_i^s$ for predicting future values. The attention score can be calculated as follows:

$$\text{score}\left(\boldsymbol{M}_i^s, \boldsymbol{h}_j\right) = \text{Cosine}\left(\boldsymbol{M}_i^s, \boldsymbol{h}_j\right), \text{Cosine}\left(\boldsymbol{M}_i^s, \boldsymbol{h}_j\right) = \frac{\boldsymbol{M}_i^s \boldsymbol{h}_j}{\|\boldsymbol{M}_i^s\|\|\boldsymbol{h}_j\|} \tag{11}$$

where $\text{score}(\cdot)$ means the attention score function defined by cosine similarity; $\boldsymbol{h}_j$ denotes the jth channel of query vector $\boldsymbol{h}$ which is calculated by an MLP temporal encoder as shown in Fig 2; Then, to get the semantic memory for the MTS, we can further aggregate these general patterns with their attention weights, formulated as:

$$\boldsymbol{m}_j^s = \sum_{i=1}^{N_1} \frac{\text{score}(\boldsymbol{M}_i^s, \boldsymbol{h}_j)}{\sum_{i=1}^{N_1} \text{score}(\boldsymbol{M}_i^s, \boldsymbol{h}_j)} \boldsymbol{M}_i^s \tag{12}$$

where $\boldsymbol{m}_j^s \in \mathrm{R}^{T \times 1}$ has gathered the general patterns information from the semantic memory for the jth channel of MTS.

Update strategy: As we mentioned before, the semantic memory is calculated by general patterns stored in the semantic memory module. When we predict the future value, each learnable general pattern within the semantic memory takes into account the prediction results of each channel in the MTS during gradient backpropagation. Such an approach effectively accomplishes the objective of identifying general patterns across different channels. Furthermore, to keep memory items as compact as possible, at the same time as dissimilar as possible, we use two constraints [Gong et al., 2019] [Jiang et al., 2023], including a consistency loss L1 and a contrastive loss L2, denoted by

$$\begin{aligned} L_1 &= \sum_j^N \left\|\boldsymbol{h}_j - \boldsymbol{M}_j^{s,1}\right\|^2 \\ L_2 &= \sum_j^N \max\left\{\left\|\boldsymbol{h}_j - \boldsymbol{M}_j^{s,1}\right\|^2 - \left\|\boldsymbol{h}_j - \boldsymbol{M}_j^{s,2}\right\|^2 + \lambda, 0\right\} \end{aligned} \tag{13}$$

where $\boldsymbol{M}_j^{k,i} \in \mathrm{R}^{T \times 1}$ denotes the ith similar semantic pattern of channel j; λ denotes the margin between the first similar general pattern and the second similar general pattern; N indicates the number of MTS channels. More details about the update strategy are described in Appendix B.1

Algorithm 1 The update strategy of episodic memory

Set mini-batch size as N_{batch};
Initialize the candidate queue as $\{Q_i^e\}_{N_3}$, the episodic memory as $\{M_i^e\}_{N_2}$, the frequency matrix $F_{M_i^e}$;
Use **select** function Eq.(14) to select k special patterns from the mini-batch until the episodic memory is full;
repeat
    Sort the patterns by frequency matrix $F_{M_i^e}$ in the episodic memory $\{M_i^e\}_{N_2}$ and the last k element in the candidate queue $\{Q_i^e\}_{N_3}$ together;
    Pop out the last k elements in the candidate queue;
    Pop in the new k special patterns selected from the current mini-batch.
until no more new patterns
return global episodic memory $\{M_i^e\}_{N_2}$.

4.2 Episodic Memory

As shown in the Fig 2, there are N_2 episodic memory blocks in episodic memory module denoted as $M^e = \{M_i^e\}_{N_2}$, where i denotes the ith block and e means the episodic memory. Unlike semantic memory, episodic memory tends to focus on storing more special patterns within MTS directly, meaning no learnable parameters are included in it.

Storage strategy: The design of the episodic memory draws inspiration from prior works [Fortunato et al., 2019, Guo et al., 2020]. The key concept is to utilize a memory module to gather a subset of representative data samples. Specifically, the i-th block (pattern) of the episodic memory is a vector of univariate series selected from the query vector of MTS, denoted as $\boldsymbol{h}$, and is named special pattern, which has been encountered during training:

$$M_i^e = \text{select}(\boldsymbol{h}) \tag{14}$$

where $\text{select}(\cdot)$ means the select function which is defined in update strategy; $\boldsymbol{h}$ denotes the latent feature of past MTS. The episodic memory is empty at first because it has never seen any MTS.

Recall strategy: Similar to the recall method in semantic memory, we first calculate the attention score according to the query vector $\boldsymbol{h}_j$. Then we select top-k similar results for aggregating steps, which is a little different from the semantic memory. It is because semantic memory aims to learn the general patterns during the training process, while episodic memory is designed to focus on finding the special patterns that can be referenced across different channels. The detail of recalling episodic memory is defined as follows:

$$\boldsymbol{m}_j^e = \sum_{i=1}^{K} \frac{\text{score}(M_i^e, \boldsymbol{h}_j)}{\sum_{i=1}^{K} \text{score}(M_i^e, \boldsymbol{h}_j)} M_i^e \tag{15}$$

the episodic memory $\boldsymbol{m}_j^e \in \mathrm{R}^{T \times 1}$ has gathered the historical special event information through recalling the episodic memory M^e for the jth channel.

Update strategy: Inspired by the fact that it is important for the human brain to remember those special examples he/she often recalls, we design a special pattern selection and frequency-based episodic memory update strategy.

$\boldsymbol{i}$) **special pattern selection:** When a batch of MTS is presented during training, we compute the loss of each MTS in the batch. Subsequently, we select the MTS with the largest loss as special examples. Each special example is then divided into n univariate series, denoted as special patterns, which are stored in n episodic memory blocks $\{M_i^e\}_n$, where n represents the number of channels in each MTS. This process defines the select function Eq.(14) mentioned earlier.

$\boldsymbol{ii}$) **frequency-based episodic memory update strategy**: It is acknowledged that the most frequently accessed special pattern is the most representative. In order to record how many times M_i^e was recalled during training, we use a frequency matrix denoted as $\boldsymbol{F}_{M_i^e}$. Traditionally, M_i^e is updated by replacing the least used pattern in the block if its capacity exceeds the limit N_2. However, since newly incoming patterns have a lower access frequency rate than previously incoming patterns, these new patterns are more likely to be replaced, and we define this issue as memory freshness degradation.

To address this issue, we introduce a circular candidate queue within the episodic memory, designed to store the special patterns from the past N_3 occurrences, denoted as $Q^e = \{Q_i^e\}_{N_3}$, with the constraint $N_3 \leq N_2$. It's worth noting that the frequency matrix $\boldsymbol{F}_{M_i^e}$ is reset to a zero matrix following each

Table 1: Multivariate prediction MAEs on the real-world time series datasets (subscript is the rank). CSDI runs out of memory on Electricity. Results of all baselines are from [Shen et al., 2024].

	NorPool	Caiso	Electricity	Weather	Exchange	ETTh1	ETTm1	Wind	avg rank
Bim-Diff	$\underline{0.604}\pm0.001_{(2)}$	$\mathbf{0.193}\pm0.002_{(1)}$	$0.248\pm0.001_{(3)}$	$\underline{0.320}\pm0.002_{(2)}$	$\mathbf{0.079}\pm0.001_{(1)}$	$\mathbf{0.413}\pm0.002_{(1)}$	$\mathbf{0.361}\pm0.001_{(1)}$	$\mathbf{0.673}\pm0.002_{(1)}$	$\mathbf{1.5}$
mr-Diff	$\underline{0.604}_{(2)}$	$0.219_{(5)}$	$0.252_{(4)}$	$0.324_{(3)}$	$0.082_{(4)}$	$0.422_{(3)}$	$0.373_{(3)}$	$\underline{0.675}_{(2)}$	$\underline{3.3}$
TimeDiff	$0.611_{(4)}$	$0.234_{(8)}$	$0.305_{(7)}$	$\mathbf{0.312}_{(1)}$	$0.091_{(9)}$	$0.430_{(4)}$	$\underline{0.372}_{(2)}$	$0.687_{(4)}$	4.9
TimeGrad	$0.821_{(20)}$	$0.339_{(19)}$	$0.630_{(22)}$	$0.381_{(16)}$	$0.193_{(21)}$	$0.719_{(23)}$	$0.605_{(23)}$	$0.793_{(23)}$	20.9
CSDI	$0.777_{(18)}$	$0.345_{(20)}$	-	$0.374_{(14)}$	$0.194_{(22)}$	$0.438_{(7)}$	$0.442_{(17)}$	$0.741_{(12)}$	15.7
SSSD	$0.753_{(15)}$	$0.295_{(12)}$	$0.363_{(13)}$	$0.350_{(10)}$	$0.127_{(18)}$	$0.561_{(19)}$	$0.406_{(12)}$	$0.778_{(20)}$	13.6
D^3VAE	$0.692_{(11)}$	$0.331_{(17)}$	$0.372_{(15)}$	$0.380_{(15)}$	$0.301_{(23)}$	$0.502_{(16)}$	$0.391_{(10)}$	$0.779_{(21)}$	16.0
CPF	$0.889_{(22)}$	$0.424_{(22)}$	$0.643_{(23)}$	$0.781_{(24)}$	$0.082_{(5)}$	$0.597_{(21)}$	$0.472_{(18)}$	$0.757_{(17)}$	18.8
PSA-GAN	$0.890_{(23)}$	$0.477_{(23)}$	$0.533_{(21)}$	$0.578_{(23)}$	$0.087_{(8)}$	$0.546_{(18)}$	$0.488_{(19)}$	$0.756_{(15)}$	18.8
N-Hits	$0.643_{(8)}$	$0.221_{(6)}$	$\underline{0.245}_{(2)}$	$0.335_{(5)}$	$0.085_{(6)}$	$0.480_{(10)}$	$0.388_{(8)}$	$0.734_{(10)}$	6.9
FiLM	$0.646_{(10)}$	$0.278_{(10)}$	$0.320_{(9)}$	$0.336_{(6)}$	$\mathbf{0.079}_{(1)}$	$0.436_{(6)}$	$0.374_{(4)}$	$0.717_{(7)}$	6.6
Depts	$0.611_{(4)}$	$0.204_{(3)}$	$0.401_{(20)}$	$0.394_{(11)}$	$0.100_{(11)}$	$0.491_{(14)}$	$0.412_{(14)}$	$0.751_{(14)}$	12.3
NBeats	$0.832_{(21)}$	$0.235_{(9)}$	$0.370_{(14)}$	$0.420_{(19)}$	$0.081_{(3)}$	$0.521_{(17)}$	$0.409_{(13)}$	$0.741_{(12)}$	13.5
iTransformer	$0.645_{(9)}$	$0.217_{(4)}$	$0.258_{(5)}$	$0.340_{(7)}$	$0.085_{(6)}$	$0.431_{(5)}$	$0.380_{(7)}$	$0.710_{(6)}$	6.1
PatchTST	$0.710_{(12)}$	$0.293_{(11)}$	$0.348_{(12)}$	$0.555_{(22)}$	$0.147_{(15)}$	$0.489_{(13)}$	$0.392_{(11)}$	$0.720_{(8)}$	13
FedFormer	$0.744_{(13)}$	$0.317_{(14)}$	$0.341_{(11)}$	$0.347_{(9)}$	$0.233_{(23)}$	$0.484_{(11)}$	$0.413_{(15)}$	$0.762_{(18)}$	14.3
Autoformer	$0.751_{(14)}$	$0.321_{(15)}$	$0.313_{(8)}$	$0.354_{(11)}$	$0.167_{(16)}$	$0.484_{(11)}$	$0.496_{(20)}$	$0.756_{(15)}$	13.8
Pyraformer	$0.781_{(19)}$	$0.371_{(21)}$	$0.379_{(16)}$	$0.385_{(17)}$	$0.112_{(13)}$	$0.493_{(15)}$	$0.435_{(16)}$	$0.735_{(11)}$	16
Informer	$0.757_{(16)}$	$0.336_{(18)}$	$0.383_{(17)}$	$0.364_{(12)}$	$0.192_{(20)}$	$0.605_{(22)}$	$0.542_{(21)}$	$0.772_{(19)}$	18.1
Transformer	$0.765_{(17)}$	$0.321_{(15)}$	$0.405_{(19)}$	$0.370_{(13)}$	$0.178_{(19)}$	$0.567_{(20)}$	$0.592_{(22)}$	$0.785_{(22)}$	20
SCINet	$\mathbf{0.601}_{(1)}$	$\mathbf{0.193}_{(1)}$	$0.280_{(6)}$	$0.344_{(8)}$	$0.137_{(17)}$	$0.463_{(9)}$	$0.389_{(9)}$	$0.732_{(9)}$	7.5
NLinear	$0.636_{(6)}$	$0.223_{(7)}$	$\mathbf{0.239}_{(1)}$	$0.328_{(4)}$	$0.091_{(9)}$	$\underline{0.418}_{(2)}$	$0.375_{(5)}$	$0.706_{(5)}$	4.9
DLinear	$0.640_{(7)}$	$0.497_{(24)}$	$0.336_{(10)}$	$0.444_{(20)}$	$0.102_{(12)}$	$0.442_{(8)}$	$0.378_{(6)}$	$0.686_{(3)}$	8.8
LSTMa	$0.974_{(24)}$	$0.305_{(13)}$	$0.444_{(19)}$	$0.501_{(21)}$	$0.534_{(24)}$	$0.782_{(24)}$	$0.699_{(24)}$	$0.897_{(24)}$	21.6

update of the episodic memory. The detailed memory updating process algorithm is described in Algorithm 1. More details can be found in Appendix B.1.

5 Model Training

Similar to conditional diffusion models, the optimization objective of our model can be computed as

$$Loss = \mathcal{L}_{\text{condition}} + \alpha_1 L_1 + \alpha_2 L_2 \tag{16}$$

where $\mathcal{L}_{\text{condition}}$ is defined by Eq.(9), L_1, L_2 is computed by Eq.(13); α_1, α_2 indicates the balance parameter of two constraints, which is chosen by grid search on the validation set.

6 Experiments

6.1 Baselines and Experimental Settings

We assess the effectiveness of our model on eight datasets for MTS forecasting, namely ETTh1, ETTm1, NorPool, Caiso, Wind, Weather, Electricity, and Exchange-rate [Shen et al., 2024]. Data preprocessing follows the approach outlined in previous work [Shen et al., 2024]. We employ Mean Absolute Error (MAE) and Mean Squared Error (MSE)[Shen et al., 2020], to measure the performance of MTS forecasting models. We conduct time series prediction experiments by comparing 23 recent strong prediction models. The summary details are provided in Appendix A.

6.2 Main Results

Table 1 showcases the comprehensive prediction performance, with the best results highlighted in boldface and the second-best results underlined. Evaluation results indicate that our proposed method outperforms other state-of-the-art approaches in most settings. Overall, its average ranking is better than all other baselines (which include the most recent diffusion models). We report the results and the standard deviation of our model performance under five runs with different random seeds in Table 1, which exhibits that the performance of Bim-Diff is stable.

6.3 Ablation Study

Effect of brain-inspired memory: In our model, we have two key memories: semantic memory and episodic memory. We conducted an ablation study on the ETTh1, Wind, and Weather datasets, as presented in Table 2. Here, "w/o" indicates the absence of a particular module, and "w/o both" refers

Table 2: Ablation study on three datasets and the best results are highlighted in bold.

Dataset			ETTh1			Wind			Weather		
predict_length			48	168	672	48	168	672	48	168	672
ours	MSE		**0.339**	**0.408**	**0.456**	**0.600**	**0.893**	**1.269**	**0.120**	**0.185**	**0.305**
	MAE		**0.372**	**0.414**	**0.461**	**0.490**	**0.659**	**0.819**	**0.151**	**0.224**	**0.320**
w/o semantic	MSE		0.341	0.413	0.465	0.609	0.904	1.281	0.128	0.195	0.310
	MAE		0.370	0.422	0.472	0.503	0.664	0.835	0.159	0.235	0.329
w/o episodic	MSE		0.342	0.414	0.460	0.610	0.899	1.275	0.125	0.187	0.306
	MAE		0.371	0.423	0.465	0.504	0.659	0.830	0.150	0.227	0.325
w/o both	MSE		0.352	0.430	0.477	0.627	0.930	1.298	0.136	0.206	0.322
	MAE		0.386	0.437	0.481	0.525	0.679	0.853	0.170	0.249	0.346
w/o shared memory	MSE		0.348	0.420	0.470	0.605	0.905	1.285	0.121	0.188	0.310
	MAE		0.379	0.427	0.476	0.495	0.662	0.844	0.160	0.235	0.331

to the conditional diffusion model without any memory modules, serving as the baseline. In the table, "ours" denotes the BimPGM without any ablations. We analyze the results shown in Table 2.

1) Comparing row 4 (baseline) with rows 2 and 3, it's evident that both the semantic memory and the episodic memory contribute significantly to our model's performance. This demonstrates that both modules effectively utilize temporal patterns recurring across different channels, thereby enhancing the representation capacity of the conditional diffusion model.

2) Comparing the results on the different datasets, it can be seen that the semantic memory exhibits greater universality than the episodic memory, and this alignment with our design intention is expected. Specifically, semantic memory is designed to summarize general temporal patterns across all channels. In contrast, episodic memory focuses on remembering the special patterns across different channels that are challenging to predict.

3) Comparing row 1 (Bim-Diff) with rows 2 and 3, it's evident that the combined memory improves model performance. This reaffirms that the two memories can indeed provide effective recurrent temporal patterns from distinct perspectives: general temporal patterns and special temporal patterns.

w/o shared memory update: In this experiment, we design a brain-inspired memory for each channel independently instead of using a shared memory across different channels. This approach deviates from the shared memory update strategy proposed in this paper. The primary aim of this experiment is to assess the shared memory's capability to capture recurrent temporal patterns across different channels. This study is labeled as "w/o shared-memory." It's worth noting that comparing the results in rows 1 and 5 in Table 2 validates that our global shared memory module effectively captures the potential recurrent characteristics of temporal patterns across different channels.

6.4 Analysis of Recurrent Patterns and Channel Correlations

To further validate the efficacy of our Brain-inspired Memory in capturing recurrent temporal patterns across diverse channels, we conducted experiments visualizing memory attention scores using the ETTm1 dataset, as depicted in Fig 3. The results in the figure demonstrate that distinct channels recall relevant memories with notable similarity. This phenomenon demonstrates that our memory module can capture temporal patterns effectively, and these patterns recur among different channels. Furthermore, we suppose that the attention score of the brain-inspired memory provides an interpretable and quantifiable method for evaluating correlations between channels. For instance, we can analyze channel correlations, such as those between channel 0 and channel 2, by examining the attention score distribution across different memories. Based on the attention score, it is reasonable for the neural network to infer that channel 0 exhibits strong correlations with channel 2.

6.5 Efficiency Analysis

Fig 4(c) shows the inference time of different kinds of diffusion models. As can be seen, the inference of Bim-Diff is more efficient than mr-diff and Time-diff. The inference efficiency of Bim-Diff over existing diffusion models is due to: (i) We replace the U-Net structure in the diffusion model with a simple MLP structure. (ii) According to TimeGrad [Rasul et al., 2021], ten forward process steps are enough for diffusion to forecast. Thus, with the help of the acceleration technique DDIM [Song et al., 2020], which further reduces the denoising steps to just one for our method. Consequently, our model can be trained and tested on a single Nvidia 3090 GPU effectively.

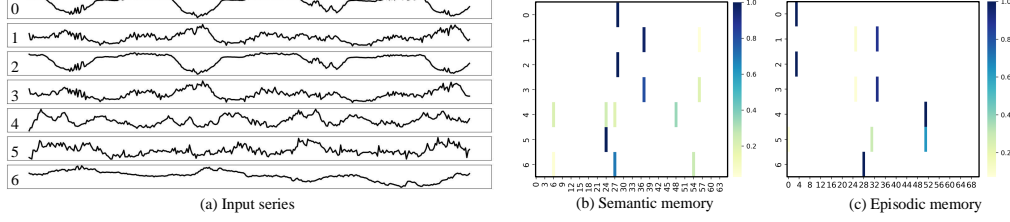(a) Input series (b) Semantic memory (c) Episodic memory

Figure 3: The visualization of input series, similar score matrix of semantic memory and episodic memory. The input series contains 7 channels. The size of the episodic memory is set to 70, and the size of the semantic memory is set to 64.
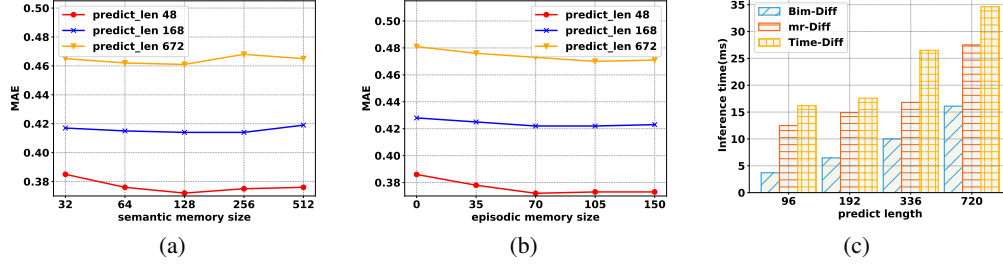

(a) (b) (c)

Figure 4: (a)-(b): The effect of the size of semantic (a) and episodic (b) memory on ETTh1 dataset. (c): Inference time (in ms) of various time series diffusion models with different prediction horizons (H) on the ETTh1(other method results come from [Shen et al., 2024]).

6.6 Effect of Hyper-parameters

We evaluate the effect of two hyper-parameters: block number of semantic memory and block number of episodic memory on the ETTh1 dataset.

Effect of memory size on semantic memory: In Fig.4(a), we set the memory size N_1 from 32 to 512 and evaluate MSE with different prediction lengths on ETTh1 dataset. To emphasize the impact of the memory size of the semantic memory, we conducted experiments without the episodic memory module. It becomes apparent that the semantic memory size N_1 follows a non-linear relationship with performance. An excessively large memory size can be detrimental because the semantic memory must learn to summarize patterns independently. In such cases, it might struggle to effectively capture useful patterns. Conversely, when the memory size is too small, it may not have the capacity to capture the essential patterns efficiently. Finding an optimal memory size is essential for striking the right balance between semantic retention and model efficiency.

Effect of memory size on episodic memory: In Fig.4(b), the memory size N_2 is set from 0 to 140 and evaluates MSE with different predicted lengths on ETTh1 dataset. In order to highlight the effect of the memory size of the episodic memory, we remove the semantic memory. The results in Fig.4(b) demonstrate that the performance tends to improve as the memory size increases. However, there is a point at which the performance gains start to diminish. This phenomenon underscores the effectiveness of the episodic memory we designed. This module stores special patterns for each channel, and when the memory size is sufficiently large, it can record all useful temporal patterns.

7 Conclusions

In this paper, we present a novel approach named Bim-Diff designed to capture general and special temporal patterns across different channels in multivariate time series forecasting tasks. Bim-Diff consists of two fundamental components: semantic memory and episodic memory, each capturing recurrent temporal patterns from different perspectives. Bim-Diff demonstrates excellence in extracting robust and expressive representations from MTS, leading to superior performance compared to other models in MTS forecasting tasks. Additionally, due to our effective simplification, the diffusion model for MTS forecasting can be trained efficiently. Empirical results from various MTS forecasting experiments provide compelling evidence of the effectiveness of our proposed model.

9

References

Liang Dai, Tao Lin, Chang Liu, Bo Jiang, Yanwei Liu, Zhen Xu, and Zhi-Li Zhang. Sdfvae: Static and dynamic factorized vae for anomaly detection of multivariate cdn kpis. *in WWW*, pages 3076–3086, 2021.

Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. *in NeurIPS*, 29, 2016.

Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. *in NeurIPS*, 28, 2015.

Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter Stewart. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. *in NeurIPS*, 29, 2016a.

Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F Stewart, and Jimeng Sun. Doctor ai: Predicting clinical events via recurrent neural networks. pages 301–318, 2016b.

Iwao Maeda, Hiroyasu Matsushima, Hiroki Sakaji, Kiyoshi Izumi, David deGraw, Atsuo Kato, and Michiharu Kitano. Effectiveness of uncertainty consideration in neural-network-based financial forecasting. *in AAAI*, pages 673–678, 2019.

Yuechun Gu, Da Yan, Sibo Yan, and Zhe Jiang. Price forecast with high-frequency finance data: An autoregressive recurrent neural network model with technical indicators. pages 2485–2492, 2020.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. *arXiv preprint arXiv:2210.02186*, 2022.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2023.

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. *In ICML*, 2022.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. *in AAAI*, 2021.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128, 2023.

George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

Adebiyi A Ariyo, Adewumi O Adewumi, and Charles K Ayo. Stock price prediction using the arima model. In *2014 UKSim-AMSS 16th international conference on computer modelling and simulation*, pages 106–112. IEEE, 2014.

Lifeng Shen, Weiyu Chen, and James Kwok. Multi-resolution diffusion models for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=mmjnr0G8ZY`.

Lifeng Shen and James Kwok. Non-autoregressive conditional diffusion models for time series prediction. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 31016–31029. PMLR, 23–29 Jul 2023. URL `https://proceedings.mlr.press/v202/shen23d.html`.

Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *International Conference on Machine Learning*, pages 8857–8868. PMLR, 2021.

Juan Miguel Lopez Alcaraz and Nils Strodthoff. Diffusion-based time series imputation and forecasting with structured state space models. *arXiv preprint arXiv:2208.09399*, 2022.

Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. Csdi: Conditional score-based diffusion models for probabilistic time series imputation. *Advances in Neural Information Processing Systems*, 34:24804–24816, 2021.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations*, 2022.

Xiaoxuan Yang, Bonan Yan, Hai Li, and Yiran Chen. Retransformer: Reram-based processing-in-memory architecture for transformer acceleration. In *Proceedings of the 39th International Conference on Computer-Aided Design*, pages 1–9, 2020.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. Recurrent memory transformer. *Advances in Neural Information Processing Systems*, 35:11079–11091, 2022.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR, 2020.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

Endel Tulving et al. Episodic and semantic memory. *Organization of memory*, 1(381-403):1, 1972.

Louis Renoult, Patrick SR Davidson, Daniela J Palombo, Morris Moscovitch, and Brian Levine. Personal semantics: at the crossroads of semantic and episodic memory. *Trends in cognitive sciences*, 16(11):550–558, 2012.

Xizewen Han, Huangjie Zheng, and Mingyuan Zhou. Card: Classification and regression diffusion models. *Advances in Neural Information Processing Systems*, 35:18100–18115, 2022.

David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *in International Journal of Forecasting*, 36(3): 1181–1191, 2020.

Muyao Wang, Wenchao Chen, and Bo Chen. Considering nonstationary within multivariate time series with variational hierarchical transformer for forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 15563–15570, 2024.

Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. pages 95–104, 2018.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

Minhao Liu, Ailing Zeng, Zhijian Xu, Qiuxia Lai, and Qiang Xu. Time series is a special sequence: Forecasting with sample convolution and interaction. *arXiv preprint arXiv:2106.09305*, 1(9), 2021.

Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*, 2019.

Cristian Challu, Kin G Olivares, Boris N Oreshkin, Federico Garza Ramirez, Max Mergenthaler Canseco, and Artur Dubrawski. Nhits: Neural hierarchical interpolation for time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 6989–6997, 2023.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34:22419–22430, 2021.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Dong Gong, Lingqiao Liu, Vuong Le, Budhaditya Saha, Moussa Reda Mansour, Svetha Venkatesh, and Anton van den Hengel. Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1705–1714, 2019.

Renhe Jiang, Zhaonan Wang, Jiawei Yong, Puneet Jeph, Quanjun Chen, Yasumasa Kobayashi, Xuan Song, Shintaro Fukushima, and Toyotaro Suzumura. Spatio-temporal meta-graph learning for traffic forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 8078–8086, 2023.

Meire Fortunato, Melissa Tan, Ryan Faulkner, Steven Hansen, Adrià Puigdomènech Badia, Gavin Buttimore, Charles Deck, Joel Z Leibo, and Charles Blundell. Generalization of reinforcement learners with working and episodic memory. *Advances in neural information processing systems*, 32, 2019.

Yunhui Guo, Mingrui Liu, Tianbao Yang, and Tajana Rosing. Improved schemes for episodic memory-based lifelong learning. *Advances in Neural Information Processing Systems*, 33:1023–1035, 2020.

Lifeng Shen, Zhuocong Li, and James Kwok. Timeseries anomaly detection using temporal hierarchical one-class network. *in NeurIPS*, 33:13016–13026, 2020.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

A Details of Experiments

A.1 Datasets statistics

The details of the datasets: (1) ETT dataset includes the time series of oil de-stationary factors and power load collected by electricity transformers from July 2016 to July 2018. ETTm1 is recorded every 15 minutes, and ETTh1 is recorded every 60 minutes. (2) The exchange dataset includes the panel data of daily exchange rates from 8 countries from 1990 to 2016. (3) The ILI dataset collects the ratio of influenza-like illness patients versus the total patients in one week, which is recorded weekly by the Centers for Disease Control and Prevention of the United States from 2002 and 2021. (4) Weather dataset includes meteorological time series with 21 weather indicators collected from the Weather Station of the Max Planck Biogeochemistry Institute in 2020 every 10 minutes. (5) The Electricity dataset includes the electricity consumption of 321 customers recorded hourly from 2012 to 2014. (6) NorPool dataset includes eight years of hourly energy production volume series in multiple European countries; (7) The Caiso dataset contains eight years of hourly actual electricity load series in different zones of California; (8) The Wind dataset contains wind power records from 2020-2021 at 15-minute intervals

Table 3: Summary of statistics of datasets

Datasets	Samples	channel number	Sample Rate	steps(H)
ETTh1	17420	7	60 min	168 (1 week)
ETTm1	69680	7	15 min	192 (2 days)
Exchange	7588	8	1 day	14 (two weeks)
Wind	48673	7	15 min	192(2 days)
Norpool	70128	18	60 min	720 (1 month)
Caiso	74472	10	60 min	720 (1 month)
Weather	52695	21	10 min	672 (1 week)
Electricity	26304	321	60 min	168 (1 week)

A.2 Evaluation metrics

We use two evaluation metrics which is usually used in time series forecasting tasks to measure the performance of predictive models. Let $X_{:,i} \in \mathbb{R}^{N \times 1}$ be the ground truth data of all channels at time step i, $X'_{:,i} \in \mathbb{R}^{N \times 1}$ be the predicted values, and Ω be indices of observed samples. The metrics are defined as follows.

Mean Absolute Error (MAE)

$$MAE = \frac{1}{|\Omega|} \sum_{i \in \Omega} \left| X_{:,i} - X'_{:,i} \right| \tag{17}$$

Mean Square Error (MSE)

$$MSE = \frac{1}{|\Omega|} \sum_{i \in \Omega} \left| X_{:,i} - X'_{:,i} \right|^2 \tag{18}$$

A.3 Baseline methods

The details of the baselines are as follows: The input lengths for all baseline experiments are chosen from $\{96, 336, 720, 960\}$, and the reported results are based on selecting the best-performing outcome within this range.

iTransformer: iTransformer is reproduced using the original paper's configuration in the official code or quoted from their original paper.

Other methods' results are cited from [Shen et al., 2024].

A.4 Implementation details

Our model: All the experiments are implemented with PyTorch and conducted on a single NVIDIA 3090 24GB GPU. Each model is trained by an ADAM optimizer using MSE loss. The input length for the Bim-Diff is chosen from $(96, 336, 720, 960)$. For the illness dataset, the input length is set to 14. The splitting ratio is set to 7:1:2 for train, val, and test set on the illness, weather, exchange, and Electricity dataset, which is the same as previous work. The splitting ratio is set at 3:1:2 for train, val, and test set on ETTh1 and ETTm1 datasets, which is the same as previous work.

B Additional Model Analysis

B.1 Update strategy analysis

B.1.1 Episodic memory

In our approach, we make an initial assumption that a new special pattern should be incorporated into the memory after each iteration. At the outset, the episodic memory is empty. As the memory fills up, we implement a sorting mechanism based on access frequency. Notably, only the last pattern in the candidate queue participates in this sorting process, as illustrated in Figure 5. When a new special pattern arrives, the last element in the candidate queue is popped out. Subsequently, the new pattern pops in the candidate queue. This strategy effectively addresses the memory solidification issue that we previously discussed.
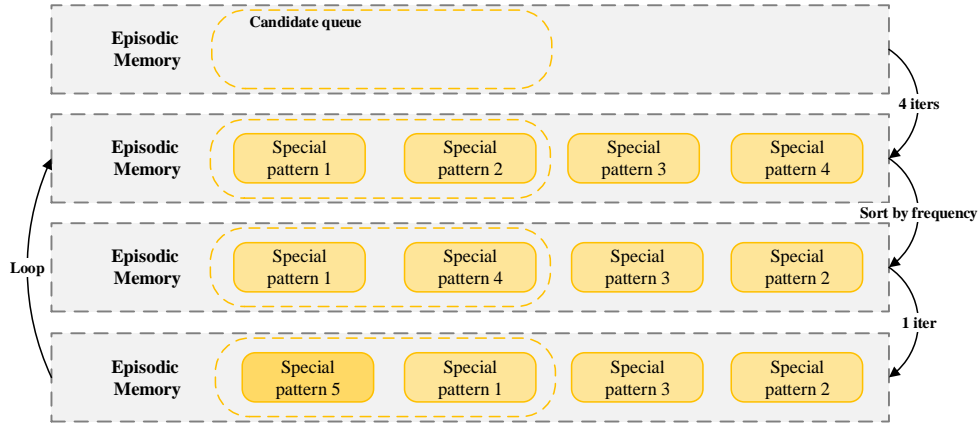

Figure 5: The details of episodic memory update strategy.
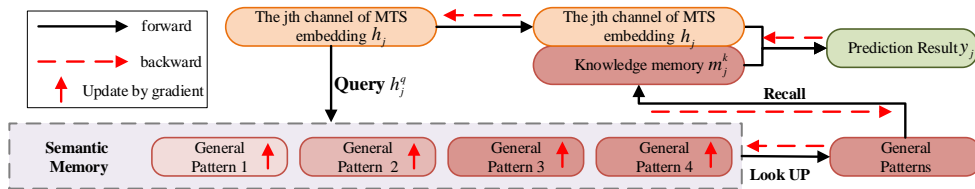
B.1.2 Semantic memory

Figure 6: The details of semantic memory update strategy.

The specific update process of the semantic memory is elaborated in Figure 6. Since each general pattern contributes to predicting the jth channel of the MTS $\boldsymbol{y}_j$, every pattern undergoes updates

14

during the backward process and accumulates information from the jth channel of the MTS. Because all general patterns are shared when utilized to forecast the outcomes of each channel, each pattern aggregates information from every channel, enabling it to capture general patterns across different channels effectively.

B.2 Complexity analysis

We have added computational complexity of the proposed approach compared to existing methods as shown below: The original Transformer has $O(L^2)$ complexity on both time and space, where L is the number of input lengths. iTransformer uses an inverted Transformer and achieves complexity $O(D^2)$ on both time and space, where D is the number of channels. Dlinear is a linear-based model, so the complexity of both time and space is $O(L)$. Because we replace the U-Net in the traditional diffusion model with an MLP, the complexity of our model is $O(DM)$ mainly caused by the calculation of attention score, which is (D is the number of channels, M is the memory size). In most cases, D, M is much smaller than L, so the complexity of our model is smaller than the transformer-based model.