Time-delayed Multivariate Time Series Predictions

Hao Niu* Guillaume Habault* Roberto Legaspi* Chuizheng Meng† Defu Cao†
Shinya Wada* Chihiro Ono* Yan Liu†

Abstract

A major issue with real-time monitoring is to collect complete data. Hardware or software failures, network issues or, more frequently, time delays can disrupt such a collection. This results in having two versions of the same information: one in real-time but with potentially missing data, and the another, albeit complete, is delayed. Many works have studied how to handle missing data for classification and prediction. However, to the best of our knowledge, they do not consider how to leverage the delayed complete data to assist in learning the representation of real-time available data with missing values. This is despite the fact that the delayed complete data contain all the information (e.g., periodicities and trends). In this paper, we propose a framework to enhance the representation learning of the real-time available data by aligning the representation of past real-time but with missing data to that of past delayed but complete data. We test both a distance metric and contrastive learning to achieve this alignment. We implement our framework on a Transformer-based model and experiment it on three datasets. The efficiency of our solution is evaluated against seven baselines and considering four distinct patterns of missing data. Our experiments show that this proposal has a significant improvement in prediction accuracy (5.21% on average) over the baselines.

Keywords: multivariate time series prediction, time delay, representation alignment, contrastive learning

1 Introduction

Real-time data is information that is available as soon as — without any delay — it is created and acquired. The timeliness of this information is crucial for supporting live, this-instant decision making to ensure reliable, high-quality services. This data is ubiquitous, powering everything from health monitoring, bank transactions, online education delivery, energy generation and consumption, to COVID-19 information maps that have emerged during this years-long pandemic.

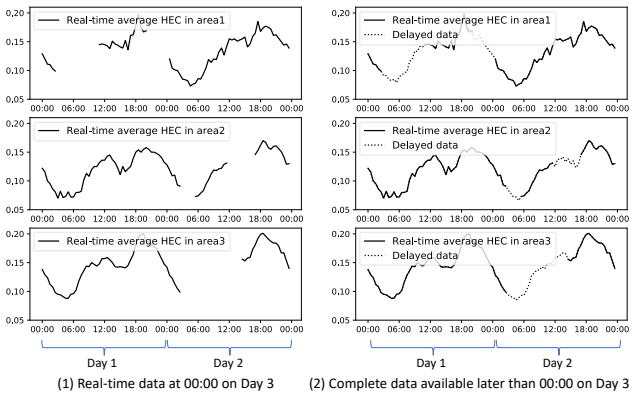

Figure 1: Example of time-delayed Multivariate Time Series data from an electricity retailer. The figures show the average household electricity consumption (HEC) in three postcode areas for two days (Day 1-2). Left figures depict data collected in real-time at 00:00 on Day 3, at which real-time predictions should be performed for electricity planning. Right figures represent the same plots, with the data that were later retrieved (dotted line). As the characteristic of the delay varies depending on the emerging issue, delayed data are usually ignored when performing real-time predictions.

However, missing values in real-time data are prevalent due to network communication failure, device replacement, equipment malfunctions, among many other reasons. Missing data refers to the absence of values for the variables of interest. The presence of missing values in real-time data can diminish its usability and interpretability, leading to skewed results in subsequent statistical analyses. In most cases, the missing data are not permanently lost. They can still be recovered, but not void of delay. A delay is contingent to the associated issue, such as a network congestion or transmission delay that slows down data retrieval [33]. Another example is when energy smart-meters are functioning, but communications are out of service, requiring a technician to physically retrieve the monitored data at a later time, e.g., for billing. Data will eventually arrive, yes, but be delayed.

*KDDI Research, Inc., {ha-niu, xgu-habault}@kddi.com, rolegaspi@kddi-research.jp, {sh-wada, ci-ono}@kddi.com. Shinya Wada and Chihiro Ono are also with KDDI Corporation.

†University of Southern California, {chuizhem, defucao, yanliu.cs}@usc.edu.

In light of the above, there can be two versions of the same information: the one in real-time but with potentially missing data (we refer hereafter as real-time available data (RAD)), and the one that is delayed but without missing data (hereafter, delayed complete data (DCD)). Figure 1 illustrates both, depicting two days of average Household Electricity Consumption (HEC) in three different postal code areas. The left part shows real-time data at a given time step with missing values (depicted by the blank sections). The right part exhibits the same information, but now including data that were later on retrieved (plotted with dotted lines), hence, delayed but completing parts that were previously missing.

Further, the real-time data being Multivariate Time Series (MTS) adds more complexity to the problem. As shown in Figure 1, missing data in the various instances do not necessarily start at the same moment nor last within the same time period. It becomes even more difficult when both the dependent and independent variables contain missing information. Such issues make it harder to correctly learn the possible relations or dependencies between the different variables of interest. Add to these the high variability in the length of the delay (e.g., a few seconds to several days), and data later retrieved are just conveniently retained, e.g., to be treated as ground truth afterward. Several methods exist to handle missing data in MTS. Some attempt to fill-in the missing values: the simplest is to perform zero imputation, which treats the value of a missing entry as zero; or use interpolation (e.g., linear or polynomial) based on the available data. More recent works use Neural Networks (NN) to make sense of data patterns in order to perform a more complex imputation. Another way is to directly treat MTS with missing data as an irregularly sampled MTS, i.e., the time between two consecutive measurements varies. Nevertheless, handling missing data in MTS remains a daunting problem [2].

In this work, we leverage delayed complete MTS data to improve predictions of MTS when only using real-time available data, which may contain missing portions. To accomplish this, our proposed framework learns a representation of each version of the information (RAD and DCD). The representation of RAD is partial (based on what is incompletely known), while the one for DCD contains all known aspects of the data. Finally, and equally non-trivial, we enhance the representation of RAD by aligning it with the representation of DCD. This alignment is first realized by reducing the distance between them. However, a representation distance loss may not capture the plausibly various disparities across samples (depending on the nature of the dataset). Hence, we introduce contrastive learning, in which portions of the RAD and DCD that completely overlap are considered as positive pairs, and portions that differ in starting times to constitute the negative pairs. With a contrastive loss, therefore, our framework can learn meaningful representations by optimizing the similarity of positive pairs in contrast to the similarity of negative pairs, thereby accounting for discrepancies in the representations. To the best of our knowledge, using DCD to enhance the representation of RAD has not been previously investigated.

Our work's main contributions are as follows:

- A new application-oriented (i.e., time-delayed) MTS prediction task where only RAD (with potentially missing parts) are available for real-time prediction, but historical DCD are also accessible.
- A novel MTS prediction framework, namely Enhanced Representation Learning (ERL), which enhances the representation learning of RAD with the aid of DCD through representation alignment. In ERL, both versions of the data are necessary in the training phase, but only RAD is required during testing. In addition, ERL adopts both distance metric and contrastive learning for this alignment.
- Experiments on both public and proprietary MTS datasets to evaluate our solution using ERL against baseline methods that could handle missing data for time-delayed MTS prediction. ERL improves the performance of its base model and beats other baseline methods. On average, it improves the prediction accuracy by 5.21%. In a liberal electricity market scenario, such improvements help electricity retailers mitigate under- or over-procuring electricity, ultimately leading to significant cost reduction when balancing procurement and actual demand.

2 Related Works

Missing data is a real challenge, as it can significantly hinder the performance of various services, such as predictions. Handling missing data is still an important research topic. Especially, the latest advancement in AI provides additional ways to treat them. As mentioned previously, NN can be used to fill in missing values, e.g., IPnet [22] and SSGAN [18]. Other works, like [4, 5], further study the importance of missing patterns. It is also possible to consider real-time available data with potentially missing parts as irregularly sampled data. In such a consideration, techniques such as Neural Ordinary Differential Equations [6, 21] or Attention mechanism [23] can be used to account for any missing information. Finally, in the past years, Transformer [26]-based models have proved to be efficient in many tasks such as image classification [25], Natural

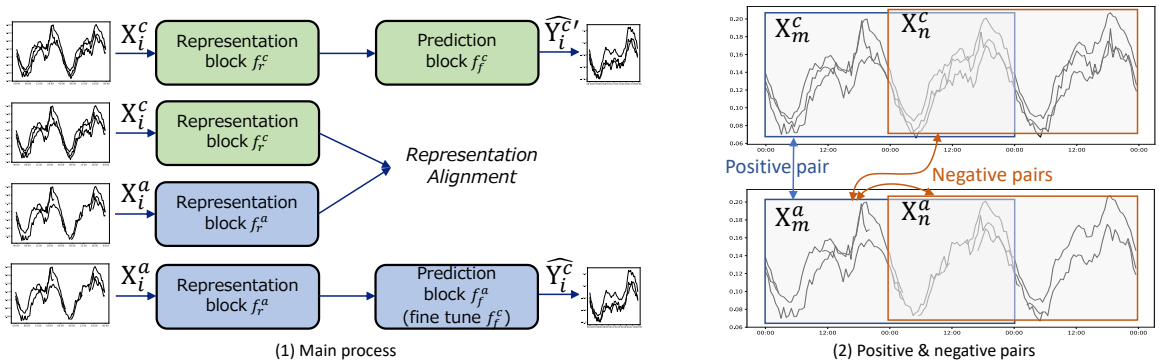

Figure 2: ERL's main process, and examples of the positive and negative pairs for contrastive learning.

Language Processing (NLP) tasks [10] and time series predictions [17, 34]. It is therefore not surprising to see it applied to the missing data issue [7]. In addition, [20] demonstrates that a modified version of Informer [34] could outperform existing methods for the data imputation task.

In recent years, Contrastive Learning (CL) has demonstrated its superiority in computer vision applications [12, 14]. Such a technique helps models learn meaningful representations of samples by maximizing the similarity of positive pairs against the similarity of negative pairs. Lately, this solution has shown rising interest and has been applied to several other domains such as NLP [3, 8], or recommendation [31, 35]. In the case of time series applications, [24] treats neighborhood samples as positive pairs, while in [32], the representations at the same timestamp in different samples are regarded as the positive pairs. Besides, CL can also be considered within the frequency domain as studied in [29]. In addition to contrastive loss, some works also highlight the importance of representation alignment with distance metrics [9, 27, 28].

3 Task Formulation

We consider the time-delayed MTS data: $\mathbf{V}_{N,T} \in \mathbb{R}^{N \times T}$, where N is the number of variables and T is the number of time steps. As we deal with time-delayed data, there are two versions of the same information: $\mathbf{V}^a_{N,T}$, the real-time available data, in which at a given moment in time, some values may be unavailable for some variables at potentially different time steps; and $\mathbf{V}^c_{N,T}$, the delayed complete data where all values are known but potentially acquired later. Note that in this work we dismiss scenarios where some values are really lost (e.g., device failure without a redundant design).

Our task is to perform real-time MTS prediction using only the RAD with potential missing portions as inputs. Considering that t_k is the current time, and that there are p future steps to predict using o past steps of observation, we aim to predict $\mathbf{Y}^c_k = \mathbf{V}^c_{N,T^k_{+p}}$ using $\mathbf{X}^a_k = \mathbf{V}^a_{N,T^k_{-o}}$, where $T^k_{+p} = [t_{k+1}, \cdots, t_{k+p}]$ and $T^k_{-o} = [t_{k-o+1}, \cdots, t_k]$. In other words, we train a prediction model f such that $\mathbf{Y}^c_k = f(\mathbf{X}^a_k) + \epsilon$, ϵ being the prediction error of the model. In historical data beyond the previous d time steps, both $\mathbf{Y}^c_i = \mathbf{V}^c_{N,T^i_{+p}}$ and $\mathbf{X}^c_i = \mathbf{V}^c_{N,T^i_{-o}}$ $(i \leq k-d)$ are available. Apart from treating $\mathbf{Y}^c_i$ as ground truth, $\mathbf{X}^c_i$ can be additionally used to assist the training of the model f, which motivates our framework.

4 Proposed Framework: ERL

In this paper, we posit to use $\mathbf{X}^c_i$ to enhance the representation learning of $\mathbf{X}^a_i$ and so, aid in training the model f.

4.1 Main Process We consider that a prediction model f is composed of two parts: the representation learning block f_r and the prediction block f_f. As shown in Figure 2, we first train a model f^c (thereby, composed of f^c_r and f^c_f) using $\mathbf{X}^c_i$ as the input and $\mathbf{Y}^c_i$ as the ground truth. Considering that $\mathbf{R}^c(i)$ is the representation of $\mathbf{X}^c_i$ by f^c_r, the prediction obtained $\widehat{\mathbf{Y}^c_i}'$ can be expressed as:

$$\widehat{\mathbf{Y}^c_i}' = f^c_f(\mathbf{R}^c(i)), \mathbf{R}^c(i) = f^c_r(\mathbf{X}^c_i). \tag{4.1}$$

The loss used to train this model can be written as:

$$\mathcal{L}^c = \mathcal{L}\left(\widehat{\mathbf{Y}^c_i}', \mathbf{Y}^c_i\right), \tag{4.2}$$

where $\mathcal{L}$ is a generic loss function for time series prediction (e.g., Mean Squared Error (MSE)). The second step of our proposal is to train a representation block f^a_r using $\mathbf{X}^a_i$ as input and align its representation to the one of $\mathbf{X}^c_i$. $\mathbf{R}^a(i) = f^a_r(\mathbf{X}^a_i)$ is the representation of $\mathbf{X}^a_i$ with this block.

The loss function of this alignment is defined as:

$$\mathcal{L}_{align}\left(\mathbf{R}^a, \mathbf{R}^c\right). \tag{4.3}$$

In this way, we leverage the representations learned from the DCD to assist the representation learning for RAD.

At last, we combine the aligned representation of the RAD with the prediction block of the DCD model f_f^c. This combination model is then fine tuned in order to obtain our real-time prediction block f_f^a. To do this fine tuning, we train the combined model with the samples $\mathbf{X}_i^a$, using the following loss:

$$\mathcal{L}^a = \mathcal{L}\left(\widehat{\mathbf{Y}_i^c}, \mathbf{Y}_i^c\right). \tag{4.4}$$

For the evaluation phase (and also real-time application), we obtain the future p prediction steps, using both f_r^a and f_f^a blocks:

$$\widehat{\mathbf{Y}_j^c} = f_f^a\left(f_r^a\left(\mathbf{X}_j^a\right)\right), \tag{4.5}$$

where $j \geq k$. This process enables us to use DCD $\mathbf{X}^c$ only during the training phase. After completion of the training, we can perform predictions using only RAD $\mathbf{X}^a$. The process is summarized in Algorithm 1.

Algorithm 1 Main process of ERL for time-delayed MTS prediction

Input: Time-delayed MTS data $(\mathbf{X}^a, \mathbf{X}^c, \mathbf{Y}^c)$
Output: Trained prediction model
$\widehat{\mathbf{Y}^c} = f_f^a\left(f_r^a\left(\mathbf{X}^a\right)\right)$
***Train:**
1. Initialize parameters
2. Train $\widehat{\mathbf{Y}_i^c}' = f_f^c\left(f_r^c\left(\mathbf{X}_i^c\right)\right)$ with loss of Eqn. (4.2)
3. Train f_r^a with loss of Eqn. (4.3)
4. Train f_f^a with loss of Eqn. (4.4)

***Test/Application**: Perform prediction with RAD test set $\mathbf{X}_j^a$ according to $\widehat{\mathbf{Y}_j^c} = f_f^a\left(f_r^a\left(\mathbf{X}_j^a\right)\right)$

4.2 Distance-based alignment As for the representation alignment, the first and intuitive method is to reduce the distance between them like [9, 27, 28]. For this alignment loss, we used the Euclidean Distance (ED) as defined by

$$\mathcal{L}_{align}\left(\mathbf{R}^a, \mathbf{R}^c\right)_{ED} = \mathrm{ED}\left(\mathbf{R}^a, \mathbf{R}^c\right). \tag{4.6}$$

However, such a distance loss considers only the closeness of the representations of the same sample from both RAD and DCD, but does not reflect the degree of disparity among different samples (e.g., in a dataset the weekdays and weekends may have totally different profiles). This issue might affect prediction performance depending on the datasets. Thus, we also explored CL to account for differences between samples.

4.3 Contrastive learning-based alignment Contrary to ED, CL accounts for sample differences. The major step of CL is to generate positive and negative pairs. In our framework, we want the m^{th} RAD sample (i.e., $\mathbf{X}_m^a$) to be as close as possible to the corresponding DCD sample (i.e., $\mathbf{X}_m^c$). As a result, we treat them as positive pairs. In addition, we consider $\mathbf{X}_m^a$ and $\mathbf{X}_n^a$ or $\mathbf{X}_n^c$ $(n \neq m)$ as negative pairs. Therefore, we can express our contrastive loss as follows:

$$\mathcal{L}_{align}\left(\mathbf{R}^a, \mathbf{R}^c\right)_{CL} = -\log \frac{\exp\left(\mathbf{R}^a(m) \cdot \mathbf{R}^c(m)\right)}{\sum_n \left(\exp\left(\mathbf{R}^a(m) \cdot \mathbf{R}^a(n)\right) + \exp\left(\mathbf{R}^a(m) \cdot \mathbf{R}^c(n)\right)\right)}. \tag{4.7}$$

4.4 Combine versus Fuse alignment Similarly to [28] both distance and contrastive losses could be summed when training the representation block f_r^a. In such a version, the alignment loss definition becomes:

$$\mathcal{L}_{align}\left(\mathbf{R}^a, \mathbf{R}^c\right)_{ED} + \alpha \mathcal{L}_{align}\left(\mathbf{R}^a, \mathbf{R}^c\right)_{CL}. \tag{4.8}$$

However, to avoid the arduous tuning of the hyperparameter α, we propose to simply fuse the predicted results of different prediction models. For instance, denoting the outputs of the ED version $\widehat{\mathbf{Y}_{j\,ED}^c}$ and those of the CL version $\widehat{\mathbf{Y}_{j\,CL}^c}$, the fusion of these outputs is:

$$\widehat{\mathbf{Y}_{j\,ED-CL}^c} = \left(\widehat{\mathbf{Y}_{j\,ED}^c} + \widehat{\mathbf{Y}_{j\,CL}^c}\right)/2. \tag{4.9}$$

Furthermore, aligning the representation of RAD to that of DCD may result in information loss of missing patterns involved in RAD, and thus we propose to further fuse the results of the original base model trained using $\mathbf{X}^a$ and $\mathbf{Y}^c$ as the input and output directly. As a result, the final prediction result is:

$$\widehat{\mathbf{Y}_j^c} = \left(\widehat{\mathbf{Y}_{j\,ED}^c} + \widehat{\mathbf{Y}_{j\,CL}^c} + \widehat{\mathbf{Y}_{j\,Base}^c}\right)/3. \tag{4.10}$$

We denote this framework as ERL in this paper.

4.5 Implementation on a Transformer-based model In this work, we decided to implement ERL on Informer, because it has shown excellent performance with imputed data, as aforementioned in the *Related Works* section[1]. In addition, for any Transformer-based models that define an Encoder/Decoder architecture, the Decoder can be seen as our prediction block (i.e., f_f). This block has two inputs as shown in Figure 3.

[1] We are aware that other Transformer-based models, such as Autoformer [30] and Pyraformer [16], recently demonstrated better performance. But, these models share a similar architecture. Our framework can also be implemented onto them. Conclusions similar to the ones presented in this paper can be expected.

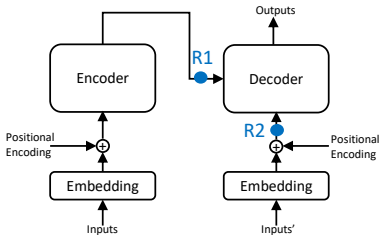

Figure 3: Illustration of the Encoder/Decoder architecture of some Transformer-based models. R1 and R2 represent the points of representation alignment.

The first one is the output of the Encoder. The second one is the concatenation of both the embedding and the positional encoding of the padded input. For ERL, we implemented the representation alignment for both as illustrated in Figure 3.

5 Experiments and Results

5.1 Datasets and baselines We test our proposed framework on three different MTS datasets: two open datasets, namely, Solar Energy (SE) [13, 19] and PEMS-Bay (PB) [15], and the commercial HEC dataset. Table 1 shows statistics and task settings of each dataset in our experiments. We have selected these datasets since each variable represents the same physical quantity, but measured at a different location: Photovoltaics (PV) generation from 137 solar power plants for SE, traffic speed from 325 sensors for PB, and electricity consumption from 20 postal code areas for HEC. In the rest, a variable refers to data from one location.

	SE	PB	HEC
period	2006.01-2006.12	2017.01-2017.06	2019.05-2020.07
time step size	30min	5min	30min
variables	137	325	20
input length	2 days/96 steps	1 day/288 steps	2 days/96 steps
output length	1 day/48 steps	half day/144 steps	1 day/48 steps

Table 1: Datasets and task descriptions.

As illustrated in Figure 4, all these datasets depict clear daily periodicity, but at the same time they have their own specificity. For instance, SE does not generate electricity during night and its daily pattern is stable. Indeed, apart from the time, the other variable that greatly alter PV generation is the weather, which yields big difference in magnitude between different days. But, even though the weather at different locations might be different, nearby locations often have very similar conditions. As a result, their PV generation will also be similar with more or less latency. PB has its own complexity. Traffic speed is quite stable. Nevertheless, it is highly subject to variations due to events (e.g., sports events), weather conditions and other effects [1]. It often results in unexpected traffic congestion. Such sudden changes are not necessarily area specific, which lead to potentially have some variables with different patterns. On the other hand, daily patterns in HEC are relatively regular and, except for weekends or holidays, the magnitude does not fluctuate much. In addition, at the postal code level, areas with similar attributes (e.g., residential versus office areas) have similar consumption patterns [11]. As a consequence, for each of the selected datasets, sets of variables share similarities (similar geographical locations for SE, similar roads for PB, and similar attributes for HEC). Such similarities can strengthen our solution. Indeed, if one variable has missing data, knowledge acquired from past correlations with other similar variables could be used to better account for the missing data.

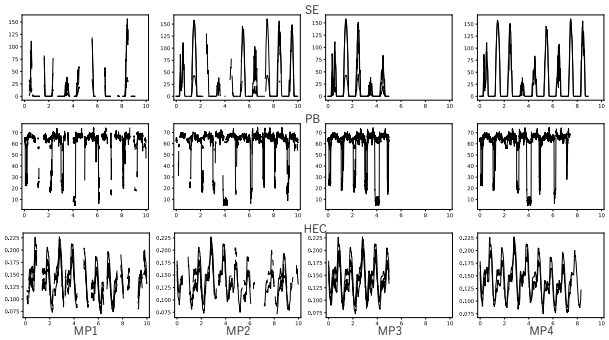

Figure 4: Representation of the missing patterns and fluctuations of the input samples (using 10 days, missing ratio = 0.5, and 3 variables for example).

Originally, the datasets we selected do not have missing data. In order to exhaustively evaluate the impact of our framework, we simulate missing data using different Missing Data Patterns (MPs) and different Missing Data Rates (MRs). As shown in Figure 4, we defined four specific MPs belonging to two categories:

- Category 1: The number of missing blocks, their positions and sizes are randomly drawn.
  -MP1: Each variable has the same missing blocks.
  -MP2: Each variable has different missing blocks.
- Category 2: There is only one missing block at the end of the input sample (most recent values).
  -MP3: The missing block size is the same and fixed for each variable.
  -MP4: The missing block size is randomly drawn for each variable, but the overall average amount of missing data corresponds to the MR.

We performed our experiments on a Linux server with a TITAN V 12GB GPU. Our task was to perform day-ahead prediction using two days of historical real-

	MP1						MP2						MP3						MP4					
MR	0.2		0.5		0.8		0.2		0.5		0.8		0.2		0.5		0.8		0.2		0.5		0.8	
Metric	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
Solar Energy																								
GRU-D	61.22	56.52	61.36	56.65	61.35	56.65	61.66	56.95	61.38	56.67	61.4	56.7	61.3	56.6	61.3	56.6	61.3	56.61	61.38	56.68	61.34	56.64	61.17	56.47
BRITS	33.82	16.95	33.99	17.13	34.12	17.08	34.0	17.15	34.17	17.2	34.58	17.61	33.91	16.95	33.91	16.95	32.39	16.97	33.27	16.79	31.97	16.25	33.69	17.1
SSGAN	32.38	16.32	32.56	16.46	32.78	16.77	37.24	18.38	36.72	18.46	36.0	18.24	32.62	16.51	32.62	16.41	32.35	16.35	32.37	16.48	32.37	16.43	32.62	16.42
L-ODE	30.94	18.62	32.5	18.63	34.07	19.97	30.98	18.44	33.3	19.82	33.78	19.55	31.24	18.87	31.23	18.41	32.23	19.5	30.08	17.81	29.44	17.07	30.18	17.98
mTAN	32.3	16.51	33.25	16.87	34.22	18.08	31.8	17.62	32.22	16.42	31.61	18.23	30.14	15.76	33.18	16.59	33.32	16.61	30.77	16.22	32.64	17.26	33.54	16.48
Informer*	29.92	16.6	30.69	16.32	33.66	17.96	28.22	15.94	29.12	16.43	29.97	16.93	28.04	15.7	29.93	15.86	30.27	16.2	29.73	16.36	29.69	15.91	28.93	15.19
Informer-	30.49	16.85	29.15	16.29	31.24	17.16	30.33	16.4	32.49	18.27	31.31	18.4	27.57	16.19	30.47	16.73	32.94	18.52	29.98	15.99	29.22	15.31	30.81	16.57
ERL	**27.99**	**15.35**	**28.53**	**15.36**	**30.69**	**15.92**	**27.87**	**15.32**	**28.27**	**15.37**	**29.08**	**15.99**	**27.49**	**15.13**	**29.58**	**15.61**	**29.95**	**15.44**	**27.81**	**15.09**	**28.04**	**15.03**	**28.2**	**15.01**
PEMS-Bay																								
GRU-D	10.22	5.53	10.24	5.53	10.35	5.55	10.18	5.49	10.23	5.51	10.27	5.52	10.17	5.5	10.17	5.5	10.17	5.5	10.18	5.52	10.15	5.49	10.18	5.52
BRITS	7.87	4.48	8.08	4.6	8.45	4.84	7.91	4.5	8.05	4.66	8.33	4.85	7.62	4.3	7.62	4.3	7.62	4.3	7.74	4.36	7.84	4.4	7.9	4.44
SSGAN	9.04	5.57	9.03	5.51	9.16	5.57	9.33	5.71	9.02	5.45	6.11	3.73	5.92	3.58	8.94	5.36	8.9	5.38	8.93	5.43	5.93	3.58	5.98	3.61
L-ODE	10.24	5.69	9.69	5.38	9.88	5.7	9.85	5.35	10.04	5.52	10.36	5.91	10.09	5.53	10.38	5.85	9.79	5.57	10.09	5.75	9.79	5.81	10.15	5.84
mTAN	9.57	5.05	9.62	5.05	10.01	5.52	9.34	4.88	9.79	5.24	9.55	5.14	9.45	4.84	9.83	5.3	9.59	5.18	9.52	4.97	9.96	5.23	9.92	5.35
Informer*	5.85	3.09	6.11	3.16	6.41	3.35	5.67	2.95	6.1	3.27	6.29	3.28	5.74	3.01	6.46	3.29	6.95	3.66	5.7	2.96	5.58	2.93	5.88	3.1
Informer-	5.83	3.08	6.02	3.2	6.48	3.38	5.7	3.01	6.01	3.16	6.18	3.21	5.75	3.04	6.43	3.32	6.98	3.72	5.66	2.94	5.64	2.9	6.55	3.38
ERL	**5.58**	**2.89**	**5.83**	**3.0**	**6.14**	**3.22**	**5.53**	**2.84**	**5.74**	**2.99**	**6.0**	**3.14**	**5.51**	**2.84**	**6.2**	**3.19**	**6.75**	**3.53**	**5.5**	**2.82**	**5.49**	**2.82**	**5.7**	**2.95**
Household Electricity Consumption																								
GRU-D	4.77	4.18	4.95	4.35	4.94	4.34	4.69	4.09	4.78	4.18	4.84	4.24	4.66	4.06	4.66	4.06	4.82	4.22	4.68	4.08	4.73	4.13	4.74	4.14
BRITS	3.38	2.71	3.51	2.82	3.64	2.95	3.47	2.8	3.44	2.78	3.57	2.88	3.41	2.73	3.41	2.73	3.65	2.96	3.36	2.69	3.35	2.68	3.49	2.8
SSGAN	2.96	2.41	2.93	2.42	2.92	2.41	3.67	2.9	4.23	3.28	4.4	3.42	2.89	2.38	2.83	2.32	2.81	2.32	2.89	2.39	2.85	2.35	2.82	2.32
L-ODE	2.17	1.63	2.44	1.84	3.17	2.39	2.53	1.94	2.7	2.14	2.59	2.0	2.36	1.86	2.71	2.16	2.53	2.02	2.12	1.6	2.3	1.72	2.28	1.75
mTAN	2.26	1.69	2.69	2.02	3.02	2.26	2.17	1.62	2.19	1.62	2.25	1.66	2.31	1.74	2.36	1.73	**2.35**	**1.75**	2.21	1.64	2.26	1.67	2.29	1.7
Informer*	2.11	1.57	2.45	1.78	2.95	2.22	2.12	1.55	2.12	1.58	2.34	1.73	2.2	1.62	2.45	1.82	2.41	1.81	1.99	1.47	2.35	1.74	2.29	1.7
Informer-	2.16	1.61	2.32	1.73	2.85	2.18	2.1	1.53	2.26	1.69	2.42	1.79	2.17	1.58	**2.23**	1.68	2.57	1.93	2.1	1.53	2.16	1.63	2.42	1.82
ERL	**1.96**	**1.44**	**2.25**	**1.65**	**2.62**	**1.97**	**2.0**	**1.45**	**2.03**	**1.5**	**2.1**	**1.53**	**2.13**	**1.54**	2.24	**1.66**	2.4	1.78	**1.98**	**1.44**	**2.1**	**1.55**	**2.18**	**1.6**

Table 2: Experiment results of our final solution (ERL) and the baselines for three datasets with different Missing Data Pattern (MP) and different Missing Data Rate (MR).

time available data for SE and HEC. Because of its granularity, we conducted half-day-ahead prediction using one day of historical data for PB. We generated samples using (i) a sliding window with one day [resp. half a day] stride for SE and HEC [resp. PB], (ii) one MP, and (iii) one MR. In addition, we constructed the train, validation, and test datasets with 60%, 20%, and 20% of the data, respectively (considering the samples in their temporal order).

We compared our proposal to seven baselines: five models designed for handling missing data (GRU-D [5], BRITS [4], SSGAN [18], Latent ODE (L-ODE) [21] and mTAN [23]), and two versions of the Informer [34]. For the methods that were originally designed for classification, similarly to Informer, we use a fully connected layer after the output to realize long-term prediction with one forward step. Contrary to the other baselines, Informer was not originally designed for handling missing data and does not have a specific mechanism for such a task. Therefore, Informer* [resp. Informer-] is an Informer where zero-imputation [resp. linear interpolation] has been applied beforehand. ERL is based on Informer*, as shown in our experiment, it beats all the baselines as well as in most cases Informer-.

5.2 Prediction performance Table 2 lists the prediction performance over different MPs and different MRs for the considered datasets. The selected evaluation metrics are Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). Each scenario (i.e., a combination of one MP and one MR) is conducted three times. The table shows the average of the obtained results (we magnified the one for HEC by 100 for readability). The best [resp. second-best] result in each scenario is emphasized in bold [resp. underscore]. We can observe from these results that: (1) imputed-Informer (* and -) is better than other baselines in MTS prediction when some missing data need to be handled; (2) Informer* outperforms Informer- in 41 over the 72 cases; (3) ERL achieves the best performance in almost all of the cases with only three exceptions for HEC (but still gets the second place).

Informer with simple imputation techniques shows excellent performance compared to the baselines. However, using DCD only as the ground truth is not enough to learn the full patterns involved in the data (especially for high MRs). Conversely, our proposal can benefit from this knowledge by aligning the representation of RAD to that of DCD.

5.3 Ablation study To obtain ERL's results, we fused the outputs of: (1) Informer with zero-imputation (*), (2) Algorithm 1 with Euclidean Distance alignment loss (referred to as P-ED), and (3) Algorithm 1 with Contrastive Learning alignment loss (referred to as P-CL). P-ED and P-CL use Informer* as their base model. Table 3 presents separately the performance of each version. In addition, we also provide results using the

	MP1						MP2						MP3						MP4						
MR	0.2		0.5		0.8		0.2		0.5		0.8		0.2		0.5		0.8		0.2		0.5		0.8		Mean
Metric	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	
Solar Energy																									
Informer*	29.92	16.60	30.69	16.32	33.66	17.96	28.22	15.94	29.12	16.43	29.97	16.93	28.04	15.70	29.93	15.86	30.27	16.20	29.73	16.36	29.69	15.91	28.93	15.19	
P-ED	28.52	15.62	28.69	15.31	30.25	**15.66**	28.99	16.00	28.93	15.81	30.32	16.30	28.33	15.46	29.29	15.94	30.37	16.14	28.71	15.65	28.58	15.56	28.80	15.44	
Imp(%)	*4.7*	*5.9*	*6.5*	*6.2*	*10.1*	*12.8*	*-2.7*	*-0.4*	*0.7*	*3.8*	*-1.2*	*3.7*	*-1.0*	*1.5*	*2.1*	*-0.5*	*-0.3*	*0.4*	*3.4*	*4.3*	*3.7*	*2.2*	*0.4*	*-1.6*	*2.70*
P-CL	28.82	16.14	29.34	16.47	30.95	16.59	28.22	16.03	29.19	15.99	30.05	17.19	27.81	15.85	31.60	17.58	31.04	16.15	27.93	15.62	28.13	15.42	28.30	15.72	
Imp(%)	*3.7*	*2.8*	*4.4*	*-0.9*	*8.1*	*7.6*	*0.0*	*-0.6*	*-0.2*	*2.7*	*-0.3*	*-1.5*	*0.8*	*-1.0*	*-5.6*	*-10.8*	*-2.5*	*0.3*	*6.1*	*4.5*	*5.3*	*3.1*	*2.2*	*-3.5*	*1.02*
P-ED+CL	28.20	16.03	28.48	**15.29**	**30.23**	15.85	28.76	15.54	29.21	16.04	30.02	**15.94**	27.81	15.42	**29.22**	15.66	30.16	15.90	28.6	15.31	28.83	15.37	28.63	15.59	
Imp(%)	*5.7*	*3.4*	*7.2*	*6.3*	*10.2*	*11.7*	*-1.9*	*2.5*	*-0.3*	*2.4*	*-0.2*	*5.8*	*0.8*	*1.8*	*2.4*	*1.3*	*0.4*	*1.9*	*3.8*	*6.4*	*2.9*	*3.4*	*1.0*	*-2.6*	*3.18*
P-ED-CL	28.27	15.65	**28.45**	15.59	**30.23**	15.84	28.34	15.77	28.73	15.64	29.77	16.47	27.76	15.41	29.86	16.27	30.27	15.74	28.00	15.34	28.17	15.33	28.27	15.35	
Imp(%)	*5.5*	*5.7*	*7.3*	*4.5*	*10.2*	*11.8*	*-0.4*	*1.1*	*1.3*	*4.8*	*0.7*	*2.7*	*1.0*	*1.8*	*0.2*	*-2.6*	*0.0*	*2.8*	*5.8*	*6.2*	*5.1*	*3.6*	*2.3*	*-1.1*	*3.36*
ERL	**27.99**	**15.35**	28.53	15.36	30.69	15.92	**27.87**	**15.32**	**28.27**	**15.37**	**29.08**	15.99	**27.49**	**15.13**	29.58	**15.61**	**29.95**	**15.44**	**27.81**	**15.09**	**28.04**	**15.03**	**28.20**	**15.01**	
Imp(%)	*6.5*	*7.5*	*7.0*	*5.9*	*8.8*	*11.4*	*1.2*	*3.9*	*2.9*	*6.5*	*3.0*	*5.6*	*2.0*	*3.6*	*1.2*	*1.6*	*1.1*	*4.7*	*6.5*	*7.8*	*5.6*	*5.5*	*2.5*	*1.2*	*4.72*
PEMS-Bay																									
Informer*	5.85	3.09	6.11	3.16	6.41	3.35	5.67	2.95	6.10	3.27	6.29	3.28	5.74	3.01	6.46	3.29	6.95	3.66	5.70	2.96	5.58	2.93	5.88	3.10	
P-ED	5.71	2.96	6.01	3.09	6.36	3.33	5.67	2.94	5.85	3.05	6.15	3.22	5.61	2.94	6.54	3.32	7.44	3.75	5.62	2.93	5.62	2.88	5.84	3.07	
Imp(%)	*2.4*	*4.2*	*1.6*	*2.2*	*0.8*	*0.6*	*0.0*	*0.3*	*4.1*	*6.7*	*2.2*	*1.8*	*2.3*	*2.3*	*-1.2*	*-0.9*	*-7.1*	*-2.5*	*1.4*	*1.0*	*-0.7*	*1.7*	*0.7*	*1.0*	*1.04*
P-CL	5.73	3.01	6.14	3.21	6.77	3.56	5.70	2.98	5.95	3.13	6.30	3.34	5.62	2.92	6.86	3.56	6.93	3.70	5.67	2.95	5.70	2.94	5.91	3.05	
Imp(%)	*2.1*	*2.6*	*-0.5*	*-1.6*	*-5.6*	*-6.3*	*-0.5*	*-1.0*	*2.5*	*4.3*	*-0.2*	*-1.8*	*2.1*	*3.0*	*-6.2*	*-8.2*	*0.3*	*-1.1*	*0.5*	*0.3*	*-2.2*	*-0.3*	*-0.5*	*1.6*	*-0.70*
P-ED+CL	5.74	2.96	6.01	3.07	6.32	3.27	5.64	2.93	5.86	3.06	6.15	3.16	5.64	2.92	6.64	3.46	6.92	3.55	5.59	2.89	5.61	2.94	5.85	3.01	
Imp(%)	*1.9*	*4.2*	*1.6*	*2.8*	*1.4*	*2.4*	*0.5*	*0.7*	*3.9*	*6.4*	*2.2*	*3.7*	*1.7*	*3.0*	*-2.8*	*-5.2*	*0.4*	*3.0*	*1.9*	*2.4*	*-0.5*	*-0.3*	*0.5*	*2.9*	*1.62*
P-ED-CL	5.65	2.93	5.94	3.07	6.32	3.32	5.63	2.92	5.83	3.04	6.14	3.23	5.55	2.87	6.46	3.33	6.94	3.60	5.57	2.88	5.60	2.87	5.77	2.99	
Imp(%)	*3.4*	*5.2*	*2.8*	*2.8*	*1.4*	*0.9*	*0.7*	*1.0*	*4.4*	*7.0*	*2.4*	*1.5*	*3.3*	*4.7*	*0.0*	*-1.2*	*0.1*	*1.6*	*2.3*	*2.7*	*-0.4*	*2.0*	*1.9*	*3.5*	*2.26*
ERL	**5.58**	**2.89**	**5.83**	**3.00**	**6.14**	**3.22**	**5.53**	**2.84**	**5.74**	**2.99**	**6.00**	**3.14**	**5.51**	**2.84**	**6.20**	**3.19**	**6.75**	**3.53**	**5.50**	**2.82**	**5.49**	**2.82**	**5.70**	**2.95**	
Imp(%)	*4.6*	*6.5*	*4.6*	*5.1*	*4.2*	*3.9*	*2.5*	*3.7*	*5.9*	*8.6*	*4.6*	*4.3*	*4.0*	*5.6*	*4.0*	*3.0*	*2.9*	*3.6*	*3.5*	*4.7*	*1.6*	*3.8*	*3.1*	*4.8*	*4.29*
Household Electricity Consumption																									
Informer*	2.11	1.57	2.45	1.78	2.95	2.22	2.12	1.55	2.12	1.58	2.34	1.73	2.20	1.62	2.45	1.82	2.41	1.81	1.99	1.47	2.35	1.74	2.29	1.70	
P-ED	2.00	1.46	2.33	1.75	**2.57**	**1.95**	1.99	1.45	2.11	1.55	2.09	1.54	2.09	**1.51**	2.26	1.68	2.71	2.05	2.02	1.47	2.07	1.57	2.24	1.65	
Imp(%)	*5.2*	*7.0*	*4.9*	*1.7*	*12.9*	*12.2*	*6.1*	*6.5*	*0.5*	*1.9*	*10.7*	*11.0*	*5.0*	*6.8*	*7.8*	*7.7*	*-12.4*	*-13.3*	*-1.5*	*0.0*	*11.9*	*9.8*	*2.2*	*2.9*	*4.47*
P-CL	1.97	1.47	2.35	1.72	2.79	2.11	2.02	1.47	2.07	1.53	2.11	1.55	2.22	1.62	2.27	1.69	**2.33**	**1.74**	2.09	1.53	2.08	1.56	2.21	1.64	
Imp(%)	*6.6*	*6.4*	*4.1*	*3.4*	*5.4*	*5.0*	*4.7*	*5.2*	*2.4*	*3.2*	*9.8*	*10.4*	*-0.9*	*0.0*	*7.3*	*7.1*	*3.3*	*3.9*	*-5.0*	*-4.1*	*11.5*	*10.3*	*3.5*	*3.5*	*4.46*
P-ED+CL	2.00	1.46	2.35	1.75	2.85	2.16	**1.96**	**1.43**	2.11	1.56	**2.06**	**1.52**	**2.06**	**1.51**	2.28	1.70	2.46	1.84	2.02	1.48	2.06	1.56	2.23	1.64	
Imp(%)	*5.2*	*7.0*	*4.1*	*1.7*	*3.4*	*2.7*	*7.5*	*7.7*	*0.5*	*1.3*	*12.0*	*12.1*	*6.4*	*6.8*	*6.9*	*6.6*	*-2.1*	*-1.7*	*-1.5*	*-0.7*	*12.3*	*10.3*	*2.6*	*3.5*	*4.78*
P-ED-CL	**1.94**	**1.43**	**2.25**	1.67	2.60	1.97	2.00	1.45	2.07	1.52	2.07	1.53	2.14	1.56	**2.23**	**1.66**	2.47	1.83	2.03	1.47	**2.04**	**1.53**	2.20	1.62	
Imp(%)	*8.1*	*8.9*	*8.2*	*6.2*	*11.9*	*11.3*	*5.7*	*6.5*	*2.4*	*3.8*	*11.5*	*11.6*	*2.7*	*3.7*	*9.0*	*8.8*	*-2.5*	*-1.1*	*-2.0*	*0.0*	*13.2*	*12.1*	*3.9*	*4.7*	*6.18*
ERL	1.96	1.44	**2.25**	**1.65**	2.62	1.97	2.00	1.45	**2.03**	**1.50**	2.10	1.53	2.13	1.54	2.24	**1.66**	2.40	1.78	**1.98**	**1.44**	2.10	1.55	**2.18**	**1.60**	
Imp(%)	*7.1*	*8.3*	*8.2*	*7.3*	*11.2*	*11.3*	*5.7*	*6.5*	*4.2*	*5.1*	*10.3*	*11.6*	*3.2*	*4.9*	*8.6*	*8.8*	*0.4*	*1.7*	*0.5*	*2.0*	*10.6*	*10.9*	*4.8*	*5.9*	*6.62*

Table 3: Comparison of the different versions' performance for the considered datasets with different Missing Data Pattern (MP) and different Missing Data Rate (MR).

loss of Eqn. (4.8) (referred as P-ED+CL), and the fusion following Eqn. (4.9) (referred as P-ED-CL). For P-ED+CL, we set $\alpha = 0.01$ so that each loss has the same order of magnitude. In this table, the error reduction (or increase) of the considered version derived from our proposal is also compared to the Informer* performance (cf. rows ***Imp(%)***). The positive [resp. negative] value indicates the percentage decrease [resp. increase] of the considered RMSE (or MAE).

We observe: (1) P-ED is better than Informer* for all datasets, and (2) P-CL is better than Informer* for SE and HEC. These results show the effectiveness of leveraging the DCD in our framework. (3) P-CL is worse than Informer* for PB, especially for high MRs. The likely reason behind this trend is that, the lack of pattern similarities makes it difficult to accurately see the difference between positive and negative pairs. (4) Both P-ED+CL and P-ED-CL beat Informer* for all datasets. However, without careful tuning of α, P-ED+CL is worse than P-ED-CL. (5) ERL achieves the best performance improvement, in average by 5.21%.

5.4 Predicted samples Figure 5 plots some predicted results in order to better appreciate the improvements achieved by ERL. For these visualizations, we consider MP3 with an MR of 0.8 using all of the three datasets. Each row represents a variable of the MTS. Plots on the left [resp. right] part of the blue vertical line illustrate the inputs [resp. outputs] of the models. For SE, we observe that the peak shape predicted by ERL (red curve) is closer to the ground truth. Besides, for the "zero electricity generation" at night, ERL forecasts fewer fluctuations compared to the imputed-Informer (dark blue curve). For PB, we can also notice that the predicted curves of ERL have a better match with the ground truth. For HEC, ERL performs better during morning and evening (when patterns are more regular). But, it is slightly worse in daytime, where there might be more fluctuations.

All these plots show that ERL manages to better grasp the pattern despite a high MR. They also confirm that SE's and HEC's variables have more similar shapes compared to those of PB.

5.5 Further discussion and limitations First, our experiments validate that Informer with simple imputation techniques performs better than methods with a specific mechanism for handling missing data. In particular, when several consecutive data points are missing, it is more difficult to accurately interpolate these values.

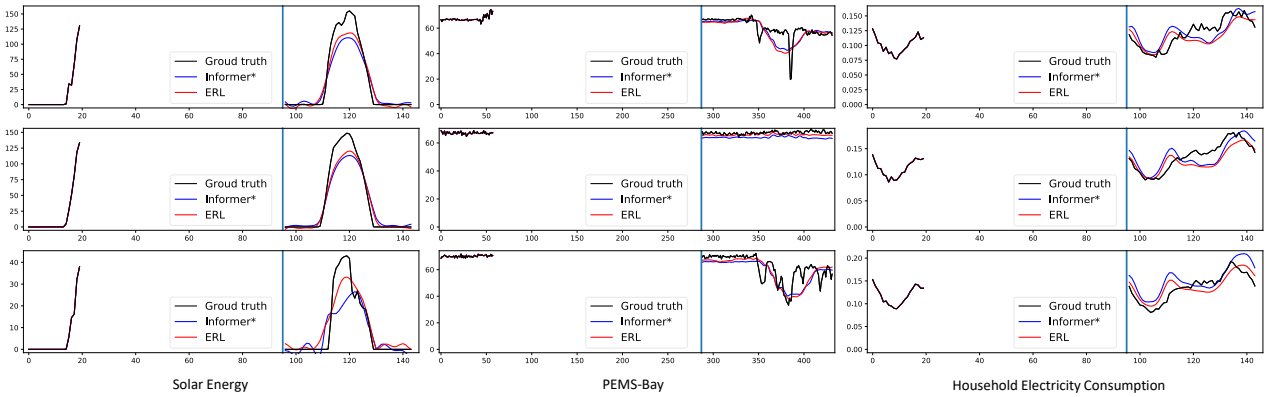

Figure 5: Examples of predicted samples for three variables on three datasets (each row represents a variable).

In such scenarios, complex interpolation techniques are required, but a zero-imputation associated with an attention mechanism can already help identify these missing blocks and probably give them less attention.

As shown in Figure 5, all three MTS datasets have more or less spatial dependencies among their variables. Such correlations can benefit the model when data are missing in one or several variables. In addition, in Table 2, if we focus on Informer-based results (baselines and our proposal), we can observe that despite the MR, results for MP2 are better than those of MP1. This observation shows that Informer can benefit from the spatial correlations between variables in order to account for the missing data. Such a trend is true for all considered datasets and especially for high MR. However, one limitation of our study is actually the selected datasets. Indeed, our experiments focus on spatial MTS, and even if it is clear that models benefit from spatial correlation, it prevents from fully perceive the impact of our framework. Therefore, in the future, we will investigate its performance with univariate and generic multivariate time series datasets.

Additionally, when comparing the efficiency of the Euclidean Distance (ED) and Contrastive Learning (CL) versions, we noticed that ED usually has a greater effect for reducing the error. This trend is especially true for PB. As previously mentioned, sudden changes reduce the pattern similarities among variables, which may make the defined positive pairs dissimilar and close to being negative. We therefore assume that the spatial dependencies associated with our definition of the positive and negative pairs are influencing the effect of CL in the proposed framework. As a consequence, more complex definitions of positive and negative pairs could increase the efficiency of CL. Such complex definitions will be explored in the future in order to better account for the impact of CL on our framework.

Finally, our current definition of positive and negative pairs does not account for the periodicity of the dataset. For example, in the case of HEC that has a strong weekly pattern, the samples of one day and the same day a week later could be considered as positive pairs. We hypothesize that such a revised definition might help CL have superiority over ED for any MPs and MRs. However, it supposes that the model is aware of the major periodicity of the dataset. In the future, we will conduct some experiments in order to determine the feasibility of such an idea.

6 Conclusion

In this paper, when performing time-delayed Multivariate Time Series (MTS) predictions, we proposed to use the representations of delayed complete data (DCD) to assist the representation learning of real-time available data (RAD) with representation alignment. Then, we connected the representation block of RAD with the prediction block of DCD, and fine-tuned the prediction block to perform predictions using only RAD. We investigated two types of representation alignment: distance metric and contrastive learning. In our experiments, we adopted Informer as our base model due to its superiority to predict with incomplete data. Our experimental results confirm that our proposal with either of the two alignments can achieve a better performance than Informer with few exceptions. Moreover, simply fusing the results of both alignments further improved the performance, and combining it with the results of Informer obtains the best performance. In the future, we will mainly focus on more effective methods for generating the positive and negative pairs in contrastive learning.

References

[1] H. Akatsuka, Y. Kamata, T. Nagata, N. Komiya, M. Goto, and M. Terada, *Traffic Dispersion by Predicting Traffic Conditions based on Population Distribution*, in IEEE Big Data, 2021.

[2] F. Bashir and H.-L. Wei, *Handling missing data in multivariate time series using a vector autoregressive model-imputation (var-im) algorithm*, Neurocomputing, (2018).

[3] A. Bhattacharjee, M. Karami, and H. Liu, *Text transformations in contrastive self-supervised learning: A review*, in IJCAI, 2022.

[4] W. Cao, D. Wang, J. Li, H. Zhou, L. Li, and Y. Li, *Brits: Bidirectional recurrent imputation for time series*, in NeurIPS, 2018.

[5] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, *Recurrent neural networks for multivariate time series with missing values*, Scientific reports, (2018).

[6] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, *Neural ordinary differential equations*, in NeurIPS, 2018.

[7] W. Du, D. Cote, and Y. Liu, *SAITS: Self-Attention-based Imputation for Time Series*, arXiv, (2022).

[8] H. Fang, S. Wang, M. Zhou, J. Ding, and P. Xie, *Cert: Contrastive self-supervised learning for language understanding*, arXiv, (2020).

[9] T. Gao, X. Yao, and D. Chen, *Simcse: Simple contrastive learning of sentence embeddings*, in EMNLP, 2021.

[10] A. Gillioz, J. Casas, E. Mugellini, and O. A. Khaled, *Overview of the Transformer-based Models for NLP Tasks*, in FedCSIS, 2020.

[11] G. Habault, S. Wada, R. Kimura, and C. Ono, *Elucidating the extent by which population staying patterns help improve electricity load demand predictions*, in IEEE Big Data, 2020.

[12] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, *Supervised contrastive learning*, in NeurIPS, 2020.

[13] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, *Modeling long-and short-term temporal patterns with deep neural networks*, in ACM SIGIR, 2018.

[14] P. H. Le-Khac, G. Healy, and A. F. Smeaton, *Contrastive representation learning: A framework and review*, IEEE Access, (2020).

[15] Y. Li, R. Yu, C. Shahabi, and Y. Liu, *Diffusion convolutional recurrent neural network: Data-driven traffic forecasting*, in ICLR, 2018.

[16] S. Liu, H. Yu, C. Liao, J. Li, W. Lin, A. X. Liu, and S. Dustdar, *Pyraformer: low-complexity pyramidal attention for long-range time series modeling and forecasting*, in ICLR, 2022.

[17] A. L'Heureux, K. Grolinger, and M. A. M. Capretz, *Transformer-based model for electrical load forecasting*, Energies, (2022).

[18] X. Miao, Y. Wu, J. Wang, Y. Gao, X. Mao, and J. Yin, *Generative semi-supervised learning for multivariate time series imputation*, in AAAI, 2021.

[19] NREL. `https://www.nrel.gov/grid/solar-power-data.html`. Accessed: 2021-10-01.

[20] Y. Qian, L. Tian, B. Zhai, S. Zhang, and R. Wu, *Informer-WGAN: High Missing Rate Time Series Imputation Based on Adversarial Training and a Self-Attention Mechanism*, Algorithms, (2022).

[21] Y. Rubanova, R. Chen, and D. Duvenaud, *Latent odes for irregularly-sampled time series*, in NeurIPS, 2019.

[22] S. N. Shukla and B. M. Marlin, *Interpolation-prediction networks for irregularly sampled time series*, in ICLR, 2019.

[23] S. N. Shukla and B. M. Marlin, *Multi-time attention networks for irregularly sampled time series*, in ICLR, 2021.

[24] S. Tonekaboni, D. Eytan, and A. Goldenberg, *Unsupervised representation learning for time series with temporal neighborhood coding*, in ICLR, 2021.

[25] H. Touvron, M. Cord, A. Sablayrolles, G. Synnaeve, and H. Jegou, *Going deeper with Image Transformers*, in ICCV, 2021.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, *Attention is all you need*, in NeurIPS, 2017.

[27] C. Wang, Y. Yu, W. Ma, M. Zhang, C. Chen, Y. Liu, and S. Ma, *Towards representation alignment and uniformity in collaborative filtering*, in ACM SIGKDD, 2022.

[28] T. Wang and P. Isola, *Understanding contrastive representation learning through alignment and uniformity on the hypersphere*, in ICML, 2020.

[29] G. Woo, C. Liu, D. Sahoo, A. Kumar, and S. Hoi, *Cost: Contrastive learning of disentangled seasonal-trend representations for time series forecasting*, in ICLR, 2022.

[30] H. Wu, J. Xu, J. Wang, and M. Long, *Autoformer: decomposition transformers with auto-correlation for long-term series forecasting*, in NeurIPS, 2021.

[31] J. Wu, X. Wang, F. Feng, X. He, L. Chen, J. Lian, and X. Xie, *Self-supervised graph learning for recommendation*, in ACM SIGIR, 2021.

[32] Z. Yue, Y. Wang, J. Duan, T. Yang, C. Huang, Y. Tong, and B. Xu, *Ts2vec: Towards universal representation of time series*, in AAAI, 2022.

[33] J. Yun, C. Joo, and A. Eryilmaz, *Optimal Real-Time Monitoring of an Information Source Under Communication Costs*, in IEEE CDC, 2018.

[34] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, *Informer: Beyond efficient transformer for long sequence time-series forecasting*, in AAAI, 2021.

[35] Y. Zhou, K. Zhou, W. X. Zhao, C. Wang, P. Jiang, and H. Hu, *C^2-crs: Coarse-to-fine contrastive learning for conversational recommender system*, in WSDM, 2022.