Learning Gaussian Mixture Representations for Tensor Time Series Forecasting

Jiewen Deng1, Jinliang Deng1,2, Renhe Jiang2,∗1 and Xuan Song1,3

1Southern University of Science and Technology
2University of Technology Sydney
3The University of Tokyo
dengjw1@outlook.com, jinliang.deng@student.uts.edu.au
jiangrh@csis.u-tokyo.ac.jp, songx@sustech.edu.cn

Abstract

Tensor time series (TTS) data, a generalization of one-dimensional time series on a high-dimensional space, is ubiquitous in real-world scenarios, especially in monitoring systems involving multi-source spatio-temporal data (e.g., transportation demands and air pollutants). Compared to modeling time series or multivariate time series, which has received much attention and achieved tremendous progress in recent years, tensor time series has been paid less effort. Properly coping with the tensor time series is a much more challenging task, due to its high-dimensional and complex inner structure. In this paper, we develop a novel TTS forecasting framework, which seeks to individually model each heterogeneity component implied in the time, the location, and the source variables. We name this framework as GMRL, short for Gaussian Mixture Representation Learning. Experiment results on two real-world TTS datasets verify the superiority of our approach compared with the state-of-the-art baselines. Code and data are published on https://github.com/beginner-sketch/GMRL.

1 Introduction

Tensor time series (TTS) data occur as sequences of multi-dimensional arrays [Rogers et al., 2013; Jing et al., 2021]. For example, urban public transportation management involves data coming from multiple times, locations, and transportation modes, i.e., Taxi/Bike Inflow/Outflow [Zhang et al., 2016; Zhang et al., 2017], based on which compelling predictions are crucial to subsequent decision-making, especially for those urban-sensitive cases. Compared with multivariate time series (MTS), TTS involves an additional variable apart from the initial time and location variables, namely the source variable. The most significant difference between MTS and TTS is that MTS involves data in a matrix, where the rows represent the time variable and the columns define the location variable. In contrast, TTS concerns tensor data, which usually takes on more substantial heterogeneity and dynamics. Although identifying and extracting similar patterns is reasonable and typical for time series prediction, the components over variables can be dynamic in many real-world applications. Failure to explicitly obtain the quantitative form of patterns may mistakenly cause the model to capture the pattern change in TTS. We use an example of NYC traffic demand data in Figure 1 to illustrate it. We collect the historical data of the three frames/locations/sources and plot their data distribution in the upper part of Figure 1. Based on the characteristics (the mean and variance) of the data distributions, we can trivially draw a conclusion that the instance at 6 am is more similar to 12 am than 8 pm. This is consistent with the evolving patterns of observation in the lower part of Figure 1. Such distinguishability over the time variable also holds water over the location and source variables. Given the complex patterns, capturing heterogeneities arising from multiple variables is an essential problem in TTS forecasting.

Deep forecasting models have achieved great success in more effective time series forecasting. Graph neural network (GNN) [Wu et al., 2020b; Shao et al., 2022], temporal convolution networks (TCN) [Wu et al., 2020b], recurrent neural networks (RNN) [Lai et al., 2018], and attention mechanisms [Wu et al., 2021; Zheng et al., 2020] are well-used for prediction problems. Each individual or a combination of these manipulations performs well in modeling the characteristics of a single domain or spatial-temporal regularities. Such methods are more prone to focus on low-level hi-
erarchical information and are blind to the rich information and challenges brought by source variables. An increasing number of recent works are drawing forth the evolution from other modes as an auxiliary input, adding ancillary information to borrow similar historical patterns [Wu et al., 2020a; Fang et al., 2021; Han et al., 2021]. These practices, to a certain degree, bring mode-wise awareness. Nevertheless, they mainly help with the spatial-temporal parts, and the heterogeneity or the interactions brought by variables (i.e., time, location, and source) are still underexplored. Recently, cluster-wise learning breaks the independence between instances by exploiting the latent cluster information among samples. The learned representations are expected to retain higher-level heterogeneous information by taking advantage of additional prior information brought by clustering. However, existing approaches [Zhu et al., 2021; Duan et al., 2022] (i) neglect the dynamic nature of time series, assuming that both the representations and the members of the clusters will remain the same over time and (ii) the interactions over multiple variables as well, which are not applicable for intricate and complex TTS. Thus far, while evolution regularities in time series have been studied intensively, dynamic heterogeneity and interactions of multiple variables in TTS have yet to be adequately tackled.

Therefore, we are motivated to propose a novel TTS learning framework for accurate forecasting, namely Gaussian Mixture Representation Learning (GMRL). It consists of three steps: (i) incorporating the multiple variables information into the model through a TTS embedding (TTSE); (ii) designing a dynamic Gaussian mixture representation extractor to explicitly represent the complicated heterogeneous components among multiple variables; (iii) leveraging techniques of memory that enhances the learned representation to learn to distinguish and generalize to diverse scenarios. The contributions of this work are summarized as follows:

- We design a Gaussian Mixture Representation Extractor (GMRE), to explicitly disentangle and quantify the heterogeneity components in the time, the location, and the source variables.
- We propose a Hidden Representation Augmenter (HRA) to equip the learned representations with sequence-specific global patterns on the fly, intrinsically improving model adaptability to TTS.
- We conduct thorough experiments on two real-world datasets, one is traffic dataset containing four sources of transportation demands (i.e., Taxi/Bike Inflow/Outflow), another is air quality dataset containing three sources of pollutants (i.e., PM2.5, PM10, SO2). The results verify the superiority of our method over the state-of-the-arts.

### 2 Preliminaries

In this section, we formulate the tensor time series forecasting problem, and introduce the used notations (in Table 1).

**Problem 1** (Tensor Time Series Forecasting). Given $T$ consecutive time steps, $L$ discretized locations, and $S$ sources, the multi-source spatio-temporal data can be aggregated into a Tensor Time Series $X \in \mathbb{R}^{T \times L \times S}$. Given $X$, the tensor time series forecasting aims to forecast the next $O$ steps as $\hat{Y} \in \mathbb{R}^{O \times L \times S}$, which can be expressed by the following conditional distribution:

$$P(\hat{Y} | X) = \prod_{t=1}^{O} P(\hat{Y}_{t+1} | X)$$

### 3 Methodology

In this section, we elaborate on the technical details of the proposed framework Gaussian Mixture Representation Learning (GMRL), as demonstrated in Figure 2.

#### 3.1 Tensor Time Series Embedding

Since the evolution patterns of TTS are constrained by the location and the source variables, it is crucial to incorporate the variables’ information into the predictive model. Specifically, we generate location embedding and source embedding to encode corresponding variables into vectors that preserve static information, respectively expressed as $e^L_l \in \mathbb{R}^{d_l}$ and $e^S_s \in \mathbb{R}^{d_s}$. The above embedding vectors only provide static representations, which could not represent the dynamic correlations among locations and sources. Thus, we further utilize temporal embedding $e^T_t \in \mathbb{R}^{d_t}$ to encode each historical time step into a vector. To obtain the variable representations, we fuse the aforementioned embedding vectors in an additive manner. For location $l$ and source $s$ at time step $t$, the embedding is defined as: $e_{l,s,t} = e^L_l + e^T_t + e^S_s$. Then we get the TTSE ($E \in \mathbb{R}^{T \times L \times S \times d_e}$) of $L$ locations and $S$ sources in $T$ time steps. Next, the initial representation fed into subsequent modules is $H = [f(X), E]$, where $f(\cdot)$ represents linear transformation and $[\cdot]$ denotes concatenation operation.

#### 3.2 Gaussian Mixture Representation Extractor

The canonical cluster analysis method is used to reveal the underlying nature and patterns of data, enabling the classification of the internal correlation structure of data. However, existing time series modeling methods rely on cluster distinctions that do not necessarily indicate an optimal solution. Additionally, time series data feature a dynamic context, implying that the characteristics of the clusters, which describe real-time time series data, will change over time as a function of the dynamic context.

To overcome the above limitations, we develop a dynamic Gaussian Mixture Representation Extractor (GMRE) to explicitly capture the patterns of time series data based on the
Figure 2: Gaussian Mixture Representation Learning (GMRL): (i) taking a three-dimension TTS as input; (ii) multiple proposed Gaussian Mixture Representation Extractors (GMRE) and Temporal Encoders (TE) are stacked layer by layer; (iii) applying the skip connection to fuse the representation output from each GMRE-TE layer; (iv) a Hidden Representation Augmenter (HRA) is deployed to strengthen representation with global patterns; (v) a Predictor is used to generate the prediction for TTS.

underlying context. Moreover, we expect different channels to hold information about distinct contextual attributes, i.e., the factors of variation, exhibiting mutual independence and complementarity. For instance, some channels correspond to spatial identities, remaining invariant over time, while others correspond to temporal identities, displaying variations, potentially with different frequencies, over time. Motivated by this, we perform channel-wise clustering instead of the conventional sample-wise clustering, in order to separate the clusters for each channel. A practical advantage of injecting inter-channel independence lies in reducing the number of learnable parameters involved in the model, as modeling interactions among the channels is no longer necessary.

Given a representation \( H \in \mathbb{R}^{T \times L \times S \times d_h} \) derived from an input tensor \( X \), GMRE aims to learn a mixture of Gaussian distributions over each scalar value within \( H \). Technically, the probability distribution is conditioned on the observation (i.e., input tensor). However, for simplicity, we omit this condition in the following probability formulas. Firstly, we introduce a latent variable \( z[i] \in \{1, K\} \) to identify the membership (belonging to which of the \( K \) clusters) of each scalar value for each channel \( i \in \{1, d_h\} \). Given \( h \in \mathbb{R}^{d_h} \) (one of the \( T \times L \times S \) small cubes in Figure 2) from the representation \( H \), GMRE infers the probability distribution for each scalar value in \( h[i] \) (i.e., \( i \)-th channel of \( h \)). Formally, the probability of \( h[i] \) can be decomposed as follows:

\[
P(h[i]) = \sum_{k=1}^{K} P(z[i] = k) P(h[i] | z[i] = k)
\]

Then, we can specify the expressions for \( P(z[i] = k) \) and \( P(h[i] | z[i] = k) \). Formally, we obtain the prior distribution of \( z \) as follows:

\[
P(z[i] = k) = \alpha_k \left( H[\ldots, i] \right) = \frac{\exp(W^\alpha_{k,i} \cdot \text{vec}(H[\ldots, i]))}{\sum_k \exp(W^\alpha_{k,i} \cdot \text{vec}(H[\ldots, i]))}
\]

(3)

where the membership score associated with the \( k \)-th channel belonging to the \( k \)-th clusters is parameterized by a linear mapping with \( W^\alpha_{k,i} \in \mathbb{R}^{TLS} \). For \( P(h[i] | z[i] = k) \), we assume it follows a Gaussian distribution \( \mathcal{N}(h[i] | \mu_{k,i}, \sigma^2_{k,i}) \), where the mean \( \mu_{k,i} \) and variance \( \sigma^2_{k,i} \) are generated by:

\[
\begin{align*}
\mu_{k,i} &= W^\mu_{k,i} \cdot \text{vec}(H[\ldots, i]) + b^\mu_{k,i}, \\
\sigma^2_{k,i} &= \exp(W^\sigma_{k,i} \cdot \text{vec}(H[\ldots, i]) + b^\sigma_{k,i})
\end{align*}
\]

(4)

where \( W^\mu_{k,i}, W^\sigma_{k,i} \in \mathbb{R}^{TLS} \) are weights to project representation \( h[\ldots, i] \) to \( \mu_{k,i} \) and \( \sigma^2_{k,i} \), and \( b^\mu_{k,i}, b^\sigma_{k,i} \in \mathbb{R} \) are the corresponding biases. \( \exp(\cdot) \) ensures that \( \sigma^2_{k,i} \) remains greater than zero.

With \( P(z[i] = k) \) and \( P(h[i] | z[i] = k) \) available, we can derive the posterior probability \( P(z[i] = k | h[i]) \) as follows, using Bayes’ theorem:

\[
P(z[i] = k | h[i]) = \frac{P(z[i] = k) P(h[i] | z[i] = k)}{P(h[i])} = \frac{\alpha_k \left( H[\ldots, i] \right) \cdot \mathcal{N}(h[i] | \mu_{k,i}, \sigma^2_{k,i})}{\sum_k \alpha_j \left( H[\ldots, i] \right) \cdot \mathcal{N}(h[i] | \mu_{j,i}, \sigma^2_{j,i})}
\]

(5)

Following Eq. 5, we can assign the \( i \)-th channel of \( h \) to cluster \( k = \arg\max_k P(z[i] = k | h[i]) \).
Cluster Normalization. Inspired by the simplex and efficiency of normalization technology, we devise Cluster Norm to increase the distance between clusters while reducing the distance between samples within a cluster. Cluster-wise normalization is applied to each sample belonging to the \( k \)th cluster to get a set of representations as:

\[
\hat{H} = \begin{bmatrix} H[t, l, s, i] - \mu_{k,i} \\ \sigma_{k,i} + \varepsilon \end{bmatrix}_{t, l, s, i}
\]

where \( \varepsilon \) is a small constant to preserve numerical stability. Then, we can obtain the Gaussian mixture representation \( H^{(gm)} = [H, \hat{H}] \in \mathbb{R}^{T \times L \times S \times 2d_k} \).

Probability Regularization. To enable \( P(h[i]) \) to describe the empirical data properly, we must regularize the probability of empirical data under \( P(h[i]) \). Following the convention, we alternatively minimize minus log \( P(h[i]) \), equivalent to maximizing the \( P(h[i]) \):

\[
\log P(h[i]) = \log \sum_z P(h[i], z[i]) = \sum_z P(z[i]|h[i]) \log P(h[i]|z[i]) = \mathbb{E}_{P(z[i]|h[i])} \log P(h[i]|z[i]) - \text{KL}(P(z[i]|h[i])|P(z[i]))
\]

where \( KL(\cdot) \) denotes Kullback–Leibler divergence. However, directly maximizing the LB (in Eq. 6) may cause mode collapsing problem, which means the model cannot distinguish features of each cluster in the expected way [Zhu et al., 2021]. We import the mutual information \( I(z[i], h[i]) \) into LB [Zhao et al., 2018] to address this issue. Formally, we derive the following cluster objective to minimize:

\[
L^{(\text{cluster})}_{\text{KL}} = \frac{1}{d_k} \sum_i [\text{KL}(Q(z[i])||P(z[i])) - \mathbb{E}_{P_{\text{aug}}(h[i])} \mathbb{E}_{P(z[i]|h[i])} \log P(h[i]|z[i])]
\]

where \( Q(z[i]) = \mathbb{E}_{P_{\text{aug}}(h[i])} P(z[i]|h[i]) \) is the expectation of posteriors with respect to the empirical data distribution.

### 3.3 Temporal Encoder

We implement a Temporal Encoder by applying a set of standard extended 1D convolution filters to extract high-level temporal features. Specifically, our Temporal Encoder consists of three parts: (i) two Dilated Causal Convolutions (DCC) [van den Oord et al., 2016] for achieving a larger receptive field for the representation \( H^{(gm)} \), i.e., the receptive field size grows in a linear progression with the depth of the network and the kernel size of the filter; (ii) a Gated Linear Unit (GLU) for introducing the nonlinearity into the network; (iii) a \( 1 \times 1 \) convolution function for capturing sequential patterns of time series data. Formally, it works as follows:

\[
H^{(te)} = f_{\text{conv}}(\text{tanh}(H^{(gm)} \ast W_{dc1}) \odot \sigma(H^{(gm)} \ast W_{dc2}))
\]
Optimization. In the learning phase, the Mean Squared Error (MSE) is used as the regression loss:

\[
\mathcal{L}^{\text{reg}} = \sum_{t=1}^{O} \sum_{l=1}^{L} \sum_{s=1}^{S} \left\| Y_{t,l,s} - \hat{Y}_{t,l,s} \right\|^2
\]  

(11)

Eq. 7 and Eq. 11 are combined as the loss function for our TTS forecasting model:

\[
\mathcal{L} = \mathcal{L}^{\text{reg}} + \lambda \mathcal{L}^{\text{cluster}}
\]  

(12)

where \( \lambda \geq 0 \) is the balancing parameter.

4 Experiments

4.1 Experiment Setup

Datasets. We conduct experiments on two real-world TTS datasets, namely NYC Traffic Demand and BJ Air Quality as listed in Table 2, the details of which are as follows:

- **NYC Traffic Demand dataset**\(^1\) is collected from NYC Bike Sharing System, which consists of 98 locations and four sources: Bike Inflow, Bike Outflow, Taxi Inflow, and Taxi Outflow.

- **BJ Air Quality dataset**\(^2\) is collected from the Beijing Municipal Environmental Monitoring Center, which contains 10 locations and three pollutant sources: PM2.5, PM10, and SO2.

Settings. We implement the network with the Pytorch toolkit. For the model, the number of GMRE-TE layers and cluster components \( K \) are set to 4 and 17. The kernel size of each dilated causal convolution component is 2, and the related expansion rate is \{2, 4, 8, 16\} in each GMRE-TE layer. This enables our model to handle the 16 input steps. The dimension of hidden channels \( d_z \) is 24. The parameters for memory bank \( m \) and \( d_m \) are set to 8 and 48. The batch size is 8, and the learning rate of the Adam optimizer is 0.0001. In addition, the inputs are normalized by Z-Score.

Baselines. To quantitatively evaluate the prediction accuracy of our model, we implement eight baselines for comparison.


- **MTGNN** [Wu et al., 2020b]. A spatial-temporal graph convolutional network that combines graph convolution and dilated convolution.

- **Graph Wavenet (GWN)** [Wu et al., 2019]. A spatial-temporal network that combines adaptive graph convolutions with dilated casual convolution.

- **AGCRN** [Bai et al., 2020]. A recurrent neural network with adaptive graph convolution.

- **StemGNN** [Cao et al., 2020]. A spectral temporal graph neural network that maps temporal-spatial domain to spectral domain.

- **STtrans** [Wu et al., 2020a]. A deep neural network based on attention mechanism that integrates dynamic spatial, temporal and semantic dependencies.

- **ST-Norm** [Deng et al., 2021]. A normalization-based approach that refines the temporal components and the spatial components.

- **MiST** [Huang et al., 2019]. A co-predictive model for multi-categorical abnormal events forecasting, that models the spatial and categorical embeddings incorporated.

4.2 Overall Performance

We evaluate the performance of our proposed model as well as the above baselines for tensor time series forecasting on all two datasets. We repeat the experiment five times for each model on each dataset and report the average results. Through Table 3, we can find our model outperformed the state-of-the-arts to a large degree. For all four sources of the NYC Traffic Demand dataset, there have improvements of \( \Delta \text{MAE}\{14.88\%, 15.10\%, 11.14\%, 12.61\%\} \), and \( \Delta \text{RMSE}\{12.45\%, 12.78\%, 7.94\%, 11.02\%\} \) on three horizons average. Among the co-prediction models, MiST performed better than STtrans. However, MiST is second only to MTGNN in terms of bike demand and much inferior in taxi demand. Because it requires reference sources as auxiliary input while outputting the target source prediction, it is not adequate to quantify heterogeneity on three variables (i.e., the time, the location, and the source variable). By contrast, StemGNN and ST-Norm gave a worse performance because ignoring the role of source variables too much makes them not applicable for tensor time series forecasting.

The comparison results with baselines on the BJ Air Quality dataset are shown in Table 4. GMRL has the improvements of \( \Delta \text{MAE}\{1.87\%, 0.90\%, 1.45\%\} \), and \( \Delta \text{RMSE}\{1.05\%, 1.66\%, 7.94\%\} \) for three sources on three horizons average. The BJ Air Quality dataset only has 10 locations and less corrections, and many weather-related features are unavailable. This causes GMRL to perform less well on data with insufficient information, but it fully exploits the correlations among available features. In this case, baseline models are still not as efficient as GMRL.

4.3 Ablation Study

To make a thorough evaluation of key components in our model, we create several variants as follows:

\[\text{https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page}\]

\[\text{https://ride.citibikenyc.com/system-data}\]

\[\text{https://archive.ics.uci.edu/ml/datasets/Beijing+Multi+Site+Air-Quality+Data}\]
Bike Inflow

<table>
<thead>
<tr>
<th>Source</th>
<th>Bike Inflow</th>
<th>Bike Outflow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>MAE</td>
<td>RMSE</td>
</tr>
<tr>
<td>AGCRN</td>
<td>9.85</td>
<td>15.39</td>
</tr>
<tr>
<td>MTGNN</td>
<td>9.82</td>
<td>14.95</td>
</tr>
<tr>
<td>GWN</td>
<td>9.78</td>
<td>15.39</td>
</tr>
<tr>
<td>STrans</td>
<td>10.55</td>
<td>15.46</td>
</tr>
<tr>
<td>MiST</td>
<td>13.98</td>
<td>17.77</td>
</tr>
<tr>
<td>GMRL</td>
<td>9.53</td>
<td>14.80</td>
</tr>
</tbody>
</table>

Table 3: Performance on NYC Traffic Demand Dataset for the first/second/third horizon

<table>
<thead>
<tr>
<th>Source</th>
<th>Taxi Inflow</th>
<th>Taxi Outflow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>MAE</td>
<td>RMSE</td>
</tr>
<tr>
<td>AGCRN</td>
<td>9.85</td>
<td>15.39</td>
</tr>
<tr>
<td>MTGNN</td>
<td>9.82</td>
<td>14.95</td>
</tr>
<tr>
<td>GWN</td>
<td>9.78</td>
<td>15.39</td>
</tr>
<tr>
<td>STrans</td>
<td>10.55</td>
<td>15.46</td>
</tr>
<tr>
<td>MiST</td>
<td>13.98</td>
<td>17.77</td>
</tr>
<tr>
<td>GMRL</td>
<td>9.53</td>
<td>14.80</td>
</tr>
</tbody>
</table>

Table 4: Performance on BJ Air Quality Dataset for the first/second/third horizon

4.4 Case Study

Hidden Representations before/after Cluster Norm

In this subsection, we conduct multiple studies to qualitatively analyze the representations learned from GMRE of the last GMRE-TE layer. In Figure 3, we visualize variations of clusters in different channels before and after Cluster Norm. The clusters corresponding to spatial identifiers maintain a stable behavior over time, which remains unchanged even after Cluster Norm. The performance of clusters in Channel 8 before and after Cluster Norm is similar to that in Channel 1, as shown in Figure 3c and Figure 3d. However, in Channel 8, clusters 2, 9, and 16 represent temporal identifiers, and they exhibit changes with varying frequencies over time compared to Channel 1. Clearly, after applying Cluster Norm, each cluster in different channels exhibits heterogeneous evolution with sufficient discriminability. This study confirms the last GMRE-TE layer.
Component clustering capability and time-adaptability of GMRL.

**Gaussian Mixture Parameters**

In Figure 4, we display multiple Gaussian mixture parameters learned in GMRE. As the GMRE-TE layer goes deeper, Gaussian mixture parameters $\alpha$, $\mu$, and $\sigma^2$ can perfectly describe six heterogeneous components after passing through the last GMRE-TE layer. For example, cluster components 4 (purple) and 5 (brown) tend to be similar on parameters $\alpha$ and $\mu$, but their variances are very different. When the models converge, the Gaussian distribution they determine is sufficient to represent heterogeneous components.

**4.5 Hyper-parameter Study**

We further study the effect of key hyper-parameters manually set in the proposed GMRL, including the number of cluster components $K$, the dimension of the hidden channel $d_z$, and the balance parameter $\lambda$, as shown in Figure 5. We find that as $K$ and $\lambda$ increase, the prediction error of GMRL first decreases and then increases. This phenomenon follows our understanding that the model can suffer from underfitting or overfitting due to improper settings of cluster components and balance parameter. Surprisingly, the dimension of the hidden channel has little effect on GMRL: setting it to 24 can achieve fairly good performance, and increasing the dimension of the hidden channel hardly changes the performance.

**5 Related Work**

Time series forecasting has been studied for decades and applied to various fields. Some studies utilized the attention mechanism for dealing with lost memories [Li et al., 2019; Vaswani et al., 2017; Zhou et al., 2021]. To extract complex spatial-temporal patterns, existing works applied various operations over temporal and spatial domains. Specifically, the graph neural network is developed to model the spatial relation [Yu et al., 2017; Li et al., 2017; Oreshkin et al., 2021; Shang et al., 2021; Chen et al., 2020]; the attention mechanism and its variants contain spatial attention [Fang et al., 2019; Zheng et al., 2020] and temporal attention [Wu et al., 2021; Zheng et al., 2020; Liu et al., 2021; Zhou et al., 2022]; the convolution operator is leveraged to spatial convolution [Deng et al., 2022b; Guo et al., 2021], temporal convolution [Wu et al., 2020b; Wu et al., 2019], spatial-temporal convolution [Guo et al., 2019; Yang et al., 2021] and adaptive convolution [Pan et al., 2019]. [Zhu et al., 2021] proposes an end2end mixture model to cluster microscopic series for macroscopic time series forecasting. [Jiang et al., 2021] reviews the deep learning models and builds a standard benchmark. [Deng et al., 2021; Deng et al., 2022a] refine the high-frequency and local components from MTS data by using normalization technology. Unlike MTS involved data within a matrix, data points within TTS usually take on more substantial heterogeneity and dynamics. Since such unique properties of TTS compared to MTS, traditional methods for predicting MTS (or MTS with reference source) are not entirely suitable for TTS prediction. The inability to explicitly learn and encode interactions from TTS brings up some intuitive concerns. A new line of research further extends the spatial-temporal dependency modeling by considering another domain (i.e., source or categorical domain) [Ye et al., 2019; Huang et al., 2019; Huang et al., 2021; Han et al., 2021; Wang et al., 2021] or extra data source [Jiang et al., 2023]. Our work distinguishes itself from these methods in the following aspects: (i) our GMRE is dynamic, which could explicitly represent the complicated heterogeneous components among multiple variables in TTS; (ii) our HRA enhances the learned representations with sequence-specific global patterns to distinguish and generalize to diverse scenarios, which could help model adaptability to TTS.

**6 Conclusion**

In this paper, we develop a dynamic Gaussian mixture representations learning framework GMRL, which seeks to individually model each heterogeneity component implied in multiple variables. In addition, we design an HRA module that enhances the learned representation with sequence-specific global patterns to distinguish and generalize to diverse scenarios. Extensive experiments on two real-world datasets verify our perspective’s insightfulness and the proposed approach’s effectiveness. In the next step, we plan to improve the speed of adaptation and efficiency of GMRL.

**Contribution Statement**

Renhe Jiang is the corresponding author for this study.

**References**


[Cao et al., 2020] Defu Cao, Yujing Wang, Juanyong Duan, Ce Zhang, Xia Zhu, Congrui Huang, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, et al. Spectral temporal


